МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВОЛИНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЛЕСІ УКРАЇНКИ

Географічний факультет
Кафедра геодезії, землевпорядкування та кадастру

ГЕОПРОСТОРОВІ БАЗИ ДАНИХ. КОНСПЕКТ ЛЕКЦІЙ.

рівень вищої освіти	другий (магістерський)
галузь знань	19 Архітектура та будівництво
спеціальність	193 Геодезія та землеустрій
освітня професійна програма	Геодезія та землеустрій

Луцьк – 2022

УДК 911:114]:004.65(07)
М 48
Рекомендовано до друку методичною радою Волинського національного університету імені Лесі Українки (протокол № 10 від 21 червня 2022 р.)

Рецензенти:
Барський Ю.М., доктор економічних наук, професор, декан географічного факультету,
Мельник Ю.А., кандидат технічних наук, доцент, доцент кафедри будівництва та цивільної інженерії Луцького НТУ

М54
Геопросторові бази даних. Конспект лекцій: навч.-метод. вид. для студентів геогр. ф-ту / О.В. Мельник. – Луцьк: РВВ "Вежа-Друк", 2022. – 156 с.

Навчально-методичне видання призначене для використання студентами при вивченні дисципліни "Геопросторові бази даних" за освітньо-професійною програмою Геодезія та землеустрій спеціальності 193 Геодезія та землеустрій галузі знань 19 Архітектура та будівництво денної та заочної форм навчання.

У даному навчально-методичному виданні сформульовано загальні положення, історію, структуру, вимоги до геопросторових баз даних, а також розглянуті сучасні технології обробки геопросторової інформації, моделі, що лежать у їх основі, сучасні напрями застосування баз геопросторових даних і перспективи розвитку

528.4:[378.22:001.817(072)
М 54

ЗМІСТ

ВСТУП

Одним із компонентів програмного забезпечення ГІС є бази даних, де міститься просторова і атрибутивна інформація. Збереження та накопичення даних є однією з основних дій, здійснюваних над інформацією, яка забезпечує її доступність протягом певного проміжку часу.

Збором і накопиченням даних, коригуванням, сортуванням та відбором необхідних даних тією або іншою мірою займається кожна людина, кожен фахівець, незалежно від сфери своєї діяльності. Протягом життя людина накопичує різноманітні дані: телефони, адреси, імена та дати народження друзів і знайомих; відомості про те, які книги зберігаються у власній бібліотеці, які ігри, аудіо- та відеозаписи знаходяться на дисках тощо. Для забезпечення зручного користування цими даними їх необхідно зберігати в систематизованому вигляді.

Для того щоб інформація, яка накопичена окремою людиною або суспільством, була доступною для подальшого використання, її зберігають у систематизованому вигляді в спеціальних сховищах - базах даних.

ЛЕКЦІЯ 1
ЗАГАЛЬНІ ВІДОМОСТІ ПРО СИСТЕМИ КЕРУВАННЯ БАЗАМИ ДАНИХ

1.1. Передумови виникнення концепції баз даних

Незважаючи на суттєві досягнення сучасних систем керування базами даних (СКБД), що здійснені за декілька останніх десятиліть, треба відзначити, що обробка інформації головним чином з метою обліку ресурсів має історію в декілька тисяч років. Проте обробка цієї інформації здійснювалась переважно вручну. І тільки на початку XX ст. з'явилась можливість її опрацьовувати автоматизовано, за допомогою перфокарт.

В історії обчислювальної техніки можна прослідкувати розвиток двох основних напрямків її використання.

Перший напрямок - застосування обчислювальної техніки для виконання числових розрахунків, які або дуже довго або взагалі неможливо проводити вручну. Розвиток цього напрямку сприяв інтенсифікації методів числового розв'язку складних математичних завдань, появі мов програмування, які були розраховані виключно на обробку числової інформації (Fortran, Algol) , орієнтованих на зручний запис числових алгоритмів, становленню зворотного зв'язку з розробниками нової архітектури ЕОМ. Характерною особливістю даного напрямку застосування обчислювальної техніки є наявність складних алгоритмів обробки, які застосовуються до простих за структурою даних, об'єм яких порівняно невеликий.

Другий напрямок - це використання засобів обчислювальної техніки в автоматичних або автоматизованих інформаційних системах (ІС) . Зазвичай такі системи мають справу з великими обсягами інформації, що мають доволі складну структуру. Класичними прикладами ІС є банківські системи, автоматизовані системи управління (АСУ) підприємствами, системи резервування авіаційних або залізничних квитків, місць у готелях тощо.

Даний напрямок використання обчислювальної техніки виник трохи пізніше першого. Це пов'язано з тим, що на зорі розвитку обчислювальної техніки можливості комп'ютерів зі збереження інформації були доволі обмеженими. Дані подавались у вигляді простих послідовних файлів на магнітній стрічці, були частиною програм і розташовувались відразу за кодом програми в так званому сегменті даних (рис. 1.1), тобто залежали від програм обробки.

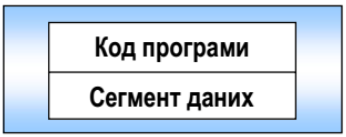


Рис. 1.1. Дані як частина програм

Якщо змінювалась організація даних або тип запам'ятовуючого пристрою, програмісту доводилось заново переписувати програму, що призводило до численних версій одного й того ж файлу, а отже, й до високого ступеня дублювання даних, їх надлишковості.

Наступним кроком стало збереження даних в окремих файлах (рис. 1.2).

Рис. 1.2. Збереження даних в окремих файлах

Прикладні програми самі визначали розташування даних на магнітній стрічці чи барабані й здійснювали обмін інформацією між оперативною та зовнішньою пам'яттю за допомогою програмно-апаратних засобів

Інформаційна система (англ. information system) - сукупність організаційних і технічних засобів для збереження та обробки інформації з метою забезпечення інформаційних потреб користувачів низького рівня (машинних команд або викликів відповідних програм операційної системи).

Такий спосіб роботи не дозволяв або дуже ускладнював підтримку на одному зовнішньому носієві кількох архівів інформації, якщо вони мали зберігатися тривалий час. Крім того, для кожної прикладної програми доводилося вирішувати проблеми іменування частин даних та їхньої структуризації в зовнішній пам'яті.

Недоліком цих двох підходів була залежність програм від даних: відомості про структуру даних включались до коду програми. При зміні структури даних виникала потреба внесення змін у програмі.

Логічним продовженням цієї еволюції є перенесення опису даних до масиву даних (рис. 1.3). Це дозволило забезпечити незалежність даних від програм.

Рис. 1.3. Забезпечення незалежності даних від програм

Таким чином склалося два підходи до організації інформаційних масивів:

- файлова організація інформаційних масивів;
- організація інформаційних масивів у вигляді бази даних.

1.1.1. Файлова організація масивів даних

Файл - це іменована частина зовнішньої пам'яті, в яку можна записувати і з якої можна зчитувати дані.

Файлова організація масивів даних передбачає спеціалізацію та збереження інформації, орієнтованої, як правило, на одну прикладну задачу, та

забезпечується прикладним програмістом. Така організація дозволяє досягнути високої швидкості обробки інформації, але характеризується низкою недоліків.

Користувачі бачать файл як лінійну послідовність записів і можуть виконати над ними низку стандартних операцій:

• створити файл необхідного типу і розміру;

• відкрити раніше створений файл;

• зчитати з файлу певний запис (поточний, наступний, попередній, перший, останній);

• записати у файл на місце поточного запису новий, додати новий запис у кінець файлу.

В різних файлових системах ці операції можуть різнитись, але загальний зміст їх буде саме таким. Головне - це те, що структура запису файлу відома тільки програмі, яка з ним працює, система керування файлами її не знає. І тому для того, щоб здобути певну інформацію з файлу, необхідно точно знати структуру запису файлу з точністю до біта. Кожна програма, що працює з файлом, повинна містити в собі структуру даних, відповідну структурі цього файлу. Тому при зміні структури файлу потрібно було змінювати структуру програми, а це вимагало нової компіляції, тобто процесу перекладу програми у виконувані машинні коди. Така ситуація характеризується як залежність програм від даних.

Характерна риса файлового підходу - вузька спеціалізація як програм обробки, так і файлів даних, що є причиною великої надлишковості (одні й ті ж елементи даних зберігаються в різних системах). Оскільки керування здійснюється різними особами (групами осіб), відсутня можливість виявити порушення суперечливості збереженої інформації. Розроблені файли для спеціалізованих прикладних програм не можна використовувати для задоволення запитів користувачів, які перекривають дві і більше предметні сфери.

Предметна сфера - частина реального світу, дані з якого відображені в базі даних.

Наприклад, як предметну сферу можна обрати бухгалтерію підприємства, відділ кадрів, банк, магазин тощо. Предметна сфера нескінченна і містить як істотно важливі поняття і дані, так і малозначущі або взагалі не значущі дані. Так, якщо як предметну сферу обрати облік товарів на складі, то поняття "накладна" і "рахунок-фактура" є істотно важливими поняттями, а те, що співробітниця, яка приймає накладні, має двох дітей - це для обліку товарів неважливо. Однак з точки зору відділу кадрів дані про наявність дітей є вкрай важливими. Таким чином, важливість даних залежить від вибору предметної сфери.

Крім того, файлова організація даних внаслідок відмінностей структури записів і форматів представлення даних не забезпечує виконання багатьох інформаційних запитів навіть у тих випадках, коли всі необхідні елементи даних містяться в наявних файлах. Тому виникає потреба відокремити дані від їхнього

опису, визначити таку організацію збереження даних з урахуванням існуючих зв'язків між ними, яка б дозволила використовувати ці дані одночасно для багатьох додатків (застосувань). Зазначені причини й обумовили появу баз даних.

У комп'ютерах першого покоління використовувалися два види пристроїв зовнішньої пам'яті - магнітні стрічки і барабани. Ємність магнітних стрічок була достатньо велика, але за своєю фізичною природою вони могли забезпечити тільки послідовний доступ до даних. Магнітні ж барабани (вони найближчі до сучасних магнітних дисків з фіксованими головками) давали можливість довільного доступу до даних, але мали обмежений об'єм інформації. Ці обмеження не були істотними для суто числових розрахунків. Навіть якщо програма й повинна була опрацювати великий об'єм інформації, при програмуванні завжди можна продумати розташування цієї інформації в зовнішній пам'яті так, щоб забезпечити ефективне виконання цієї програми.

Однак в ІС потреба користувачів у пошуку інформації вимагає порівняно швидкої реакції системи на їх запити. Саме вимоги нечисленних застосувань в ІС викликали появу знімних магнітних дисків з рухомими головками, що стало революційною подією в історії обчислювальної техніки. Ці пристрої зовнішньої пам'яті мали істотно більшу ємкість на відміну від магнітних барабанів, забезпечували задовільну швидкість доступу до даних у режимі довільної вибірки, а можливість зміни дискового пакету на пристрої дозволяла мати практично необмежений архів даних.

З появою магнітних дисків почалася історія систем керування даними в зовнішній пам'яті. До цього кожна прикладна програма, за допомогою якої потрібно було зберігати дані в зовнішній пам'яті, сама визначала розташування кожної порції даних на магнітній стрічці або барабані і виконувала обмін між оперативною пам'яттю та пристроями зовнішньої пам'яті за допомогою програмно-апаратних засобів низького рівня (машинних команд або викликів відповідних програм операційної системи). Такий режим роботи не дозволяв або дуже ускладнював підтримку на одному зовнішньому носії декількох архівів довготривалого збереження. Крім того, кожній прикладній програмі доводилося вирішувати проблеми іменування частин даних і структуризації даних у зовнішній пам'яті.

Тому важливим кроком у розвитку ІС з'явився перехід до використання централізованих систем управління файлами. Правила іменування файлів, спосіб доступу до даних, що зберігаються в файлі, структура цих даних залежать від конкретної системи керування файлами і від типу файлу. Система керування файлами бере на себе розподіл зовнішньої пам'яті, відображення імен файлів у відповідні адреси в зовнішній пам'яті і забезпечення доступу до даних.

Для ІС характерною є наявність великої кількості різних користувачів (програм), кожен з яких має свої специфічні алгоритми обробки інформації, що зберігаються в одних і тих же файлах.

1.1.2. Необхідність розробки СКБД

Необхідність розробки СКБД викликана такими причинами:

1. Зміна структури файлу, яка необхідна для однієї програми, вимагає виправлення, перекомпіляції й додаткового налагодження решти програм, що працюють з цим файлом. Це перший істотний недолік файлових систем, який стимулював створення нових систем збереження й керування інформацією. Оскільки файлові системи є загальним сховищем файлів, що належать різним користувачам, системи керування файлами повинні забезпечувати авторизацію доступу до файлів. У загальному вигляді підхід полягає в тому, що до кожного зареєстрованого користувача даної обчислювальної системи, для кожного існуючого файлу вказуються дії, які дозволені або заборонені даному користувачеві.

2. Відсутність централізованих методів керування доступом до інформації стала другою причиною розробки СКБД.

3. Третьою причиною стала необхідність забезпечення ефективної паралельної роботи багатьох користувачів з одними й тими ж файлами. В загальному випадку системи керування файлами забезпечували режим багатокористувацького доступу. Якщо операційна система підтримує багатокористувацький режим, цілком реальна ситуація, коли два або більше користувачів одночасно намагаються працювати з одним і тим же файлом. Якщо всі користувачі збираються тільки зчитувати файл, нічого страшного не відбувається. Але якщо хоча б один з них буде змінювати файл, то для коректної роботи цих користувачів потрібна взаємна синхронізація їх дій по відношенню до файлу.

В системах керування файлами зазвичай застосовувався такий підхід. В операції відкриття файлу (першій і обов'язковій операції, з якою повинен починатись сеанс роботи з файлом) серед інших параметрів вказувався режим роботи (читання або зміна). Якщо до моменту виконання цієї операції певним користувацьким процесом PR1 файл був уже відкритий іншим процесом PR2 в режимі зміни, то залежно від особливостей системи процесу PR1 або повідомлялося про неможливість відкриття файлу або він блокувався доти, поки в процесі PR2 не виконувалась операція закриття файлу.

При подібному способі організації одночасна робота декількох користувачів, пов'язаних з модифікацією даних у файлі, або взагалі не буде реалізовуватись, або буде дуже сповільнена.

Ці недоліки стимулювали розробників ІС запропонувати новий підхід - створення баз даних.

1.1.3. Бази і банки даних як засіб збереження даних

У базах даних сукупність взаємопов'язаних даних, що зберігаються разом, організована таким чином, що їх використання є оптимальним для одного або декількох додатків; дані є незалежними від програм, що використовують ці дані; для додавання нових або модифікації існуючих даних, а також для пошуку даних

у БД застосовують загальний спосіб керування. При цьому дані структуруються таким чином, що забезпечується можливість подальшого нарощування додатків.

База даних (БД) - структурований, організований набір даних, що описує характеристики певних фізичних чи віртуальних систем.

Структурування - це введення угод про способи подання даних.

Складно організувати пошук необхідних даних, які зберігаються в неструктурованому вигляді, а впорядкувати подібну інформацію взагалі неможливо.

Підхід був реалізований у рамках нових програмних систем, названих згодом Системами Керування Базами Даних (СКБД), а самі сховища інформації, які працювали під керуванням даних систем, називалися базами або банками даних (БД і БНД).

СКБД - комплекс програмних і мовних засобів, призначених для організації, супроводу та надання доступу до БД користувачам.

Примітка. "Базою даних" доволі часто помилково або з метою спрощення називають СКБД. Однак потрібно розрізняти набір даних (тобто БД) та програмне забезпечення, що призначене для організації та супроводження бази даних.

Банк даних - це система спеціальним чином організованих даних (баз даних), програмних, технічних, мовних, організаційно-методичних засобів, призначених для забезпечення централізованого накопичення і колективного багатоцільового використання даних.

Фактично у будь-якої сучасної СКБД існує аналог, який випускається іншою компанією і який має аналогічну сферу застосування і можливості, будь-який додаток здатний працювати із багатьма форматами подання даних, здійснювати експорт та імпорт даних завдяки наявності великої кількості конвертерів.

Серед найбільш яскравих представників СКБД можна відзначити: Lotus Approach, Microsoft Access, Borland dBase, Borland Paradox, Microsoft Visual FoxPro, Microsoft Visual Basic, а також бази даних Microsoft SQL Server і Oracle, що використовуються в додатках, побудованих за технологією "клієнт - сервер".

Для порівняння ефективності найвідоміших СКБД зазвичай використовують такі критерії оцінки:

- масштабованість;
- продуктивність;
- доступність даних;
- функціональні можливості сервера;
- відкритість;
- наявні засоби розробки.

Загальноприйнятими також є технології, які дозволяють використовувати можливості інших додатків, наприклад, текстових редакторів, пакетів побудови графіків тощо, і вбудовані версії мов високого рівня (найчастіше - діалекти SQL і / або VBA) та засоби візуального програмування інтерфейсів розроблюваних додатків. Тому вже не має істотного значення, якою мовою і на основі якого

пакету був написаний конкретний додаток та який формат даних у ньому використовується. Більш того, стандартом "де-факто" стала "швидка розробка програм", або RAD (від англ. Rapid Application Development), заснована на широко декларованому в літературі "відкритому підході", тобто можливості використання різних прикладних програм і технологій для розробки більш гнучких та потужних систем обробки даних. Тому в одному ряду з "класичними" СКБД все частіше згадуються мови програмування Visual Basic 6.0 і Visual C++, які дозволяють швидко створювати необхідні компоненти додатків, критичні за швидкістю роботи, які важко, а іноді неможливо розробити засобами "класичних" СКБД. Сучасний підхід до керування базами даних передбачає також широке використання технології "клієнт - сервер".

1.2. Етапи розвитку систем керування базами даних

Історія розвитку СКБД нараховує майже 50 років і починається... з польоту на Місяць. Американська компанія NAA (North American Aviation, зараз "Rockwell International") уклала контракт з урядом США на участь у проекті "Apollo".

Оскільки побудова космічного корабля передбачає щонайменше збірку декількох мільйонів деталей, то для цього була створена система керування файлами, яка відслідковувала інформацію про кожну деталь. Проте в ході подальшої перевірки цієї ІС виявилася величезна надмірність файлової системи збереження даних. З'ясувалося, що майже всі дані повторюються в двох і більше файлах. Зіткнувшись із завданням координації замовлень на мільйони деталей, компанія Rockwell у співпраці з IBM у 1968 р. розробила автоматизовану систему замовлень, яка одержала назву IMS (Information Management System - система керування інформацією). Саме вона заклала основу концепції СКБД.

Ключовим нововведенням IMS був поділ даних і функцій ділової логіки. Прикладні програмісти отримали можливість працювати з інформацією на логічному рівні, а база даних брала на себе завдання фізичного збереження даних. Подібний розподіл праці привів до різкого стрибка продуктивності роботи компанії.

Ще одним винаходом стала мова програмування DL/I (Data Language/I). Це була спеціалізована мова складання нерегламентованих запитів до бази даних. Її поява зробила непотрібним коштовне програмування на таких мовах, як COBOL і FORTRAN, популярних на той час.

Незважаючи на те, що IMS є першою з комерційних СКБД, вона до сьогодення залишається основною ієрархічною СКБД, що використовується на більшості великих мейнфреймів. У ній реалізована ієрархічна модель даних, у якій існує тільки один шлях від кореня ієрархії до кожного запису. Така модель стала основою для систем керування даними, вона ж дала поштовх до подальших моделей даних через свою обмеженість.

У 1971 р. відбулася конференція з мов обробки даних (Conference on Data Systems Languages, CODASYL), метою якої була розробка стандартів баз даних. Попередньо ця конференція вже стандартизувала мову COBOL.

Новий стандарт визначив низку фундаментальних понять в теорії систем баз даних, які і до сьогодення є основоположними для мережевої моделі даних.

У мережевій моделі будь-який запис може брати участь у декількох відносинах предок / нащадок. Це дозволяло обходити цілу низку обмежень ієрархічної моделі. Розробкою мережевої моделі займався Чарльз Бахман - керівник проекту IDS (Integrated Data System - інтегрована система обробки даних) у компанії General Electric.

Тим часом науковий співробітник компанії IBM доктор Едгар Кодд працював над епохальним документом для Асоціації виробників обчислювальної техніки (Association for Computing Machinery, ACM).

У червні 1970 р. цей документ був опублікований в ACM Journal під назвою "Реляційна модель для великих банків спільно використовуваних даних" ("A Relational Model of Data for Large Shared Data Banks"), який докорінно змінив теорію баз даних і приніс доктору Кодду премію Тюрінга в 1981 р.

Доктор Кодд запропонував реляційну модель, у якій дані можна було вільно описувати в їх природному вигляді без будь-яких обмежень, що накладаються середовищем фізичного збереження.

Фундаментальним поняттям реляційної БД є відношення. Це відображено і в загальній назві підходу - термін реляційний (relational) походить від англ. relation (відношення).

На фізичному рівні відношення представляють собою таблиці, розбиті на рядки і стовпці (рис. 1.4).

Рис. 1.4. Реляційна модель Е. Кодда

Це дозволило створити мову високого рівня, здатну працювати з даними незалежно від того, як вони зберігаються в комп'ютері.

Як наслідок, з'явилися дві СКБД: System R компанії IBM і Ingres Каліфорнійського університету в Берклі. В обох СКБД була реалізована реляційна модель і мова запитів. Остання в СКБД System R спочатку називалась

SEQUEL (Structured English Query Language - структурована англійська мова запитів). Пізніше з'явилася назва SQL (Structured Query Language). У 1986 р. організація ANSI опублікувала офіційний стандарт мови SQL.

Стрімкий розвиток обчислювальної техніки, зміна її принципової ролі в житті суспільства, лавиноподібне зростання можливостей персональних комп'ютерів, поява потужних робочих станцій і комп'ютерних мереж, безумовно, вплинуло на розвиток технологій баз даних. При цьому можна виділити чотири етапи в розвитку технологій обробки даних. Однак необхідно відзначити, що на цих етапах відсутні жорсткі часові обмеження: технології плавно переходять з одного етапу в інший і навіть співіснують паралельно, проте виділення цих етапів дозволяє більш чітко охарактеризувати окремі стадії розвитку технології баз даних, підкреслити особливості, характерні для конкретного етапу.

Перший етап розвитку СКБД пов'язаний з організацією баз даних на великих машинах типу IBM 360/370, ЄС-ЕОМ і міні-ЕОМ типу PDP- 11 (фірми Digital Equipment Corporation - DEC), різних моделях HP (фірми Hewlett Packard).

Бази даних зберігалися в зовнішній пам'яті центральної ЕОМ, користувачами цих баз даних були завдання, що запускались головним чином у пакетному режимі. Інтерактивний режим доступу забезпечувався за допомогою консольних терміналів, які не володіли власними обчислювальними ресурсами (процесором, зовнішньою пам'яттю) і слугували тільки пристроями введення-виведення для центральної ЕОМ. Програми доступу до БД писалися на різних мовах і запускалися як звичайні програми для числових розрахунків. Потужні операційні системи забезпечували можливість умовно-паралельного виконання всієї безлічі завдань. Ці системи можна було віднести до систем розподіленого доступу, тому що база даних була централізованою, зберігалася на пристроях зовнішньої пам'яті однієї центральної ЕОМ, а доступ до неї підтримувався від багатьох користувачів (завдань).

Особливості першого етапу розвитку СКБД:

• усі СКБД базуються на потужних мультипрограмних операційних системах (MVS SvM, RTE, OSRV, RsX, UNIX), тому в основному підтримується робота з централізованою базою даних у режимі розподіленого доступу;

• функції керування розподілом ресурсів здійснюються головним чином операційною системою (ОС);

• підтримуються мови низького рівня маніпулювання даними, які орієнтовані на навігаційні методи доступу до даних;

• значна роль відводиться адмініструванню даних;

• здійснюються серйозні роботи з обґрунтування і формалізації реляційних моделей даних, була створена перша система (System R), яка реалізувала ідеологію реляційної моделі даних;

• проводяться теоретичні роботи з оптимізації запитів і керування розподіленим доступом до централізованої БД, було введено поняття транзакції;

• результати наукових досліджень відкрито обговорюються у пресі, йде потужний потік загальнодоступних публікацій, що стосуються всіх аспектів теорії і практики баз даних, і результати теоретичних досліджень активно впроваджуються в комерційні СКБД;

• з'являються перші мови високого рівня для роботи з реляційною моделлю даних;

• відсутні стандарти для цих мов.

Другий етап розвитку СКБД пов'язаний з появою персональних комп'ютерів, які стрімко увірвались в життя світової спільноти і перевернули всі уявлення про місце та роль обчислювальної техніки в житті суспільства. Комп'ютери стали ближчими і доступнішими кожному користувачеві. Зник страх пересічних користувачів перед незрозумілими і складними мовами програмування. З'явились безліч програм, призначених для роботи непідготовлених користувачів. Ці програми були прості у використанні та інтуїтивно зрозумілі: редактори текстів, електронні таблиці тощо. Простими і зрозумілими стали операції копіювання файлів та перенесення інформації з одного комп'ютера на інший, друкування текстів, таблиць й інших документів. Системні програмісти були відсунуті на другий план. Кожен користувач міг відчути себе цілковитим господарем цього потужного і зручного пристрою, який дозволяв автоматизувати більшість аспектів діяльності людини.

Це позначилося і на роботі з базами даних. З'явилися програми, які називались СКБД і дозволяли зберігати значні об'єми інформації, мали зручний інтерфейс для заповнення баз даних, убудовані засоби для генерації різних звітів. Ці програми дозволяли автоматизувати більшість облікових функцій, які раніше велися вручну.

Постійне зниження цін на персональні комп'ютери зробило їх доступними не тільки для організацій і фірм, а й для окремих користувачів. Комп'ютери стали інструментом для ведення документації і власних облікових функцій. Це все відіграло як позитивну, так і негативну роль у розвитку баз даних.

Значна конкуренція серед постачальників програмного забезпечення примушувала вдосконалювати ці системи, пропонувати нові можливості, які покращували інтерфейс і швидкодію систем, знижували їх вартість. Наявність на ринку великої кількості СКБД, що виконували схожі функції, вимагало розробки методів експорту-імпорту даних для цих систем і відкриття форматів збереження даних. Але в цей період з'являються дилетанти-любителі, які всупереч здоровому глузду розробляли власні СКБД, використовуючи стандартні мови програмування.

Удавана простота і доступність персональних комп'ютерів та їхнього програмного забезпечення породила безліч дилетантів. Ці розробники, вважаючи себе знавцями, стали проектувати недовговічні бази даних, які не враховували багатьох особливостей об'єктів реального світу. Було створено багато систем-одноденок, які не відповідали законам розвитку і взаємозв'язку реальних об'єктів. Однак доступність персональних комп'ютерів змусила

користувачів з багатьох галузей знань, які раніше не застосовували обчислювальну техніку в своїй діяльності, звернутися до них. І попит на розвинені зручні програми обробки даних примушував постачальників програмного забезпечення постачати все нові системи, які прийнято називати настільними (desktop) СКБД. Це був тупиковий варіант, тому що подальший розвиток засвідчив, що перенести дані з нестандартних форматів у нові СКБД або дуже складно, або взагалі неможливо.

Особливості другого етапу розвитку СКБД:

• усі СКБД розраховувались на створення БД з монопольним доступом (комп'ютер персональний, не під'єднаний до мережі), тому база даних на ньому створювалася для роботи одного користувача. В деяких випадках передбачалася послідовна робота декількох користувачів, наприклад, спочатку працював оператор, який вводив бухгалтерські документи, а потім бухгалтер, який визначав проводки, що відповідали введеним первинним документам;

• більшість СКБД мали розвинений і зручний для користувача інтерфейс, існував інтерактивний режим роботи з БД, як в рамках опису БД, так і в рамках проектування запитів. СКБД пропонували розвинений і зручний інструментарій для розроблення готових додатків без програмування. Інструментальне середовище складалось з готових елементів додатку у вигляді шаблонів екранних форм, звітів, етикеток (labels), графічних конструкторів запитів, які досить просто могли бути зібрані в єдиний комплекс;

• у всіх настільних СКБД підтримувався тільки зовнішній рівень подання реляційної моделі, тобто тільки зовнішній, табличний вигляд структур даних;

• за наявності високорівневих мов маніпулювання даними типу реляційної алгебри і SQL у настільних СКБД підтримувалися низько- рівневі мови маніпулювання даними на рівні окремих рядків таблиць;

• у настільних СКБД були відсутні засоби підтримки посилальної і структурної цілісності бази даних. Ці функції повинні були виконувати додатки, проте обмеженість засобів розробки додатків іноді не дозволяла це зробити. В цьому разі ці функції повинні були виконуватися користувачем, вимагаючи від нього додаткового контролю при введенні і зміні інформації, що зберігається в БД;

• наявність монопольного режиму роботи фактично призвела до виродження функцій адміністрування БД і у зв'язку з цим - до відсутності інструментальних засобів адміністрування БД;

• порівняно скромні вимоги до апаратного забезпечення з боку настільних СКБД. Цілком працездатні додатки, розроблені, наприклад, на Clipper, працювали на PC 286. Яскравими представниками цього сімейства є СКБД Dbase (Dbase III+, Dbase IV), FoxPro, Clipper, Paradox, які широко використовувалися до недавнього часу.

Третій етап розвитку СКБД пов'язаний з появою розподілених баз даних. Добре відомо, що історія розвивається по спіралі, тому після процесу "персоналізації" почався зворотний процес - інтеграція. Збільшується кількість

локальних мереж, все більше інформації циркулює між комп'ютерами, гостро постає проблема узгодженості даних, що зберігаються і обробляються в різних місцях, але логічно пов'язаних, виникають завдання, пов'язані з паралельною обробкою транзакцій - послідовностей операцій над БД, переводять з одного несуперечливого стану в інший несупереч- ливий стан. Успішне розв'язання цих завдань приводить до появи розподілених баз даних, що зберігають усі переваги настільних СКБД і в той же час дозволяють організувати паралельну обробку інформації та підтримку цілісності БД.

Особливості третього етапу розвитку СКБД:

- практично всі СКБД забезпечують підтримку повної реляційної моделі, а саме:
- структурної цілісності - допустимими є тільки дані, подані у вигляді відношень реляційної моделі;
- мовної цілісності, тобто мов маніпулювання даними високого рівня (в основному SQL);
- посилальної цілісності, контролю за дотриманням посилальної цілісності протягом усього часу функціонування системи і гарантій неможливості з боку СКБД порушити ці обмеження;
- більшість сучасних СКБД розраховані на багатоплатформність архітектури, тобто вони можуть працювати на комп'ютерах з різною архітектурою і під різними операційними системами, при цьому для користувачів доступ до даних, що керуються СКБД на різних платформах, практично непомітний;
- необхідність підтримки одночасної роботи багатьох користувачів з базою даних і можливість децентралізованого збереження даних вимагали розвитку засобів адміністрування БД з реалізацією загальної концепції засобів захисту даних;
- потреба в нових реалізаціях зумовила створення серйозних теоретичних праць з оптимізації реалізацій розподілених БД і роботу з розподіленими транзакціями та запитами з впровадженням отриманих результатів у комерційні СКБД;
- для того щоб не втратити клієнтів, які раніше працювали на настільних СКБД, практично всі сучасні СКБД мають засоби підключення клієнтських додатків, розроблених з використанням настільних СКБД, і засоби експорту даних з форматів настільних СКБД другого етапу розвитку;
- саме до цього етапу можна віднести розробку низка стандартів в рамках мов опису і маніпулювання даними, починаючи з SQL89, SQL92, SQL99 і технологій з обміну даними між різними СКБД, до яких можна віднести і протокол ODBC (Open DataBase Connectivity), запропонований фірмою Microsoft;
- саме до цього етапу можна віднести початок робіт, пов'язаних з концепцією об'єктно орієнтованих БД - ООБД. Представниками СКБД, що відносяться до третього етапу, можна вважати MS Access і всі сучасні сервери баз даних Oracle7.3, Oracle 8.4, MS SQL6.5, MS SQL7.0, System 10, System 11, Informix,

DB2, SQL Base та інші сучасні сервери баз даних, яких зараз налічується декілька десятків.

Четвертий етап розвитку СКБД характеризується появою нової технології доступу до даних - Інтернет. Основна відмінність цього підходу від технології "клієнт - сервер" полягає в тому, що зникає потреба використання спеціалізованого клієнтського програмного забезпечення. Для роботи з видаленою базою даних використовується стандартний браузер Інтернету, наприклад Microsoft Internet Explorer або Netscape Navigator, і для кінцевого користувача процес звернення до даних відбувається аналогічно переміщенню по мережі Інтернет. При цьому вбудований в завантажені користувачем HTML-сторінки код, написаний зазвичай на мові Java, Java-script, Perl та інших, відстежує всі дії користувача і транслює їх у низькорівневі SQL-запити до бази даних, виконуючи таким чином ту роботу, якій в технології "клієнт - сервер" займається клієнтська програма.

Зручність даного підходу привела до того, що він почав використовуватися не тільки для видаленого доступу до баз даних, але і для користувачів локальної мережі підприємства. Прості завдання обробки даних, не пов’язані зі складними алгоритмами, що вимагають узгодженої зміни даних в багатьох взаємозв’язаних об’єктах, досить просто й ефективно можуть бути побудовані за даною архітектурою. В цьому випадку для підключення нового користувача до можливості використовувати дане завдання не потрібне встановлення додаткового клієнтського програмного забезпечення. Проте алгоритмічно складні завдання рекомендується реалізовувати в архітектурі "клієнт - сервер" з розробкою спеціального клієнтського програмного забезпечення.

У кожного з вищеперерахованих підходів до роботи з даними є свої п і свої недоліки, які й визначають сферу застосування того або іншого методу, і в даний час всі підходи широко використовуються.

1.3. Структурні елементи бази даних

Мета будь-якої інформаційної системи - обробка даних про об’єкти реального світу. Будь-яка прикладна програма є відображенням певної предметної сфери, що складається з реальних об’єктів (наприклад, автомобілі, люди, країни, земельні ділянки) та об’єктів абстрактних (наприклад, інтервал часу). Такі об’єкти називають сутностями. Предметна сфера містить його формалізований опис у вигляді даних.

Предметна сфера (ПС) - це частина реального світу, що розглядається в межах певного дослідження або певної діяльності.

Кожен об’єкт ПС характеризується сукупністю властивостей. Ці властивості відображуються за допомогою елементарних одиниць інформації - атрибутів. Наприклад, об’єкт Автомобіль може мати такі атрибути, як модель, рік виготовлення, потужність двигуна, тип коробки передач, а об’єкт Землевласник - прізвище, ім’я, рік народження, площа земельної ділянки, тип ґрунтів тощо.

Кожен атрибут має конкретне значення, наприклад, значення атрибутів об'єкта Автомобіль можуть бути такими: модель - Nissan Note, рік виготовлення - 2014, об'єм двигуна - 1,6 л, тип коробки передач - автоматична. Очевидно, що атрибути та їх значення пов'язані між собою. Крім того, сутності предметної сфери перебувають у певних відношеннях одна до одної, які називаються зв'язками.

Серед розмаїття атрибутів можна виділити істотні і малозначущі. Визнання певної властивості істотною має відносний характер. Наприклад, атрибут Кадастровий номер ділянки для співробітника ДЗК є істотним, а для пересічного громадянина - малозначущим.

Атрибут - це неподільний під час передавання та зберігання елемент інформаційного простору. З атрибутів будуються всі інші, більш складні, інформаційні конструкції. Атрибут відображає певну властивість деяких класів об'єктів.

Значення або екземпляр атрибута - це інформація про дану властивість одного конкретного об'єкта.

Клас об'єктів - це сукупність об'єктів, яка володіє однаковим набором властивостей.

Основні ідеї сучасної інформаційної технології базуються на концепції баз даних (БД).

У загальному випадку база даних - це сукупність відомостей про конкретні об'єкти реального світу в певній предметній сфері.

Бази даних - це, по суті, не що інше, як комп'ютеризована система збереження однотипних записів.

Користувачами бази даних можуть бути різні прикладні програми, програмні комплекси, а також фахівці предметної сфери, які виступають в ролі споживачів чи джерел даних, що називаються кінцевими користувачами (рис. 1.5).

Створюючи базу даних, користувач намагається упорядкувати інформацію за різними ознаками і швидко здобути вибірку з довільним сполученням ознак. Зробити це можливо тільки в тому разі, якщо дані структуровані.

Прикладом неструктурованих даних можуть слугувати дані, записані в текстовому файлі. Легко переконатися в тому, що організувати пошук даних, які зберігаються в неструктурованому вигляді, дуже складно, а упорядкувати подібну інформацію взагалі неможливо.

Для забезпечення можливості автоматизації пошуку необхідних даних потрібно їх попередньо систематизувати (структурувати), тобто розробити і виконати певні домовленості про способи подання даних.

Структура даних - сукупність правил й обмежень, які відображають зв'язки, що існують між окремими частинами даних.

Система керування базами даних (СКБД). Класифікацію структур даних подано на рис. 1.6.

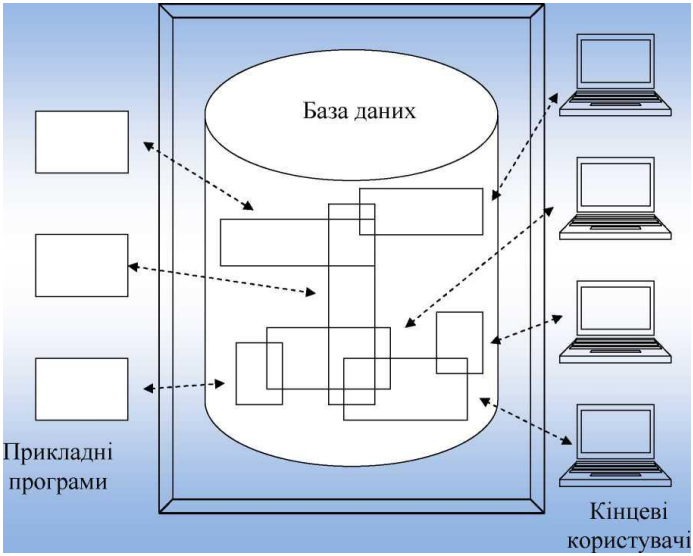


Рис. 1.5. Спрощена схема бази даних

Рис. 1.6. Класифікація структур даних

Структура даних визначається користувачем і залежить від конкретного завдання. Приклад, як записує дані людина, наведено в табл. 1.1.

Таблиця 1.1

Приклад запису даних людиною

Прізвище	Прищик
Ім'я	Мефодій
По батькові	Аристархо

	вич
Рік народження	1995
Телефон	256-14-92

У цьому запису проглядаються елементи комп'ютерного файлу даних. Тут поєднані два типи інформації. Інформація першого типу (ліва частина таблиці) визначає іншу (права частина таблиці). Якщо їх розділити, то запис можна представити наступним чином (табл. 1.2):

Таблиця 1.2

Структура	Інформація
Прізвище	Прищик
Ім'я	Мефодій
По батькові	Аристархович
Рік народження	1995
Телефон	256-14-92

Основна відмінність структури від інформації в СКБД полягає в тому, що структура залишається незмінною, а інформація змінюється при кожному введенні даних.

Таким чином, БД - це поіменована сукупність структурованих даних, що відносяться до певної предметної сфери.

СКБД (DataBase Management System - DBMS) - це комплекс програмних і мовних засобів, необхідних для створення баз даних, підтримки їх в актуальному стані і організації пошуку в них необхідної інформації.

До СКБД пред'являється низка вимог, головними з яких є забезпечення можливості:

а) введення даних. В системі повинна існувати структура, яка була б здатна накопичувати дані. Крім того, в СКБД необхідно передбачити можливість перегляду цих даних та внесення змін, для того щоб забезпечити актуальність інформації. Прикладом структури даних може слугувати звичайна анкета, створена за допомогою паперу та чорнил;

б) здійснення запитів. Запит - це формулювання користувачем своєї інформаційної потреби до певної бази даних. Система повинна надавати користувачеві можливість пошуку та перегляду окремих частин накопиченої інформації в базі даних відповідно до визначеного критерію;

в) складання звітів. Час від часу виникає потреба узагальнювати інформацію, що зберігається в базі даних. Звіт відрізняється від запиту. По- перше, звіт зазвичай охоплює не яку-небудь частину бази даних, а всю її цілком. По-друге, при отриманні звіту інформація зазвичай опрацьовується.

Звіти в СКБД не просто відображають зміст бази даних, але й певним чином її аналізують.

Щоб вести мову про інформацію, яка зберігається у базі даних, потрібно визначити деякі терміни. Поняття бази даних тісно пов'язане з такими поняттями структурних елементів, як поле, запис, файл (таблиця).

Поле - елементарна одиниця логічної організації даних, яка відповідає неподільній одиниці інформації - реквізиту.

Поля бази даних не просто визначають структуру бази. Вони ще визначають групові властивості даних, що записуються в комірки, які належать кожному з полів. Нижче наведено основні властивості полів таблиць баз даних на прикладі СКБД Microsoft Access.

Ім 'я поля - визначає, як треба звертатись до даних цього поля при автоматичних операціях з базою (за замовчуванням імена полів використовуються як заголовків стовпчик таблиць).

Тип поля - визначає тип даних, що можуть бути у даному полі.

Розмір поля - визначає граничну довжину (в символах) даних, які можуть розміщуватись у даному полі.

Формат поля - визначає спосіб форматування даних у комірках, що належать полю.

Маска введення - визначає форму, в якій вводяться дані в поле (засіб автоматизації введення даних).

Підпис - визначає заголовок стовпчика таблиці для даного поля (якщо підпис не вказаний, то як заголовок стовпчика використовується властивість Ім 'я поля).

Значення за замовчуванням - це значення, яке вводиться в комірки поля автоматично (засіб автоматизації введення даних).

Умова на значення - обмеження, що використовується для перевірки правильності введення даних (засіб автоматизації введення, який використовується, зазвичай, для даних, що мають числовий, грошовий або тип дати).

Повідомлення про помилку - текстове повідомлення, яке видається автоматично при спробі введення в поле помилкових даних.

Обов 'язкове поле - властивість, яка визначає обов'язковість заповнення даного поля при наповненні бази.

Порожні рядки - властивість, яка дозволяє вводити порожні рядкові дані (від властивості Обов'язкове поле відрізняється тим, що відноситься не до всіх типів даних, а лише до певних, наприклад до текстових).

Індексоване поле - якщо поле має цю властивість, всі операції, пов'язані з пошуком або сортуванням записів за значенням, що зберігаються вданому полі, істотно прискорюються. Крім того, для індексованих полів можна зробити так, що значення в записах будуть перевірятись за цим полем на наявність повторів, що дозволяє автоматично виключати дублювання даних.

Оскільки в різних полях можуть міститись дані різного типу, то й властивості у полів можуть розрізнятись залежно від типу даних. Так, наприклад, список вищезазначених властивостей полів відносяться головним чином до полів текстового типу. Поля інших типів можуть мати або не мати ці властивості, але можуть додавати до них і свої. Наприклад, для даних, що подають дійсні числа, важливою властивістю є кількість знаків після десяткової коми. З іншого боку, для полів, що використовуються для збереження малюнків, звукозаписів, відеокліпів, більшість із вищезазначених властивостей не мають сенсу.

Запис - це скінченна сукупність даних, яка містить певний обсяг інформації про цей об'єкт. Цей обсяг інформації визначається, по-перше, предметною галуззю, в якій розглядається об'єкт, а по-друге, тією задачею, у якій цей об'єкт розглядається. У нашому прикладі (табл. 1.3) вся записана інформація становить один запис, тобто запис є сукупністю всіх полів.

Таблиця 1.3

Прищик
Мефодій
Аристархович
1995
256-14-92

Ядром СКБД є файл (папка). Файл - це місце, де фактично зберігається інформація.

Файл - це іменована частина зовнішньої пам'яті, в яку можна записувати і з якої можна зчитувати дані. Правила іменування файлів, способів доступу до даних, що зберігаються в файлі, структура цих даних залежать від конкретної системи керування файлами і від типу файлу.

Система керування файлами бере на себе розподіл зовнішньої пам'яті, відображення імен файлів у відповідні адреси в зовнішній пам'яті і забезпечення доступу до даних.

З точки зору СКБД, файл - сукупність усіх записів, що зберігаються в базі даних.

Зв'язки - логічні взаємовідносини між записами або полями.

База даних - сукупність взаємопов'язаних даних (файлів), призначених для спільного використання.

Система керування базами даних (СКБД) - комплекс програм, які забезпечують взаємодію користувача з базою даних.

Основним принципом організації баз даних є спільне збереження даних і їх опису.

Опис даних називають метаданими. Метадані зберігаються в частині бази даних, яка називається каталогом або словником-довідником даних. Знаючи формат метаданих, можна запитувати і змінювати дані без написання додаткових програм.

Одна й та ж база даних може бути використана для розв'язку багатьох прикладних задач. Наявність метаданих і можливість інформаційної підтримки розв'язку багатьох задач - це принципова відмінність бази даних від будь-якої іншої сукупності даних, що розташовуються в зовнішній пам'яті комп'ютера.

1.4. Функції СКБД

1. Безпосереднє керування даними у зовнішній пам'яті. Ця функція включає забезпечення необхідних структур зовнішньої пам'яті як для збереження даних, що безпосередньо входять до БД, так і для службових цілей, наприклад, для прискорення доступу до даних у деяких випадках (зазвичай для цього використовуються індекси).

2. Керування буферами оперативної пам'яті. СКБД зазвичай працюють з БД значних розмірів; принаймні цей розмір зазвичай істотно більше доступного об'єму оперативної пам'яті. Зрозуміло, що якщо при звертанні до будь-якого елемента даних буде здійснюватись обмін із зовнішньою пам'яттю, то вся система буде працювати зі швидкістю пристрою зовнішньої пам'яті. Практично єдиним способом реального збільшення цієї швидкості є буферизація даних в оперативній пам'яті. Тому в розвинених СКБД підтримується власний набір буферів оперативної пам'яті з власною дисципліною заміни буферів.

3. Керування транзакціями. Транзакція - це послідовність операцій над БД, що розглядається СКБД як єдине ціле. Або транзакція успішно виконується, і СКБД фіксує зміни БД, здійснені цією транзакцією, у зовнішній пам'яті, або жодна з цих змін жодним чином не відображується на стані БД. Поняття транзакції необхідне для підтримання логічної доцільності БД. Та властивість, що кожна транзакція починається при цілісному стані БД і залишає цей стан цілісним після свого завершення, робить дуже зручним використання поняття транзакції як одиниці активності користувача по відношенню до БД. При відповідному керуванні транзакціями з боку СКБД, що виконуються паралельно, кожний з користувачів може в принципі відчувати себе єдиним користувачем СКБД. Таким чином, підтримка механізму транзакцій є обов'язковою умовою функціонування СКБД.

4. Журналізація. Однією з основних вимог до СКБД є надійність збереження даних у зовнішній пам'яті. Підтримка надійності збереження даних у БД вимагає надлишковості збереження даних, причому та частина даних, яка використовується для відновлення, повинна зберігатись особливо надійно. Найбільш поширеним методом підтримки такої надлишкової інформації є ведення журналу змін БД.

Під надійністю збереження розуміється те, що СКБД повинна бути в стані відновлювати останній узгоджений стан БД після будь-якого апаратного або програмного збою.

5. Підтримка мов БД. Для роботи з БД використовуються спеціальні мови, які в загальному випадку називаються мовами БД. У ранніх СКБД підтримувалось декілька спеціалізованих за своїми функціями мов, а саме:

• мова опису даних (МОД) (SDL - Schema Definition Language), яка називається також мовою опису схем, - для побудови структури ("шапки") таблиць БД;

• мова маніпулювання даними (ММД) (DML - Data Manipulation Language) - для заповнення БД даними і операцій оновлення (запис, видалення, модифікація);

• мова запитів - мова пошуку наборів величин у файлі відповідно до заданої сукупності критеріїв пошуку і видачі затребуваних даних без зміни змісту файлів і БД (мова перетворення критеріїв на систему команд).

Сьогодні функції всіх трьох мов виконує мова структурованих запитів SQL.

ЛЕКЦІЯ 2
ПРИНЦИПИ ПОБУДОВИ БАЗ ДАНИХ, ЇХ АРХІТЕКТУРА І КЛАСИФІКАЦІЯ

Теорія баз даних - порівняно молода галузь знань, однак сучасний світ інформаційних технологій важко уявити собі без використання баз даних, а геоінформаційних технологій - взагалі неможливо.

Практично всі ГІС пов ’язані з функціями тривалого збереження й обробки інформації, оскільки геоінформація є тим чинником, який багато в чому визначає ефективність будь-якої сфери діяльності.

На сучасному етапі розвитку суспільства збільшились інформаційні потоки і підвищились вимоги до швидкості обробки даних. Більшість операцій з обробки інформації просто неможливо виконувати вручну, вони потребують застосування більш перспективних комп ’ютерних технологій, якими і є геоінформаційні. Управлінські рішення в будь-якій предметній сфері вимагають чіткої і точної оцінки поточної ситуації та можливих перспектив її зміни. І якщо раніше при оцінці ситуації враховувалось декілька десятків факторів, що впливали на поточну ситуацію і які можна було обчислити вручну, то тепер кількість таких факторів становить сотні і навіть сотні тисяч, а ситуація змінюється не протягом року, а через декілька хвилин. При цьому обгрунтованість управлінських рішень, що повинні прийматись, повинна бути надзвичайно високою, оскільки ціна похибки, допущеної в процесі прийняття рішень, може бути надзвичайно високою. Отже, обійтись без ГІС, що грунтуються на базах просторових і атрибутивних даних та банках знань, просто неможливо.

Бази даних і бази знань є ядром автоматизованого банка даних.

2.1. Принципи побудови баз даних

База даних (БД) - іменована сукупність даних, що відображає стан об’єктів та їх відносини у певній предметній сфері.

Банк даних (БнД) - це система спеціальним чином організованих даних (баз даних), програмних, технічних, мовних, організаційно- методичних засобів, призначених для забезпечення централізованого накопичення і колективного багатоцільового використання даних.

Найбільш істотними науковими принципами, які лежать в основі побудови БД, є принцип інтеграції даних і принцип централізації керування ними (рис. 2.1).

Принцип інтеграції даних полягає в об’єднанні окремих, взаємно не зв’язаних даних у єдине цілісне утворення, в ролі якого виступає БД. У результаті всі дані подаються єдиним інформаційним масивом. При цьому полегшуються пошук взаємозалежних даних і їхня спільна обробка, зменшується надлишковість даних, спрощується процес ведення БД.

Інтеграцію даних необхідно розглядати на двох рівнях - логічному і фізичному. На логічному рівні множина структур даних відображається в єдину структуру даних, на фізичному рівні автономні файли об'єднуються в БД.

Рис. 2.1. Принципи побудови баз даних

Принцип цілісності даних відображає вимогу адекватності інформації, що зберігається в БД, стану предметної сфери. В будь-який момент часу дані повинні точно відповідати властивостям і характеристикам об'єктів. Порушення цілісності виникає внаслідок спотворення або навіть руйнації (стирання) всіх або частини даних, а також як результат запису в БД неправильної (спотвореної) інформації. Підтримка цілісності досягається:

- за рахунок контролю вхідної інформації;
- періодичної перевірки даних, що зберігаються;
- застосуванням спеціальної системи відновлення даних;
- іншими заходами.

Під незалежністю даних розуміють незалежність прикладних програм від даних, що зберігаються, при якому будь-які зміни в організації даних не вимагають корекції цих програм.

Шляхи забезпечення незалежності даних:

• уведення додаткових рівнів абстрагування даних (принцип багато-рівневості). Замість двох традиційних рівнів, передбачених базовим програмним забезпеченням і стандартними мовами програмування, - логічного і фізичного - в архітектурі БД використовується принцип трирівневої організації даних: логічний рівень поділяється на два - зовнішній (рівень користувача) і концептуальний (загальний системний рівень даних);

• передача СКБД частини функцій, пов ’язаних з організацією доступу до БнД, що раніше покладалися на прикладні програми. При цьому прикладна програма лише формує і передає ядру інформацію, необхідну для пошуку даних;

• застосуванням і дотриманням принципу відділення опису БД від процедур обробки даних;

• реляційний підхід до побудови БД.

Найбільший ефект досягається симбіозом усіх зазначених шляхів.

Відсутність надлишковості - це стан даних, коли кожний елемент присутній у БД тільки в одиничному екземплярі. Надлишковість може мати місце як на логічному рівні, коли в структурі даних повторюються одні й ті ж самі типи даних, так і на фізичному рівні, коли дані зберігаються в двох або більше екземплярах. Принцип інтеграції дозволяє звести надлишковість до мінімуму.

Під несуперечливістю розуміється змістова відповідність між даними. Це такий стан БД, при якому дані, що зберігаються в ній, не суперечать один одному. Розрізняють два аспекти несуперечливості: змістова відповідність різнотипних даних та ідентичність (рівність) дублюючих даних.

Принцип зв’язності даних полягає в тому, що дані в БД взаємозалежні, і зв’язки відбивають відношення між об’єктами предметної сфери. Множина типів даних і множина зв’язків утворюють логічну структуру даних. Наявність зв’язків між записами в БД дозволяє зменшувати надлишковість, спростити і прискорити пошук даних.

Принцип централізації керування полягає в передачі усіх функцій керування даними єдиному комплексу керуючих програм - СКБД. Як було зазначено вище, всі операції, пов’язані з доступом до БД, виконуються не прикладними програмами, а централізовано - ядром СКБД - на підставі інформації, яку отримують з цих програм. Дотримання цього принципу дозволяє автоматизувати роботу з БД і тим самим істотно підвищити ефект, який отримують від застосування інформаційної системи.

Відділення опису даних від процедур їхньої обробки припускає, що опис даних виключається з прикладних програм, складається і транслюється окремо від них і зберігається в БД (або поза нею у вигляді окремого файлу). Виведення цих описів за межі прикладної програми робить її більш незалежною від БД, полегшує процес програмування, зменшує розміри необхідної для програми пам’яті, підвищує гнучкість маніпулювання даними.

На основі зазначених вище принципів формується архітектура БД. Ведучи мову про архітектуру БД, мають на увазі архітектуру інформаційного забезпечення баз даних.

Архітектура БД - концепція взаємозв’язку логічних, фізичних і програмних компонентів системи.

2.2. Трирівнева архітектура баз даних

У процесі досліджень, присвячених тому, як саме повинна бути влаштована СКБД, пропонувались різні способи реалізації. Найбільш життєздатною виявилась реалізація, запропонована ANSI .

Перша спроба створення стандартної термінології та загальної архітектури СКБД була зроблена в 1971 р. групою DBTG (DataBase Task Group), яка прийшла до висновку про необхідність використання дворівневого підходу: схема (рівень адміністратора) і підсхеми (рівень користувацьких представлень).

У 1975 р. ANSI[1]/X3[2]/SPARC визнав необхідним використання три- рівневого підходу для задоволення потреб колективного використання структур даних при їх індивідуальному представленні. Хоча модель, що запропонована групою ANSI/SPARC, не стала стандартом, проте архітектура вважається класичною і є актуальною й донині.

Мета створення трирівневої архітектури СКБД є відділення користувацького представлення бази даних від її фізичного представлення:

• кожен користувач повинен мати можливість звертатися до даних, використовуючи власне уявлення про них (незалежно від уявлень інших користувачів);

• взаємодія користувачів БД не повинна залежати від особливостей збереження даних у ній (наприклад, індексування, хешування[3]);

• адміністратор БД повинен мати можливість змінювати структуру збереження даних у базі, не впливаючи на користувацькі уявлення;

• внутрішня структура БД не повинна залежати від змін фізичних аспектів збереження інформації (наприклад, використання нового пристрою збереження).

Архітектура ANSI-SPARC передбачає три різні рівні представлення даних: зовнішній, концептуальний та внутрішній (рис. 2.2).

Трирівневе представлення БД передбачає відповідний опис даних на кожному рівні й узгодження одних і тих самих даних на різних рівнях.

Рівень зовнішніх моделей (external level) - представлення БД з точки зору користувачів. Зовнішній рівень представлення даних не стосується фізичної організації (розміщення) даних у зовнішній пам’яті, тому його називають іноді логічним рівнем. Відповідно внутрішній рівень називають фізичним рівнем.

Оскільки БД є загальним ресурсом, то кожному користувачу може знадобитися своє, відмінне від інших уявлення про характеристики інформації, що зберігається в БД (користувач має справу з представленням предметної сфери, виражену в найбільш зручній для нього формі). Зовнішнє подання містить тільки ті сутності, атрибути та зв’язки предметної сфери БД, які цікаві користувачу (додатку).

Інші сутності, атрибути та зв’язки, які йому не потрібні, теж можуть бути представлені в БД, але користувач може навіть не підозрювати про їх існування. Тому на зовнішньому рівні може бути декілька різних зовнішніх представлень БД (кожний користувач має своє уявлення "реального світу" і не бачить зайве (з його точки зору)). Крім того, різні подання можуть різним чином відображати одні й ті ж дані. Наприклад, один користувач може переглядати дати в форматі

(день, місяць, рік), а інший - у форматі (рік, місяць, день). Деякі подання можуть включати похідні або обчислювані дані, що не зберігаються в БД як такі, а створюються в міру потреби (вік співробітників, термін експлуатації споруди тощо), що не вимагає зайвих оновлень БД і дозволяє зменшити об'єм БД. Уявлення можуть також включати комбіновані або похідні дані з декількох об'єктів.

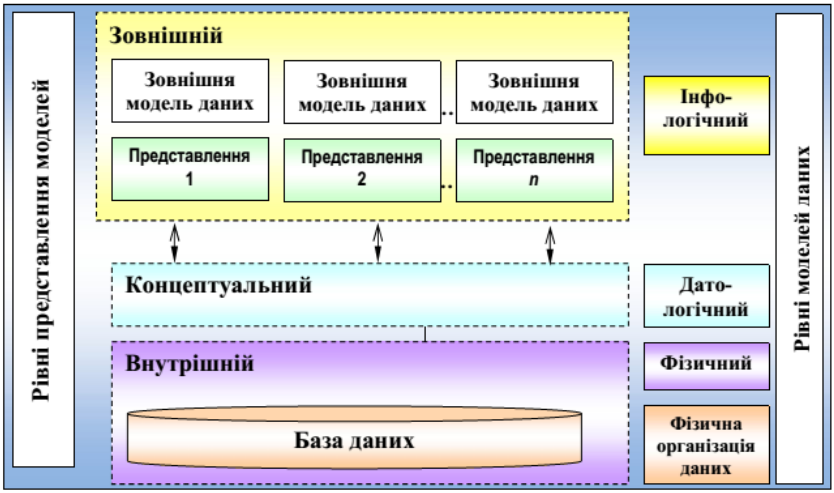


Рис. 2.2. Трирівнева модель системи керування базою даних, що запропонована ANSI називаються хеш-функціями або функціями згортки, а їх результати називають хешем, хеш- кодом або дайджестом повідомлення (англ. message digest).

Зовнішній рівень являє собою, як правило, словесний опис даних та їх взаємозв'язків і відбиває інформаційні потреби користувачів та прикладних програм (рис. 2.3).

Вимоги користувачів до зовнішнього подання охоплюють сукупність даних, які потрібні для виконання запитів користувачів.

Вимоги з боку прикладних програм до зовнішнього рівня подання даних - це перелік даних з описом їх взаємозв'язків, які необхідні для реалізації певних функціональних завдань.

Рис. 2.3. Зовнішній рівень представлення моделей

Опис зовнішнього рівня не виключає наявності дублювання, надлишковості, неузгодженості тощо.

Метою проектування на зовнішньому рівні є розробка позамашин- ного інформаційного забезпечення, яке вміщує систему вхідної (первинної) документації і характеризує певну предметну сферу (ПС), систему класифікації та кодування інформації, а також перелік відповідних вихідних повідомлень.

Під предметною сферою розуміють один або кілька об'єктів реального світу, інформація про які моделюється за допомогою БД і використовується для розв'язку різних функціональних завдань.

Існує два підходи до проектування баз даних на зовнішньому рівні:

• від предметної сфери;

• від запиту.

Підхід "від предметної сфери" полягає в тому, що формується зовнішнє інформаційне забезпечення всієї предметної сфери без урахування потреб користувачів і прикладних програм. Іноді цей підхід називають ще об 'єктним, або непроцесним.

При підході "від запиту" основним джерелом інформації про предметну сферу є вивчення запитів користувачів і потреб прикладних програм. Цей підхід також називається процесним, або функціональним. При такому підході БД проектується для виконання поточних завдань керування без урахування можливості розширення системи і виникнення нових завдань управління.

Перевагами підходу "від предметної області" є його об'єктивність, системність при відображенні ПС і стійкість інформаційної моделі, можливість реалізації великої кількості прикладних програм і запитів, у тому числі незапланованих при створенні БД. Недолік цього підходу - значний обсяг робіт, що його потрібно виконати при визначенні інформації, яка підлягає фіксації в БД, що відповідно ускладнює і збільшує термін розробки проекту.

Концептуальний рівень (conceptual level) - це узагальнене, проміжне представлення БД. Він містить логічну структуру всієї БД, а саме: всі сутності, їх атрибути і зв'язки; обмеження, семантичну інформацію про дані; інформацію про заходи безпеки і підтримки цілісності даних, що використовуються усіма додатками, які працюють з даною БД. Усі зовнішні представлення утворюються з даних цього рівня, але цей рівень не містить відомостей про методи збереження даних (тобто може мати інформацію про довжину полів, їх тип, назву, але не містить інформації про об'єми в байтах тощо).

Фактично концептуальний рівень відбиває узагальнену модель предметної сфери, для якої створювалася БД (рис. 2.4).

Рис. 2.4. Логічний рівень представлення моделей

Як і будь-яка модель, концептуальна модель відображає тільки істотні з точки зору обробки особливості об 'єктів реального світу.

Концептуальний рівень підтримує кожне зовнішнє представлення.

Даний рівень об'єднує дані, які використовуються усіма прикладними програмами, що працюють з БД. Однак концептуальний рівень не містить жодних відомостей про методи збереження даних.

Виділення концептуального рівня дозволило розробити апарат централізованого керування БД.

Внутрішній рівень (internal level) описує фізичну реалізацію бази даних. Його ще називають фізичним рівнем. Це власне дані, що зберігаються в файлах або в сторінкових структурах, що розташовані на зовнішніх носіях інформації. На внутрішньому рівні міститься така інформація: розподіл дискового простору для збереження даних та індексів; відомості про розміщення записів; відомості про стиснення даних та методи їх шифрування.

Виділяють декілька проблем фізичного подання даних, які необхідно вирішити, щоб забезпечити максимальну ефективність роботи бази даних.

По-перше, необхідно вирішити, як здійснювати пошук потрібного запису. Для цього потрібно встановити відповідність між логічним записом і адресою фізичного запису.

По-друге, необхідно вирішити, як організувати дані, щоб їх пошук був ефективним.

По-третє, як додавати нові записи до даних, видаляти старі записи і при цьому не порушувати систему адресації і пошуку.

Відмінності між рівнями представлення даних показані на рис. 2.5.

Рис. 2.5. Відмінності між рівнями представлення даних

2.3. Забезпечення незалежності СКБД від даних

Незалежність від даних - основоположний принцип побудови СКБД. Відповідно до цього принципу в системі повинні підтримуватися роздільні представлення даних для користувача ("логічна незалежність") і для системних механізмів середовища зберігання БД ("фізична незалежність"). Такий поділ позбавляє користувача від необхідності знання прийнятого способу зберігання БД і дозволяє динамічно оптимізувати спосіб зберігання БД у процесі експлуатації системи для забезпечення більш високої продуктивності системи і (або) більш раціонального використання ресурсів пам'яті.

Логічна незалежність від даних означає цілковиту захищеність зовнішніх схем від змін, що вносяться в концептуальну схему, тобто припускає можливість зміни однієї прикладної програми без коригування інших додатків, що працюють з цією ж БД, додавання і видалення нових сутностей, атрибутів і зв'язків.

Фізична незалежність від даних означає захищеність концептуальної схеми від змін, що вносяться у внутрішню схему, тобто припускає можливість перенесення інформації, що зберігається з одних носіїв на інші, при збереженні працездатності всіх додатків, що працюють з даною базою даних, модифікацію індексів тощо. Користувачі можуть помітити зміни, що відбулись у внутрішній схемі, тільки за зміною продуктивності. Це саме те, чого не вистачало при використанні файлових систем.

2.4. Відображення рівнів моделей

Архітектура СКБД, крім безпосереднього представлення трьох рівнів моделей, включає певні відображення:

- відображення концептуального рівня на внутрішній;
- декілька відображень зовнішніх рівнів на концептуальний.

Відображення "концептуальний - внутрішній" встановлює відповідність між концептуальним представленням і БД, що зберігається, тобто описує, як концептуальні записи і поля представлені на внутрішньому рівні.

Відображення "зовнішній - концептуальний' визначає відповідність між певним зовнішнім представленням і концептуальнім представленням.

Відображення "концептуальний - внутрішній' слугує основою фізичної незалежності від даних, а відображення "зовнішній - концептуальний' є ключем до логічної незалежності від даних.

Опис БД - схема БД, яка створюється в процесі її проектування, причому передбачається, що вона змінюється доволі рідко.

Стан БД - сукупність інформації, що зберігається в БД у певний момент часу.

Схема БД іноді називається змістом БД, а її стан - деталізацією.

2.5. Організація процесу проходження користувацького запиту

На рис. 2.6 представлено взаємодію користувача, СКБД і операційної системи (ОС) при обробці запиту на отримання даних. Цифрами помічено послідовність взаємодій.

1. Користувач посилає СКБД запит на отримання даних з БД.

2. Аналіз прав користувача і зовнішньої моделі даних, що відповідає даному користувачу, підтверджує або забороняє доступ даного користувача до запитуваних даних.

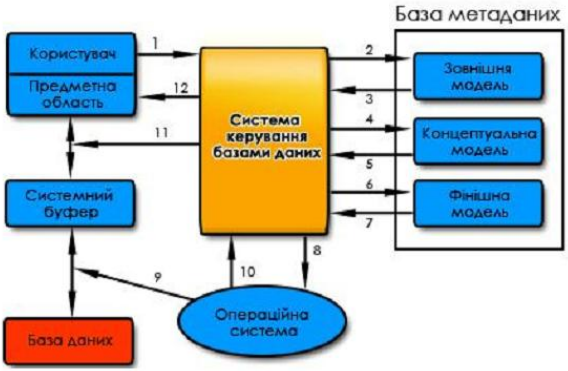


Рис. 2.6. Схема проходження запиту до БД

3. У разі заборони на доступ до даних СКБД повідомляє користувачу про це (стрілка 12) і завершує подальший процес обробки даних, в іншому випадку СКБД визначає частину концептуальної моделі, яка виділяється запитом користувача.

4. СКБД отримує інформацію про запитувану частину концептуальної моделі.

5. СКБД запитує інформацію про місце розташування даних на фізичному рівні (файли або фізичні адреси).

6. В СКБД повертається інформація про місце розташування даних у термінах операційної системи.

7. СКБД ввічливо просить ОС надати необхідні дані, використовуючи засоби ОС.

8. ОС здійснює перекачування інформації з пристроїв збереження і пересилає її в системний буфер.

9. ОС сповіщає СКБД про закінчення пересилки.

10. СКБД обирає з доставленої інформації, що знаходиться в системному буфері, тільки те, що потрібно користувачеві, і пересилає ці дані в робочу сферу користувача.

База метаданих (БМД) - місце збереження всієї інформації про використовувані структури даних, логічну організацію даних, права доступу користувачів і фізичне розташування даних.

Для керування БМД існує спеціальне програмне забезпечення адміністрування баз даних, яке призначено для коректного використання єдиного інформаційного простору багатьма користувачами.

СКБД має доволі розвинений інтелект, який дозволяє їй не повторювати безглуздих дій. І тому, наприклад, якщо цей же користувач повторно звернеться до СКБД з новим запитом, то для нього вже не будуть перевірятися зовнішня модель і права доступу, а якщо подальший аналіз запиту покаже, що дані можуть знаходитися в системному буфері, то СКБД здійснить тільки 11 і 12 кроки в обробці запиту.

Зрозуміло, що механізм проходження запиту в реальних СКБД набагато складніший, але й ця спрощена схема показує, наскільки серйозними і складними повинні бути механізми обробки запитів, що підтримуються реальними СКБД.

2.6. Користувачі СКБД

Як і будь-який програмно-організаційно-технічний комплекс, СКБД існує в часі й в просторі. Вона має певні стадії свого розвитку:

1) проектування;
2) реалізація;
3) експлуатація;
4) модернізація і розвиток;
5) повна реорганізація.

На кожному етапі свого існування із СКБД пов’язані різні категорії користувачів (рис. 2.7).

Групу внутрішніх користувачів складають: адміністратор даних, адміністратор БД (АБД), адміністратори додатків (функціональних підсистем), системні і прикладні програмісти.

Примітка. Функції адміністратора БД на стадії розробки і на стадії експлуатації різні і тому виконуються різними особами.

Адміністратори даних - це група користувачів, яка на початковій стадії розробки БД відповідає за його оптимальну організацію з точки зору одночасної роботи кінцевих користувачів.

Адміністратор даних:

- визначає потреби зовнішніх користувачів при проектуванні БД;
- ставить завдання адміністратору БД.

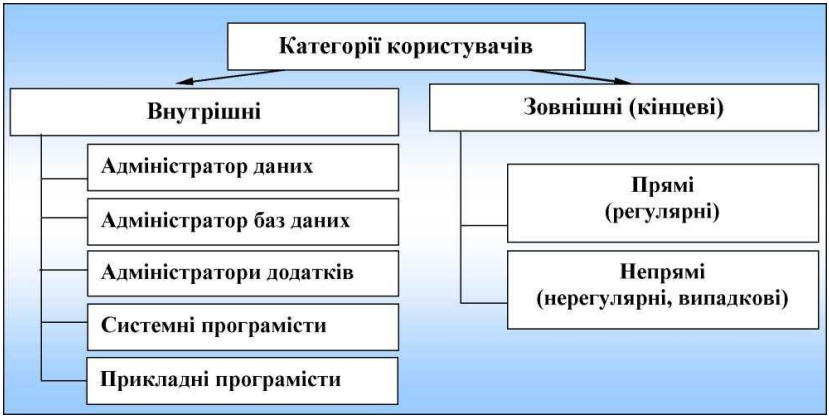


Рис. 2.7. Користувачі систем керування базами даних

В ІС, що створюються на основі СКБД, способи організації даних і методи доступу до них перестали відігравати істотну роль, оскільки стали схованими всередині СКБД. Масовий (кінцевий користувач), зазвичай, має справу тільки із зовнішнім інтерфейсом, що підтримується СКБД. Ці переваги не можуть бути реалізовані шляхом механічного об'єднання даних у БД. У системі обов'язково повинна існувати спеціальна посадова особа (група осіб) - адміністратор бази даних (АБД), який несе відповідальність за проектування і загальне керування БД.

АБД визначає інформаційний зміст БД. З цією метою він ідентифікує об'єкти БД і моделює базу, використовуючи мову опису даних. Отримувана модель слугує в подальшому довідковим документом для адміністраторів додатків і користувачів. АБД вирішує також усі питання, пов'язані з розміщенням БД в пам'яті, вибором стратегії і обмежень доступу до даних. До функції АБД входять також організація завантаження, ведення і відновлення БД та багато інших дій, які не можуть бути повністю формалізовані й автоматизовані.

АБД відповідає за коректність роботи даної БД у багатокористува- цькому режимі. На стадії розвитку та реорганізації він відповідає за можливість коректної реорганізації БД без зміни або припинення її поточної експлуатації.

На групу АБД покладаються найбільш складні обов'язки. В складі групи адміністратора БД повинні бути:

- системні аналітики;
- проектувальники структур даних і зовнішнього по відношенню до БД інформаційного забезпечення;
- проектувальники технологічних процесів обробки даних;
- системні програмісти;
- прикладні програмісти;
- фахівці з технічного обслуговування;
- - оператори.

Системні програмісти - забезпечують функціонування СКБД у середовищі операційної системи, розробляють компоненти, які розширюють програмне забезпечення СКБД.

Прикладні програмісти - розробляють прикладні програми СКБД, які забезпечують функціонування СКБД.

Адміністратор БД:

- на стадії проектування і розробки є, зазвичай, керівником проекту створення БД. Він керує роботами з розробки БД і програмного забезпечення БД;
- на стадії експлуатації відповідає за функціонування системи (захист даних від руйнування і від несанкціонованого доступу, забезпечення достовірності даних, аналіз ефективності використання ресурсів).

Основні функції групи адміністратора БД

1. Аналіз предметної сфери: опис предметної сфери, виявлення обмежень цілісності, визначення статусу (доступності, секретності) інформації, визначення потреб користувачів, визначення відповідності "дані - користувач", визначення об'ємно-часових характеристик обробки даних.

2. Проектування структури БД: визначення складу і структури файлів БД і зв'язків між ними, вибір методів упорядкування даних і методів доступу до інформації, опис БД на мові опису даних.

3. Завдання обмежень цілісності при опису структури БД і процедур обробки БД:

- завдання декларативних обмежень цілісності, притаманних предметній сфері;
- визначення динамічних обмежень цілісності, притаманних предметній сфері в процесі зміни інформації, що зберігається в БД;
- визначення обмежень цілісності, що викликана структурою БД;
- розробка процедур забезпечення цілісності БД при введенні і корегуванні даних;
- визначення обмежень цілісності при паралельній роботі користувачів у багатокористувацькому режимі.

4. Первісне завантаження і ведення БД:

- розробка технології первісного завантаження БД, яка буде різнитись від процедури модифікації і доповнення даними при штатному використанні БД;
- розробка технології перевірки відповідності введених даних реальному стану предметної сфери. БД моделює реальні об'єкти певної предметної сфери та взаємозв'язки між ними, і на момент початку штатної експлуатації ця модель повинна повністю відповідати стану об'єктів предметної сфери на даний момент часу;
- відповідно до розробленої технології первісного завантаження може знадобитись проектування системи первинного введення даних.

5. Захист даних:

- визначення системи паролів, принципів реєстрації користувачів, створення груп користувачів, що мають однакові права доступу до даних;
- розробка принципів захисту конкретних даних і об'єктів проектування; розробка спеціалізованих методів кодування інформації при її циркуляції в локальній і глобальній інформаційних мережах;
- розробка засобів фіксації доступу до даних і спроб порушення системи захисту;
- тестування системи захисту;
- дослідження випадків порушення системи захисту і розвиток динамічних методів захисту інформації в БД.

6. Забезпечення відновлення БД:
- розробка організаційних засобів архівування і принципів відновлення БД;
- розробка додаткових програмних засобів і технологічних процесів відновлення БД після збоїв.

7. Аналіз звернень користувачів БД - збір статистики за характером запитів, часом їх виконання, вихідними документами, що вимагаються.

8. Аналіз ефективності функціонування БД:
- аналіз показників функціонування БД;
- планування реструктуризації (змін структури) БД і реорганізації банків даних (БнД).

9. Робота з кінцевими користувачами:
- збір інформації про зміну предметної сфери;
- збір інформації про оцінку роботи БД;
- навчання користувачів, їх консультування;
- розробка необхідної методичної і навчальної документації для роботи кінцевих користувачів.

10. Підготовка і підтримка системних засобів:
- аналіз існуючих на ринку програмних засобів і аналіз можливості та необхідності їх використання в рамках БД;
- розробка необхідних організаційних і програмно-технічних заходів з розвитку БД;
- перевірка працездатності закуповуваних програмних засобів перед підключенням їх до БД;
- опікуватись підключенням нових програмних засобів до БД.

11. Організаційно-методична робота з проектування БД:
- вибір або створення методики проектування БД;
- визначення цілей і напрямку розвитку системи в цілому;
- планування етапів розвитку БД;
- розробка загальних словників-довідників проекту БД і концептуальної моделі;
- стикування зовнішніх моделей розроблюваних додатків;
- опікуватись підключенням нового додатку до діючої БД;

- забезпечення можливості комплексного налагоджування множини додатків, що взаємодіють з однією БД.

Адміністратори додатків (АБД) - це група користувачів, яка функціонує під час проектування, створення та реорганізації БД. Проте не для кожної БД можна виділити наведені типи користувачів. Наприклад, при розробці ГІС з використанням настільних СКБД адміністратор БД, адміністратор додатків і розробник найчастіше існують в одній особі. Однак при побудові сучасних складних корпоративних баз геоданих, які використовуються для автоматизації всіх або більшої частини процесів управління територіями, можуть існувати і групи адміністраторів додатків, і відділи розробників.

Адміністратори додатків координують роботу розробників при розробці конкретної програми або комплексу програм, об'єднаних у функціональну підсистему. Розробники конкретних програм працюють з тією частиною інформації з БД, яка потрібна для конкретного додатка.

Адміністратор додатків визначає для додатків підмоделі даних. Тим самим різні додатки забезпечуються власним "поглядом", але не на всю БД, а тільки на потрібну для конкретного додатка ("видиму") її частину. Вся інша частина БД для даного додатка буде "прозорою". Прикладні програмісти мають, зазвичай, у своєму розпорядженні один або декілька мов програмування, за допомогою яких генеруються прикладні програми.

Кінцеві користувачі - це основна категорія користувачів, в інтересах яких і створюється база даних.

Залежно від особливостей створюваного БД коло його кінцевих користувачів може істотно різнитись. Це можуть бути непрямі (випадкові) користувачі, які звертаються до БД час від часу за отриманням певної інформації, а можуть бути й прямі (регулярні) користувачі.

Прямі кінцеві користувачі працюють з СКБД в інтерактивному режимі. Фактично вони забезпечують актуальність бази даних, тобто вносять оперативну інформацію, дають команди на обробку і видачу даних.

Непрямі кінцеві користувачі не вступають у безпосередній контакт з програмно-технічними компонентами СКБД. Вони формулюють свої запити службі адміністратора БД і прямим кінцевим користувачам.

Випадковими користувачами можуть виступати, наприклад, можливі клієнти ГІС (школярі, студенти, пересічні громадяни тощо), що ознайомлюються з її можливостями послуг з узагальненим або докладним описом того чи іншого.

Регулярними користувачами можуть бути фахівці органів влади, управління навколишнього середовища, що працюють зі спеціально розробленими для них програмами, які забезпечують автоматизацію їх діяльності при виконанні своїх посадових обов'язків. Наприклад, фахівець, який планує роботу з охорони земель, має в своєму розпорядженні програму, яка допомагає йому планувати і розподіляти поточні перевірки земельного законодавства, контролювати хід їх виконання, замовляти необхідну інформацію з різних підрозділів. Головний

принцип полягає в тому, що від кінцевих користувачів не повинно вимагатися спеціальних знань у сфері ГІС, обчислювальної техніки і мовних засобів.

2.7. Класифікація СКБД і моделей баз даних

Модель даних - сукупність структур даних, обмежень цілісності та операцій їх обробки.

Модель даних є ядром будь-якої бази даних. За допомогою моделі даних можуть бути подані об'єкти предметної сфери та взаємозв'язки між ними.

Моделі даних можна класифікувати за різними ознаками:

• за структурою організації даних:

- ієрархічні;
- мережеві;
- реляційні;
- багатовимірні;
- об'єктно орієнтовані;

• за місцем збереження даних:

- внутрішня модель (фізична база даних) являє собою найнижчий рівень БД. Вона складається з різних екземплярів типів даних, які зберігаються на пристроях зовнішньої пам'яті;
- зовнішня модель. Зазвичай, користувача цікавить лише певна частина БД. Окрім того, користувач не знає, яким чином фізично зберігаються ці дані. Зовнішня модель - це інформаційний зміст бази даних у вигляді, як її уявляє собі користувач. Для звертання до бази даних можуть використовуватися як мови програмування, так і спеціалізовані мови, наприклад мова структурованих запитів SQL;

• за характером інформації, що зберігається в БД:

- фактографічні;
- документальні.

Якщо провести аналогію з прикладами інформаційних сховищ на паперових носіях, то фактографічні БД - це картотеки, а документальні - це архіви. У фактографічних БД зберігається лаконічна інформація в строго визначених форматах. В документальних БД - будь-які документи, причому не обов'язково текстові, але й графіка, відео, звук (мультимедіа).

• за технологією обробки даних (рис. 2.8):

Рис. 2.8. Класифікація баз даних за технологією обробки

Централізована база даних зберігається в пам'яті однієї обчислювальної системи. Якщо ця система є мейнфреймом[4], то доступ здійснюється за допомогою терміналів або файловим сервером локальної мережі.

Розподілена база даних складається з декількох, можливо, таких, що перетинаються або навіть дублюючих одна одну, частин, які зберігаються в різних ЕОМ обчислювальної мережі. Робота з такою базою здійснюється за допомогою системи керування розподіленою базою даних (СКРБД);

• за способом доступу до даних (рис. 2.9):

Рис. 2.9. Класифікація баз даних за способом доступу до даних

Системи централізованих баз даних з віддаленим (мережевим) доступом припускають різні архітектури подібних систем, а саме - "файл - сервер" і "клієнт - сервер"

"Файл - сервер". Архітектура БД з мережевим доступом припускає виділення одного з комп'ютерів мережі як центрального (мережевого). Концепція архітектури БД "файл - сервер" зображена на рис. 2.10.

Рис. 2.10. Концепція архітектури БД "файл - сервер "

На такій машині зберігається спільно використовувана централізована БД. Усі інші комп'ютери мережі виконують функції робочих станцій, за допомогою яких підтримується доступ користувацької системи до централізованої БД. Файли БД відповідно до користувацьких запитів передаються на робочі станції, де в основному і відбувається обробка. При великій інтенсивності доступу до одних і тих же даних продуктивність ІС знижується. Користувачі можуть також створювати на робочих станціях локальні БД, які використовуються ними монопольно.

Схема обробки інформації в БД за принципом "файл - сервер" представлена на рис. 2.11.

До переваг такої архітектури передусім слід віднести відсутність високих вимог до продуктивності сервера (головна вимога - забезпечення необхідного об'єму дискового простору та відсутність необхідності розміщення й інсталяції СКБД на сервері).

Серед недоліків такої архітектури потрібно відзначити високий мережевий трафік[5] та відсутність спеціальних механізмів захисту файлів БД з боку СКБД.

На даний час файл-серверні СКБД (Microsoft Access, Borland Paradox) вважаються застарілими.

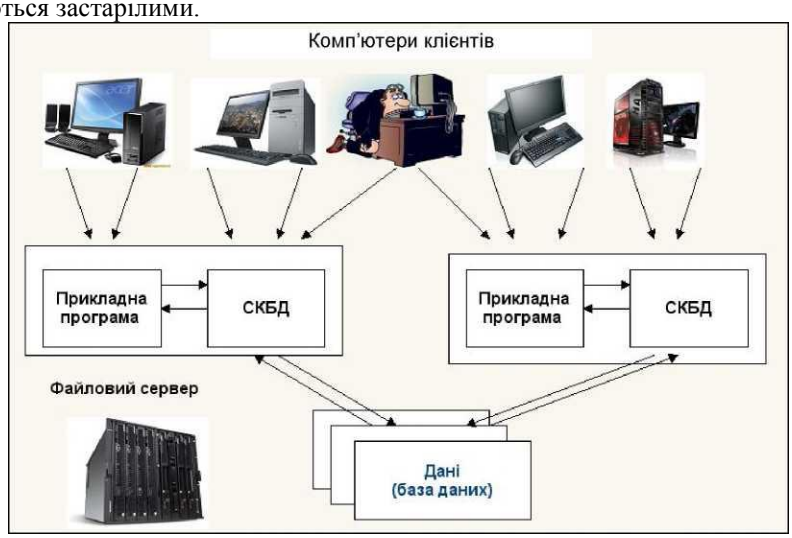


Рис. 2.11. Схема обробки інформації в БД за архітектурою "файл - сервер"

"Клієнт - сервер" За цією концепцією передбачається, що, крім збереження централізованої бази даних, центральний комп'ютер (сервер бази даних) повинен забезпечувати виконання основного об'єму обробки даних.

Запит на дані, які видаються клієнтом (робочою станцією), породжує пошук і добування даних на сервері. Добуті дані (однак не файли) транспортуються по мережі від сервера до клієнта. Специфікою архітектури "клієнт - сервер" є використання мови структурованих запитів SQL. Концепція "клієнт - сервер" умовно зображена на рис. 2.12.

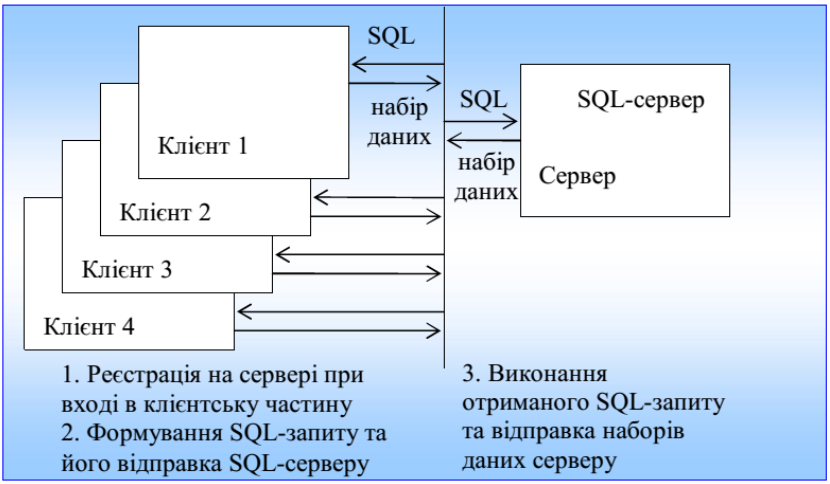


Рис. 2.12. Схема обробки інформації в БД за принципом "клієнт - сервер"

Схема обробки інформації в БД за принципом "клієнт - сервер" представлена на рис. 2.13.

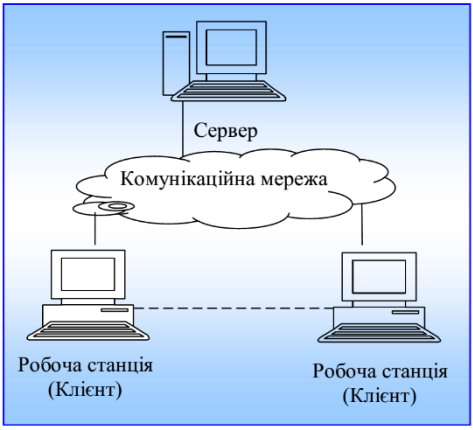


Рис. 2.13. Схема обробки інформації в БД за архітектурою "клієнт - сервер "

До переваг такої архітектури передусім треба віднести більш м'який трафік мережі та забезпечення за допомогою SQL-сервера функцій цілісності і забезпечення даних.

Крім того, робить можливим у більшості випадків розподіл функцій обчислювальної системи між декількома незалежними комп'ютерами в мережі.

Всі дані зберігаються на сервері, який, зазвичай, захищений набагато краще за більшість клієнтів і дозволяє використовувати ресурси одного сервера клієнтам з різними апаратними платформами, операційними системами тощо.

До недоліків треба віднести високу вартість устаткування. Непрацездатність сервера може зробити непрацездатною всю обчислювальну мережу. Підтримка роботи даної системи вимагає окремого фахівця - системного адміністратора.

Прикладами таких СКБД можуть слугувати Firebird, Interbase, IBM DB2, MS SQL Server, Sybase, Oracle, PostgreSQL, MySQL, ЛИНТЕР.

Клієнт-серверна архітектура розділяється на декілька типів:

• дворівнева архітектура "клієнт - сервер";

• трирівнева архітектура "клієнт - сервер";

• багаторівнева архітектура або n-рівнева архітектура.

Дворівнева архітектура "клієнт - сервер". У випадку з дворівневою архітектурою "клієнт - сервер" база даних розміщується на мережевому сервері, проте програма клієнта позбавлена можливості прямого доступу до БД. Доступ до БД регулюється спеціальною програмою - сервером БД (рис. 2.14).

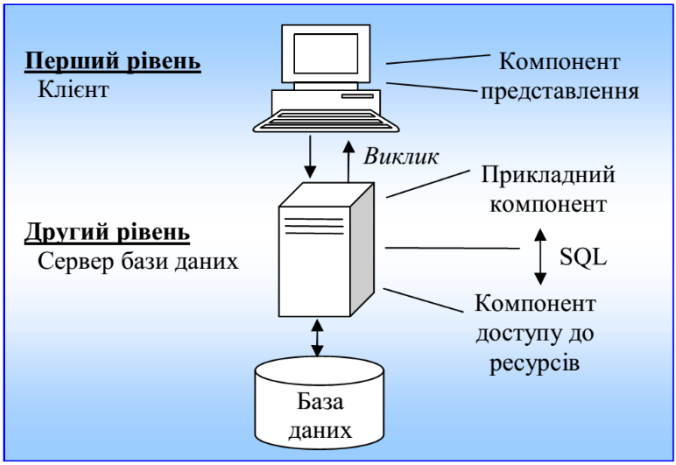


Рис. 2.14. Дворівнева архітектура ".клієнт - сервер"

Взаємодію сервера БД і клієнта реалізується за допомогою SQL-запитів, які формує і посилає серверу клієнт. Сервер, прийнявши запит, виконує його і повертає результат клієнту.

В клієнтському застосуванні в основному здійснюються інтерпретація отриманих від сервера даних, реалізація призначеного для користувача інтерфейсу, а також реалізація частини бізнес-правил.

Проте дворівнева архітектура має низку недоліків:

• погіршення продуктивності прямо пропорційна кількості користувачів;

• незалежно від того, який тип клієнта використовується, велика частина обробки даних повинна знаходитися в БД, це означає, що вона повністю залежить від можливостей, передбачених у БД виробником;

• дворівнева архітектура настільки залежить від конкретної реалізації бази даних, що перенесення існуючих застосувань для різних СКБД стає серйозною проблемою.

Трирівнева архітектура "клієнт - сервер". У цій моделі процес, що виконується клієнтом, відповідає за інтерфейс з користувачем, звертаючись за виконанням послуг до прикладного компонента. Прикладний компонент реалізований як група процесів, що виконують прикладні функції, і називається сервером додатка. Всі операції над інформаційними ресурсами баз даних виконуються компонентами доступу до ресурсів. Таким чином, на відміну від попередньої архітектури, обмін між клієнтом і сервером здійснюється за допомогою спеціально розробленого набору команд (API), а не SQL-запитів. Набір цих команд визначається розробником, що дозволяє йому обмежити набір допустимих дій з даними, доступних клієнтові. Трирівнева архітектура "клієнт - сервер" представлена на рис. 2.15.

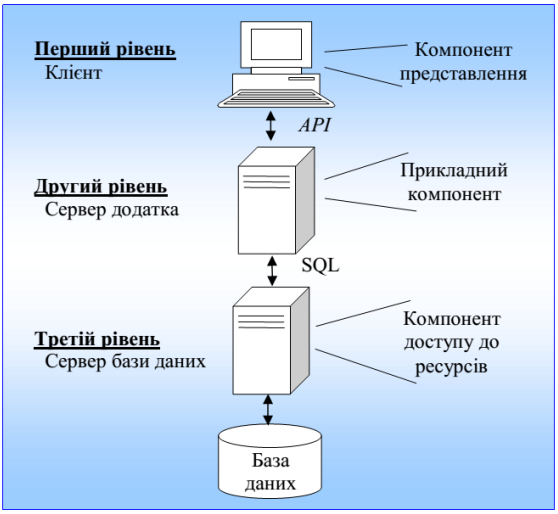


Рис. 2.15. Трирівнева архітектура ”клієнт - сервер”

Уведення додаткового рівня дозволяє здолати деякі обмеження дворівневої архітектури, описаних у загальних рисах вище. Як і з дворівневою моделлю, рівні можуть розташовуватися або на різних комп’ютерах, або на одному комп’ютері в тестовому режимі.

Багаторівнева архітектура "клієнт - сервер" - різновид архітектури "клієнт - сервер", у якій функція обробки даних винесена на один або декілька окремих серверів. Це дозволяє розділити функції зберігання, обробки і представлення даних для ефективнішого використання можливостей серверів і клієнтів.

Трирівнева архітектура є окремим випадком багаторівневої, але якщо розглядати їх окремо, то можна виділити декілька переваг багаторівневої архітектури перед трирівневою, такі як:

• масштабованість. Навантаження поділяється на декілька web- серверів, які взаємодіють з БД або з сервером наступного рівня.

• захищеність. Використання декількох фізичних рівнів підвищує захищеність інформаційної системи від мережевих атак, а також функцію

захисту можуть виконувати всі рівні сервера, розподіляючи тим самим навантаження;

• стабільність. При втраті працездатності одного з проміжних рівнів його функцію можуть поділити між собою рівні ідентичної функціональності, що залишились. Хоча продуктивність у такій ситуації й зменшиться, проте система залишиться працездатною, на відміну від трирівневої системи.

Багаторівнева архітектура зазвичай складається з чотирьох рівнів (рис. 2.16), де сервер мережі відповідає за обробку з'єднання між клієнтом, браузером і сервером системи.

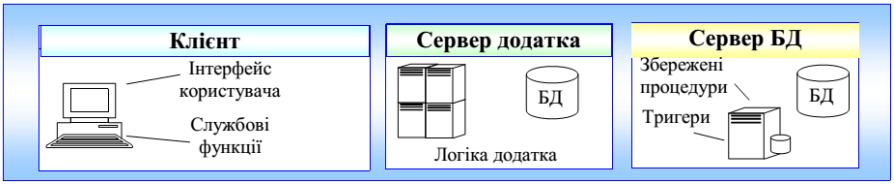


Рис. 2.16. Багаторівнева архітектура "клієнт - сервер"

До персональних СКБД відносяться VISUAL FOXPRO, ACCESS та ін. До багатокористувацьких СКБД відносяться, наприклад, СКБД ORACLE та INFORMIX.

Багатокористувацькі СКБД включають в себе сервер БД і клієнтську частину, працюють в неоднорідному обчислювальному середовищі, припускають різні типи ЕОМ і різні операційні системи. Тому на базі СКБД можна створити інформаційну систему, що функціонує за технологією "клієнт - сервер".

Універсальність багатокористувацьких СКБД проявляється відповідно у високій ціні і комп'ютерних ресурсах, необхідних для підтримки СКБД, що становлять собою сукупність мовних і програмних засобів, призначених для створення, ведення і використання БД.

Персональні СКБД забезпечують можливість створення персональних БД і дешевих додатків, що працюють з ними, і, за потреби, створення додатків, що працюють із сервером БД;

• за типом використовуваних даних:

- слабко типізовані;
- сильно типізовані.

У сильно типізованих моделях усі дані мають належати до певної категорії або типу. Якщо дані не підпадають під жодну з категорій, їх потрібно типізувати штучно. Деякі моделі будуються у такий спосіб, що категорії визначаються наперед і не можуть змінюватись динамічно. У цьому разі модельований світ начебто вміщується в гальмівну сорочку. Наприклад категорія "службовець" - строго фіксована, й усі об'єкти повинні мати однакові властивості та структуру.

Сильно типізовані моделі мають значні переваги, бо дають змогу побудувати абстракції властивостей даних і дослідити їх у термінах категорій.

Більшість моделей, що використовуються в автоматизованих системах, зокрема в БД, належать до сильно типізованих.

Для слабко типізованих належність даних до тієї або іншої категорії не має жодного значення. Категорії використовуються настільки, наскільки це доцільно в кожному конкретному випадку. Окремі дані можуть існувати як незалежно, так і у зв'язку з іншими. Інформація про категорії, якщо вони використовуються, розглядається як додаткова.

На відміну від сильно типізованих моделей, слабко типізовані забезпечують інтеграцію даних і категорій. Найкращі можливості такої інтеграції надаються численням предикатів, яке у багатьох моделях використовується для зображення знань, що підтримуються базовими засобами моделювання.

[1] Американський національний інститут стандартів (англ. American National Standards Institute, ANSI) - об'єднання американських промислових і ділових груп, що розробляє торгові і комунікаційні стандарти, член ISO.

[2] X3 - комітет обчислювальної техніки й обробки інформації ANSI, SPARC - Standards Planning and Requirements Committee (Комітет планування стандартів і норм).

[3] Хешування (іноді гешування, англ. hashing) - перетворення вхідного масиву даних довільної довжини на вихідний бітовий рядок фіксованої довжини

[4] Мейнфрейм (від англ. mainframe) - великий універсальний високопродуктивний відмовостійкий сервер зі значними ресурсами введення-виведення, великим об'ємом оперативної і зовнішньої пам'яті, призначеної для використання в критично важливих обчислювальних системах.

[5]Мережевий трафік, або інтернет-трафік (англ. traffic - "рух", "вантажообіг") - об'єм інформації, що передається через комп'ютерну мережу за певний період часу. Кількість трафіка вимірюється як в пакетах, так і в бітах, байтах та їх похідних: кілобайт (Кб), мегабайт (Мб) тощо.

ЛЕКЦІЯ 3
МОДЕЛІ БАЗ ДАНИХ

3.1. Класифікація моделей баз даних за рівнями подання

Модель даних - це інтегрований набір понять для опису даних, зв'язків між ними і обмежень, що накладаються на дані.

Реалізація (implementation) заданої моделі даних - це фізичне втілення на реальній машині компонентів абстрактної машини, яке в сукупності складають цю модель.

Загальний опис БД прийнято називати схемою бази даних. Для кожного рівня архітектури ANSI-SPARC існують свої схеми (рис. 3.1).

Предметна сфера

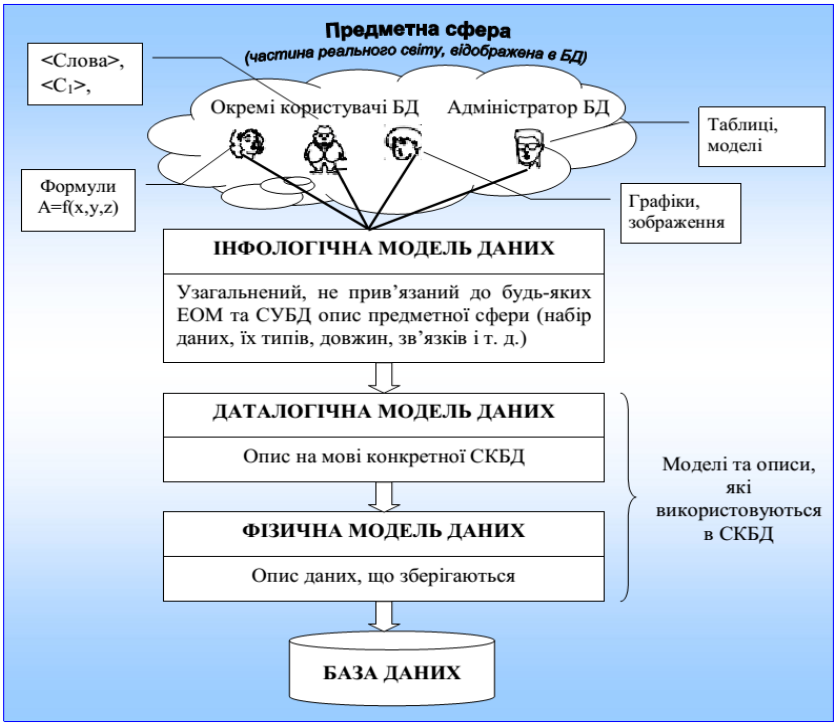


Рис. 3.1. Рівні схем баз даних

Модель даних розглядається як сполучення трьох компонентів:

• структурна частина - набір правил, за якими може бути побудована база даних;

• керуюча частина, яка визначає типи припустимих операцій з даними;

• набір обмежень підтримки цілісності даних (необов'язковий компонент), який гарантує коректність використовуваних даних.

Схема класифікації моделей даних за рівнями подання наведена на рис. 3.2.

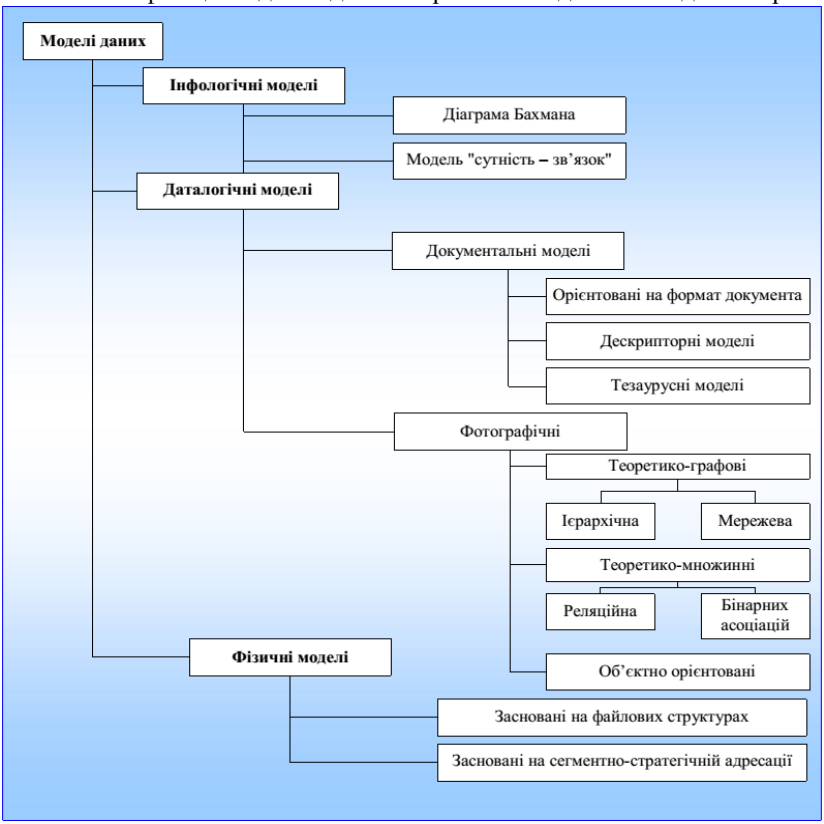


Рис. 3.2. Класифікація моделей даних за рівнями подання

3.2. Інфологічні моделі

Інфологічна модель (інформаційно-логічна модель, ІЛМ) - людиноорієнтована, незалежна від типу СКБД модель ПС, яка визначає сукупності інформаційних об'єктів, їх атрибутів і відношень між ними, динаміку змін ПО, а також характер інформаційних потреб користувачів.

Класифікація інфологічних моделей представлена на рис. 11.3.

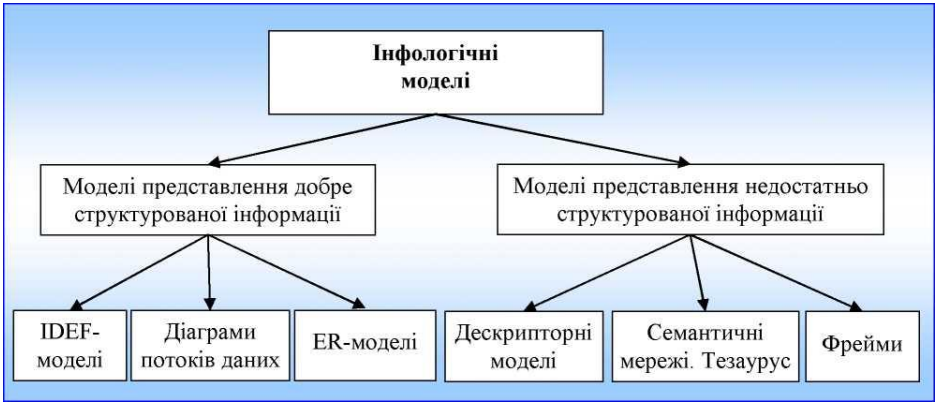


Рис. 3.3. Класифікація інфологічних моделей

ІЛМ даних використовуються на ранніх стадіях проектування для опису структур даних у процесі розробки баз даних або додатків.

Мета інфологічного проектування - створити структуровану інформаційну модель ПС, для якої розроблятиметься БД. Під час проектування на інфологічному рівні створюється ІЛМ, яка повинна:

• адекватно відображати модельовану ПС;

• однозначно трактувати модель;

• чітко визначати ПС, що моделюється (кінцевість моделі);

• легко розширюватись (забезпечувати введення і видалення нових даних без зміни попередньо визначених);

• забезпечувати композицію і декомпозицію моделі в зв'язку з великою розмірністю реальних інфологічних моделей;

• легко сприйматись різними категоріями користувачів (бажано, щоб ІЛМ модель будував або брав участь побудові фахівець, що працює в даній ПС, а не тільки проектувальник систем машинної обробки даних);

• забезпечувати можливість застосування мови специфікацій моделі як при ручному, так і при автоматизованому проектуванні СКБД.

ІЛМ ПС може бути описана різним чином (рис. 3.4).

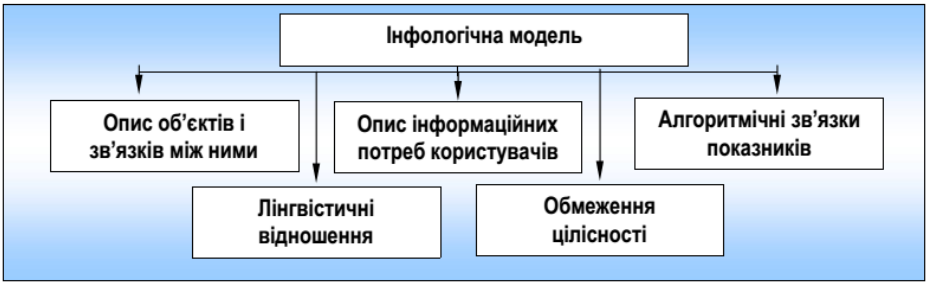


Рис. 3.4. Складові інфологічної моделі

Існує два підходи до інфологічного проектування: аналіз об’єктів і синтез атрибутів (семантичні моделі). Підхід, що базується на аналізі об’єктів, називається низхідним, а на синтезі атрибутів - висхідним.

Семантичні моделі головну увагу приділяють структурі даних. Найбільш поширеною семантичною моделлю є модель ”сутність - зв’язок” (Entity Relationship model, ER-модель).

ER-модель складається із сутностей, зв ’язків, атрибутів, доменів атрибутів, ключів. Моделювання даних відображає логічну структуру даних, так само, як блок-схеми алгоритмів відображають логічну структуру програми.

Об’єктні моделі головну увагу приділяють поведінці об’єктів даних і засобам маніпуляції даними. Головне поняття таких моделей - об’єкт, тобто сутність, яка має стан і поведінку. Стан об’єкта визначається сукупністю його атрибутів, а поведінка об’єкта визначається сукупністю операцій специфікованих для нього.

Зближення цих моделей реалізується в розширеному ER-моде- люванні (Extended Entity Relationship model, EER-модель).

Модель "сутність - зв’язок". Оскільки реальний світ складається із сутностей та зв’язків, модель "сутність - зв’язок” можна розглядати як універсальний спосіб подання даних. Основна мета побудови моделі "сутність - зв’язок” - забезпечення найбільш природного для людини способу збору та представлення даних і відомостей, які будуть зберігатися в базі даних.

Сутність - певний відокремлений об’єкт (який можна відрізнити від інших), відомості про який необхідно зберігати в базі даних.

При цьому розрізняють поняття тип сутності та екземпляр сутності. До типу сутності відносять набір однорідних даних, а кожний елемент набору буде екземпляром сутності. Наприклад, типом сутності може бути список землевласників, кожен з яких окремо буде його екземпляром.

Зв’язок - асоціювання двох або більше сутностей.

Оскільки в базі даних потрібні користувачеві дані можуть стосуватися різних сутностей, то необхідно вказати їх взаємозв’язок. Наприклад, сутність Товари у моделі даних Склад пов’язана із двома сутностями Постачальник та Споживач. При цьому зрозуміло, що один і той самий тип товару можуть постачати різні постачальники, а споживати конкретний екземпляр товару - тільки конкретний споживач. Характер зв’язків між елементами бази даних визначає модель організації даних. Найбільш відомими є ієрархічна, мережева та реляційна моделі даних.

Модель "сутність - зв’язок" (англ. Entity-relationship (ER) model або entity-relationship diagram) - модель даних, яка дозволяє описувати концептуальні схеми за допомогою узагальнених конструкцій блоків. ER- модель була запроваджена Пітером Пін-Шен Ченом[1] (Peter Chen) у 1976 р., який запропонував представляти ПС за допомогою графічних діаграм, що містять невелику кількість різнорідних компонентів.

Надалі багатьма авторами були розроблені свої варіанти подібних моделей (нотація Мартіна, нотація IDEF1X, нотація Баркера тощо). Крім того, різні

програмні засоби, що реалізують одну й ту ж нотацію, можуть відрізнятися своїми можливостями. По суті, всі варіанти діаграм "сутність - зв'язок" виходять з однієї ідеї - малюнок завжди наочніший від текстового опису. Такі діаграми використовують графічне зображення сутностей предметної сфери, їх властивостей (атрибутів) і взаємозв'язків між сутностями.

ER-модель зручна при проектуванні інформаційних систем, баз даних, архітектур комп'ютерних додатків та інших систем (моделей). За допомогою такої моделі виділяють найсуттєвіші елементи (вузли, блоки) моделі і встановлюють зв'язки між ними.

Основними поняттями моделі "сутність - зв'язок" є сутність, зв 'язок і атрибут.

Сутність (entity) - це реальний або уявний об'єкт, інформація про який повинна зберігатись у БД і бути доступною.

Кожна сутність повинна мати найменування, виражене іменником в однині. Прикладами сутностей можуть бути такі класи об'єктів, як "Район", "Ґрунт", "Водозбори". Для позначення сутності об'єкта в різних системах використовуються прямокутники, блоки з заокругленими кутами, овали з найменуванням тощо (рис. 3.5):

Рис. 3.5. Зображення сутності

Примірник сутності - це конкретний представник даної сутності.

Наприклад, представником сутності "Район" може бути "Район Деснянський". Екземпляри сутностей повинні бути помітними, тобто сутності повинні мати певні властивості, які б були унікальними для кожного екземпляра цієї сутності.

При визначенні типу сутності необхідно гарантувати, що кожний екземпляр сутності може бути відрізнений від будь-якого іншого екземпляра тієї ж сутності.

Атрибут сутності - це іменована характеристика, яка виступає певною властивістю сутності і слугує для уточнення, ідентифікації, класифікації, числової характеристики або виразу стану сутності.

Імена атрибутів заносяться в прямокутник, що зображує сутність, під ім'ям. Найменування атрибута має бути виражене іменником в однині (іноді можливо з прикметником, який його характеризує). Прикладами атрибутів сутності "Співробітник" можуть бути такі атрибути, як "Табельний номер", "Прізвище", "Ім'я", "По батькові", "Посада", "Зарплата" тощо. Атрибути зображуються в межах прямокутника, що визначає сутність (рис. 3.6).

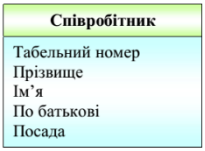


Рис. 3.6. Зображення атрибутів

Набір сутностей (entity set) - множина сутностей одного типу (що мають однакові властивості).

Сутність фактично є множиною атрибутів, що описують властивості всіх членів даного набору сутності.

Ключ сутності - це набір ненадлишкових атрибутів, значення яких у сукупності є унікальними для кожного екземпляра сутності.

Ненадлишковість атрибута означає, що видалення будь-якого атрибута з ключа порушує його унікальність.

Сутність може мати кілька різних ключів. Ключові атрибути зображуються на діаграмі підкресленням (рис. 3.7).

Зв'язок - це графічно зображувана асоціація, що встановлюється між двома типами сутностей.

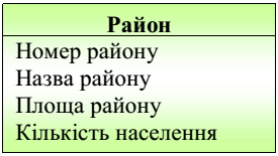


Рис. 3.7. Зображення ключа

Одна сутність може бути пов'язана з іншою сутністю або сама з собою. Зв'язки дозволяють за однією сутністю знаходити інші, пов'язані з нею. Наприклад, зв'язки між сутностями можуть виражатися наступними фразами - "СПІВРОБІТНИК може мати декілька ДІТЕЙ", "кожен СПІВРОБІТНИК зобов'язаний числитися тільки в одному ВІДДІЛІ".

Зв'язок подається у вигляді неспрямованої лінії, яка з'єднує дві сутності або яка веде від сутності до неї самої (рис. 3.8):

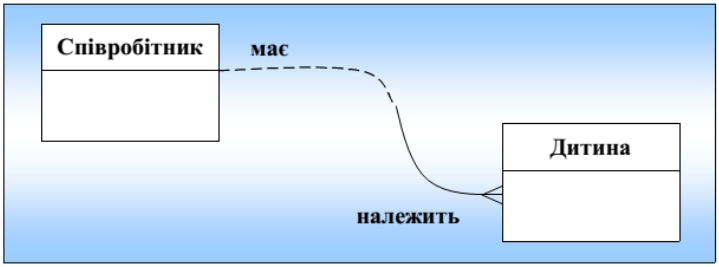


Рис. 3.8. Зображення зв'язку

В будь-якому зв'язку, відповідно до існуючої пари сутностей, що пов'язані між собою, виділяються два кінці. На кожному кінці зв'язку зазначаються ім 'я

кінця зв ’язку, ступінь кінця зв ’язку (скільки екземплярів даного типу сутності повинно бути присутнім у кожному екземплярі даного типу зв’язку), обов’язковість зв’язку (чи повинен будь-який екземпляр даного типу сутності брати участь у певному екземплярі даного типу зв’язку).

Примітка. Спостерігаються відмінності в способі зображення характеру зв’язків між об’єктами (використання "стрілок", "лапок", "точок" тощо для відображення "множинного" кінця зв’язку). Вони не накладають жодного відбитку на методологію побудови концептуальної моделі і алгоритм наступного переходу до даталогічної моделі.

Бажано, щоб позначення, що використовуються, були інтуїтивно зрозумілими, занадто не захаращували модель, були прості в зображенні. Найчастіше переваги розробників у використанні тих або інших позначень визначаються просто звичкою. За можливості треба намагатись використовувати стандартизовані позначення або які широко поширені.

Зв’язки можуть бути факультативними або обов ’язковими (рис. 3.9).

Рис. 3.9. Приклади обов’язкового і факультативного зв ’язку

Кожен зв’язок має два кінці і одне або два найменування. Найменування зазвичай виражається в невизначеній дієслівної формі: "мати", "належати" тощо. Кожне з найменувань відноситься до свого кінця зв’язку. Іноді найменування не пишуться через їх очевидність.

Кожен зв’язок може мати один із типів зв’язку представлених на рис. 3.10.

Зв’язок типу "один до одного” означає, що один примірник першої сутності (лівої) пов’язаний з одним примірником другої сутності (правої). Зв’язок "один до одного" найчастіше свідчить про те, що насправді ми маємо всього одну сутність, неправильно розділену на дві.

Зв’язок типу "один до багатьох” означає, що один примірник першої сутності (лівої) пов’язаний з декількома екземплярами другої сутності (правої). Це найбільш часто використовуваний тип зв’язку. Ліва сутність (з боку "один") називається батьківською. Права (з боку "багато") - дочірньою.

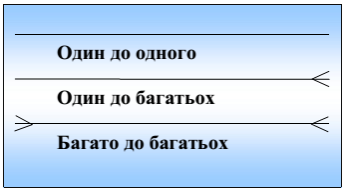


Рис. 3.10. Типи зв’язків

Зв’язок типу "багато до багатьох” означає, що кожен екземпляр першої сутності може бути пов’язаний з декількома екземплярами іншої сутності, і кожен примірник іншої сутності може бути пов’язаний з декількома примірниками першої сутності. Тип зв’язку "багато до багатьох є тимчасовим типом зв’язку, який припустимий на ранніх етапах розробки моделі. Надалі цей тип зв’язку повинен бути замінений двома зв’язками типу "один до багатьох” шляхом створення проміжної сутності.

Кожен зв’язок може мати одну з двох модальностей[2] зв’язку (рис. 3.11):

Рис. 3.11. Типи модальностей зв’язку

Модальність "може" означає, що екземпляр однієї сутності може бути пов’язаний з одним або кількома екземплярами іншої сутності, а може бути й не пов’язаний з жодним екземпляром.

Модальність "повинен" означає, що примірник однієї сутності зобов’язаний бути пов’язаний не менш ніж з одним примірником іншої сутності.

Зв’язок може мати різну модальність з різних кінців (рис. 3.11).

Описаний графічний синтаксис дозволяє однозначно читати діаграми, користуючись наступною схемою побудови фраз:

<Кожен примірник СУТНОСТІ 1> <Модальні ЗВ’ЯЗКИ> <НАЙМЕНУВАННЯ ЗВ’ЯЗКУ> <ТИП ЗВ’ЯЗКУ> <примірник СУТНОСТІ 2>

Кожен зв’язок може бути прочитаний як зліва направо, так і справа наліво. Зв’язок на рис. 3.8 можна читати зліва направо: "кожен співробітник може мати декілька дітей", справа наліво: "кожна дитина зобов ’язана належати рівно одному співробітнику".

Простота і наочність подання концептуальних схем баз даних у моделі "сутність - зв ’язок" стали причиною її широкого поширення й популярності серед розробників баз даних.

Діаграма Бахмана - це орієнтований граф, вершини якого відповідають групам (типам записів), а дуги - груповим відношенням (рис. 3.12).

Тут запис типу Поліклініка є власником записів типу Мешканець, і вони зв’язані груповим відношенням диспансеризація. Запис типу Організація також є власником записів типу Мешканець, і вони зв’язані груповим відношенням

працюють. Записи типу РЕУ і типу Мешканець є власниками записів типу Квартира з відношенням відповідно обслуговують і проживають. Таким чином, запис одного й того ж типу може бути членом одного відношенням і власником іншого.

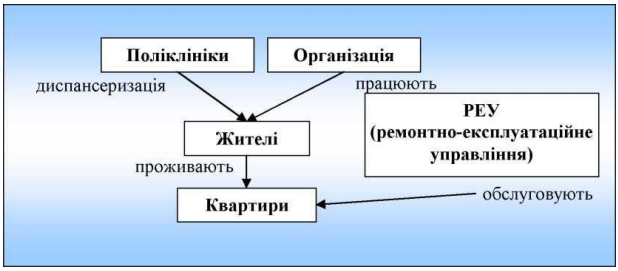


Рис. 3.12. Приклад діаграми Бахмана для фрагмента БД ”"Місто”

Організація моделі даних у СКБД мережевого типу визначається в термінах: елемент, агрегат, запис, групове відношення, БД.

Елемент - найменша одиниця структури даних.

Агрегат - іменована сукупність елементів або інших агрегатів.

Запис - агрегат, який не входить до складу жодного іншого агрегату і становить основну одиницю обробки БД.

Тип запису визначається складом її елементів.

Групове відношення - ієрархічне відношення між записами двох типів.

Записи одного типу є власниками відношення, іншого - підлеглими.

Мережева модель даних підтримує БД мережевої структури. В мережевій моделі даних припускаються такі операції над об’єктами:

Запам’ятати - заносить новий запис і автоматично включає в групове відношення з відповідною підпорядкованістю.

Вкл. в групове відношення - дозволяє зв’язати підпорядкований запис, що відповідає запису-власнику.

Переключити - змінює запис-власника в тому ж груповому відношенні.

Обновити - змінює значення елементів запису, перед оновленням відповідний запис повинен буди витягнутий.

Витягнути, Видалити, Виключити з групового відношення - розриває зв’язок між записом-власником і підпорядкованим.

Особливості обробки даних у мережевих моделях:

1. Основна одиниця обробки - запис.

2. Обробка може починатись з будь-якого запису, незалежно від її розташування в структурі.

3. Від конкретного запису можливий перехід як до запису запису- власника, так і до підпорядкованого запису.

3.3. Даталогічні моделі

На етапі даталогічного проектування розробляється перехід від інфологічної моделі до логічної (комп'ютерно орієнтованої), яка підтримується засобами конкретної СКБД. Процес переходу називається відображенням. На етапі даталогічного проектування необхідно вибрати тип СКБД і конкретну СКБД.

Основними чинниками, які впливають на даталогічне проектування з боку СКБД, є:

1. Тип логічної моделі, яку підтримують вибрані СКБД.
2. Особливості фізичної організації даних, вибраних СКБД.
3. Кількісні обмеження, які накладає СКБД.
4. Тип обчислювального середовища, в якому може працювати СКБД і необхідні обчислювальні ресурси.
5. Документування, і вартість реалізації.

Даталогічні моделі поділяються на документальні і фактографічні (рис. 3.13).

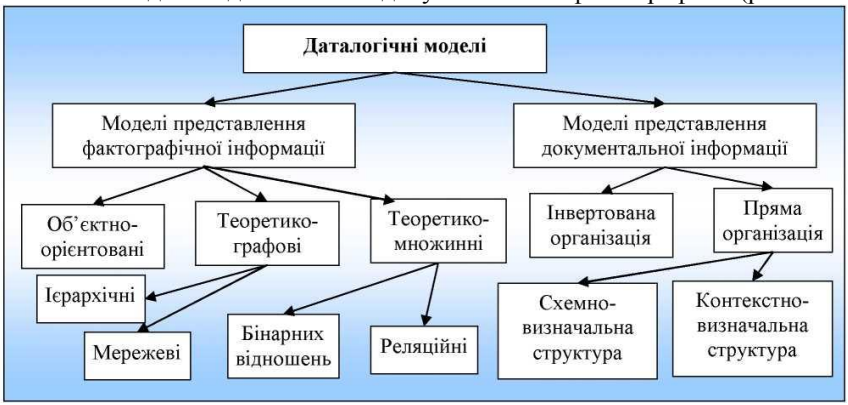


Рис. 3.13. Приклад діаграми Бахмана для фрагмента БД "'Місто "

Документальні моделі даних відповідають представленню слабо структурованої інформації, яка орієнтована головним чином на вільні формати документів, наприклад, тексти природної мови або з гіпертекс- тової розмітки. Документальні моделі класифікують на орієнтовані на формат документа, дескрипторні і тезаурусні моделі.

Моделі, орієнтовані на формат документа. Пов'язані передусім зі стандартною мовою розмітки документів - SGML (Standart Generalised Markup Language), яка була затверджена ISO як стандарт ще в 80-х рр. XX ст. Ця мова призначена для створення інших мов розмітки, визначає припустимий набір тегів[3] (посилань), їх атрибути і внутрішню структуру документа. Контроль за правильністю використання тегів здійснюється за допомогою спеціального набору правил, які називаються DTD-описами і використовуються програмою клієнта при розборці документа. Для кожного класу документів визначається свій набір правил, що описує граматику відповідної мови розмітки. За допомогою SGML можна описувати структуровані дані, організовувати інформацію, що міститься в документах, подавати цю інформацію в певному стандартизованому форматі. Але зважаючи на свою складність, SGML

використовувалась головним чином для опису синтаксису інших мов (найбільш відомою з яких є HTML), і невелика кількість додатків працювали з SGML-документами напряму.

HTML виявилася набагато простішою і зручнішою за мову SGML і дозволяє визначати оформлення елементів документа та має певний обмежений набір інструкцій - тегів, за допомогою яких здійснюється процес розмітки. Інструкції HTML у першу чергу призначені для керування процесом виведення змісту документа на екрані програми-клієнта, визначаючи при цьому спосіб представлення документа, але не його структуру. Елементом гіпертекстової бази даних, що описується HTML, виступає текстовий файл, який може легко передаватись по мережі з використанням протоколу HTTP. Ця особливість, а також те, що HTML є відкритим стандартом і величезна кількість користувачів має можливість використовувати можливості цієї мови для оформлення своїх документів, безумовно, вплинули на зростання популярності HTML і зробили її сьогодні головним механізмом представлення інформації в Інтернеті.

Однак сьогодні HTML вже не задовольняє певною мірою вимоги, що висуваються сучасними розробниками до мов подібного роду. Тому їй на зміну була запропонована нова мова гіпертекстової розмітки, потужна, гнучка і, одночасно з цим, зручна мова XML.

XML (Extensible Markup Language) - це мова розмітки, яка описує цілий клас об’єктів даних, що називаються XML-документами. Вона використовується як засіб для опису граматики інших мов і контролю за правильністю складання документів. Тобто сама по собі XML не містить жодних тегів, призначених для розмітки, вона просто визначає порядок їх створення.

Тезаурусні моделі засновані на принципі організації словників, містять певні мовні конструкції і принципи їх взаємодії в заданій граматиці. Ці моделі досить широко й ефективно використовуються в системах- перекладачах, особливо багатомовних.

Дескрипторні моделі - найпростіші документальні моделі, що широко застосовувались на ранніх стадіях використання документальних баз даних. У цих моделях кожному документу призначався дескриптор[4], який описував документ за певними ключовими характеристиками. Цей дескриптор мав жорстку структуру і описував документ відповідно до тих характеристик, які вимагались для роботи з документами в базі даних, що розроблялась. Наприклад, для БД, що містила опис патентів, дескриптор включав назву області, до якої відносився патент, номер патенту, дату видачі патенту і ще низку ключових параметрів, які заповнювались для кожного патенту. Обробка інформації в таких базах даних велась виключно за дескрипторами, тобто за тими параметрами, які характеризували патент, а не сам текст патенту.

Фактографічні моделі даних описують процеси чи об’єкти, що характеризують який-небудь факт чи подію. Такі дані подаються в структурованому вигляді; над ними можна виконувати різноманітні допустимі операції.

Фактографічні моделі даних на відміну від документальних моделей відповідають представленню про структуровану інформацію. До таких моделей відносяться моделі даних на основі інвертованих списків, теоретико-графові і теоретико-множинні моделі даних, а також об'єктно- орієнтовані моделі даних.

Моделі даних на основі інвертованих файлів. У 60-х рр. XX ст. з'явились СКБД на основі інвертованих файлів, які відрізнялись простотою організації і наявністю мов маніпулювання даними.

БД, заснована на застосуванні інвертованих файлів, є найпростішою і становить набір файлів, доступ до яких здійснюється за допомогою програм, у яких чітко прописано, в яких позиціях у файлі знаходиться необхідна інформація. Наприклад, якщо основний файл містить поля: "ім'я", "по батькові", "прізвище", "номер відділу", "номер кімнати", то, упорядкувавши цей файл за першими трьома полями, можна легко знаходити для кожного співробітника відділ, де він працює, або кімнату, де знаходиться його робоче місце. Однак, щоб отримати список усіх співробітників якогось відділу, таке упорядкування буде марним. Хоча можна інвертувати список, упорядкувавши його за відділами. Аналогічно, якщо виникне потреба визначення всіх співробітників, які працюють у певній кімнаті, то можна побудувати ще одне упорядкування - за номерами кімнат.

Недоліком такого підходу є те, що він не задовольняє моделі ANSI/SPARC, оскільки і логічна і фізична структура збереження даних[4] прихована в програмі. У міру розширення системи зростає кількість файлів і програм, які обробляються, що призводить до дублювання даних. При цьому безпека системи забезпечується лише на рівні операційної системи, що призводить до зниження ефективності роботи.

До числа найбільш відомих і типових представників таких систем відносяться Datacom/DB компанії Applied Data Research, Inc. (ADR), орієнтована на використання на машинах основного класу фірми IBM, і Adabas компанії Software AG.

Організація доступу до даних на основі інвертованих списків використовується практично в усіх сучасних реляційних СКБД, але в цих системах користувачі не мають безпосереднього доступу до інвертованих списків (індексів).

База даних, організована за допомогою інвертованих списків, схожа на реляційну БД, але з тією відмінністю, що збережені таблиці і шляхи доступу до них видні користувачам. При цьому:

- рядки таблиць упорядковані системою в певній фізичній послідовності;
- фізична упорядкованість рядків усіх таблиць може визначатися і для усієї БД (так робиться, наприклад, в Datacom/DB);
- для кожної таблиці можна визначити довільне число ключів пошуку, для яких будуються індекси. Ці індекси автоматично підтримуються системою, але явно видні користувачам.

Теоретико-графові моделі даних є ранніми моделями даних, що стали домінувати в галузі проектування баз даних, починаючи з 70-х рр. XX ст.

Перша версія ієрархічної моделі даних з'явилась у 1968 р. Простота організації, наявність заздалегідь заданих зв'язків між сутностями, схожість з фізичними моделями даних дозволяли домогтися прийнятної продуктивності ієрархічних СКБД на повільних комп'ютерах з малим об'ємом пам'яті.

Ієрархічні БД підтримують деревоподібну організацію інформації (рис. 3.14).

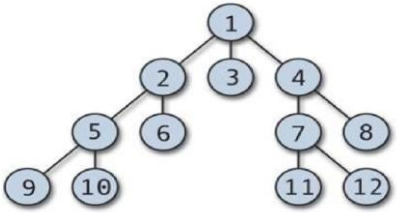


Рис. 3.14. Приклад ієрархічної бази даних

Зв'язки між записами виражаються у вигляді відношень "предок - нащадок", а у кожного запису є рівно один батьківський запис. Це допомагає підтримувати посилальну цілісність. Коли запис видаляється з дерева, всі її нащадки також повинні бути видалені. Проте, якщо дані не мають деревоподібної структури, то виникають складнощі при побудові бази даних.

Ієрархічна структура даних є найпростішою серед усіх даталогічних моделей і визначається ієрархічною впорядкованістю своїх компонентів (або вузлів).

Ієрархічна структура передбачає:

• кожний вузол на більш низькому рівні пов'язується тільки з одним вузлом, що знаходиться на більш високому рівні, тобто кожен вузол має не більше одного "батька" - старшого за ієрархією вузла;

• ієрархічне дерево має тільки один вузол, не підпорядкований жодному іншому вузлу і який знаходиться на найвищому верхньому - першому рівні;

• до кожного вузла бази даних існує тільки один ієрархічний шлях від кореневого вузла.

Ієрархічна модель даних будується за принципом ієрархії типів об'єктів, тобто один тип об'єкта визначається головним, а інші, що перебувають на більш низьких рівнях ієрархії - підпорядкованими. В теорії графів така структура називається коренем дерева.

Між головним і підпорядкованими об'єктами встановлюється зв'язок "один до багатьох".

Кількість дерев у базі даних визначається кількістю кореневих записів.

Орієнтація на кореневому дереві визначається або від коріння, або до коріння (рис. 11.15).

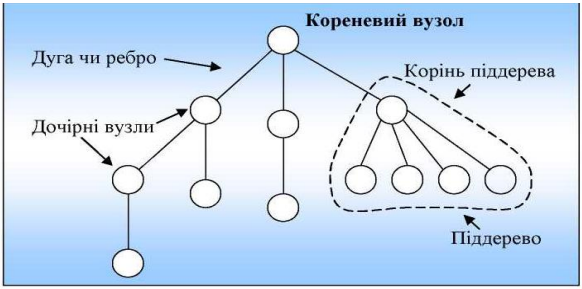


Рис. 3.15. Узагальнений вигляд деревоподібної структури

Верхній вузол називається кореневим вузлом. Він може мати нуль чи декілька дочірніх вузлів, які, у свою чергу, також можуть мати нуль чи декілька дочірніх вузлів. У результаті цього подібна структура може бути визначена рекурсивно. Всі вузли дерева, за винятком кореня, повинні мати батьківський вузол. Довільна частина дерева, що виходить з одного вузла (окрім кореня дерева), називається піддеревом. Якщо до верхівки не заходить жодне ребро, дерево стає орієнтованим.

Основними інформаційними одиницями ієрархічної структури БД є сегмент і поле.

Поле даних - мінімальна, неподільна одиниця даних, що доступна користувачу за допомогою СКБД.

Сегмент називається записом, при цьому визначаються 2 поняття: тип сегмента або тип запису; екземпляр сегмента або екземпляр запису.

Тип сегмента - це поіменована сукупність типів елементів даних, що в нього входять.

Екземпляр сегмента - конкретні значення полів або елементів даних, що входять до нього.

Кожний тип сегмента в рамках моделі утворює певний набір однорідних записів. Для можливості розрізнення окремих записів в одному наборі кожен тип сегмента повинен мати ключ або набір ключових атрибутів (полів, елементів даних).

Ключем називається набір елементів даних, які однозначно ідентифікують екземпляр сегмента.

При цьому вважають, що спрямовані ребра графів відбивають ієрархічні зв’язки між сегментами кожного екземпляра. Тип сегмента, що перебуває на найвищому рівні ієрархії, називається логічно вихідним по відношенню до типів сегментів, які з’єднані з даними, спрямовані ієрархічно ребрами.

При роботі з узагальненою деревоподібною структурою використовуються два методи доступу до всіх вузлів (типів записів) усередині дерева. Перший метод починається з доступу до кореня з наступним обробленням всього дерева та доступом до піддерев у порядку зліва направо. Це прямий порядок обходу дерева. Другий метод починається із доступу до найнижчих вузлів з поступовим висхідним переходом від одного піддерева до другого зліва направо та із

завершенням обробки в корені. Цей метод називають зворотним порядком обходу дерева.

До переваг ієрархічної моделі даних відносять простоту організації, наявність заздалегідь заданих зв'язків між сутностями, подібність з фізичними моделями даних. Це дозволило домогтися прийнятної продуктивності ієрархічних СКБД на повільних ЕОМ того часу з доволі обмеженими обсягами пам'яті.

Але якщо дані не мали деревоподібної структури, то виникало безліч труднощів при побудові ієрархічної моделі й бажанні домогтися потрібної продуктивності.

До недоліків ієрархічної моделі треба віднести її громіздкість для оброблення інформації з достатньо складними логічними зв'язками, а також складність її розуміння для звичайного користувача.

Приклади ієрархічних СКБД:

1) IMS (Information Management System) фірми IBM (перша версія 1968 р.);

2) TDMS (Time-Shared Data Management System) компанії Development Corporation;

3) Mark IV Multi-Access Retrieval System компанії Control Data Corporation;

4) System 2000 виробництва SAS-Institute;

5) сервери каталогів, такі як LDAP і Active Directory (допускають чітке подання у вигляді дерева);

6) реєстр операційної системи Windows, що побудований за принципом ієрархічної БД. Google App Engine Datastore API тощо.

Мережеві моделі також створювались для малоресурсних комп'ютерів. Мережева модель розширює ієрархічну модель, дозволяючи групувати зв'язки між записами в множині. З логічної точки зору зв'язок - це не сам запис. Зв'язки лише висловлюють відношення між записами. Як і в ієрархічній моделі, зв'язки ведуть від батьківського запису до дочірнього, але в цьому разі підтримується множинне спадкування (рис. 3.16).

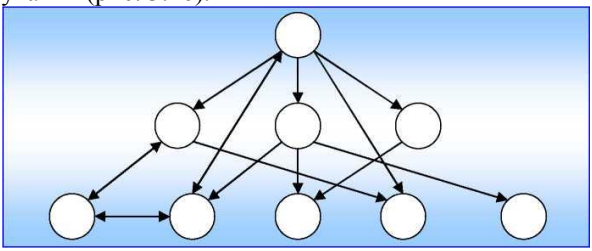


Рис. 3.16. Приклад мережевої моделі даних

Застосування мережевої моделі даних дозволило збільшити продуктивність СКБД, проте суттєво їх ускладнила.

Складність практичного використання ієрархічних і мережевих СКБД змушувала шукати інші способи представлення даних.

Теоретико-множинні моделі даних. У червні 1970 р. співробітник IBM Едгар Кодд опублікував статтю "Реляційна модель для великих банків спільно використовуваних даних", яка перевернула теорію баз даних і принесла доктору

Кодду нагороду Тюрінга в 1981 р. В реляційних моделях даних ключовим поняттям є відношення (від англійського слова "relation"), що становить собою множину елементів, які називаються кортежами, кожному з яких відповідає атрибут.

У реляційній моделі даних об'єкти і взаємозв'язки між ними подаються за допомогою таблиць (рис. 3.17).

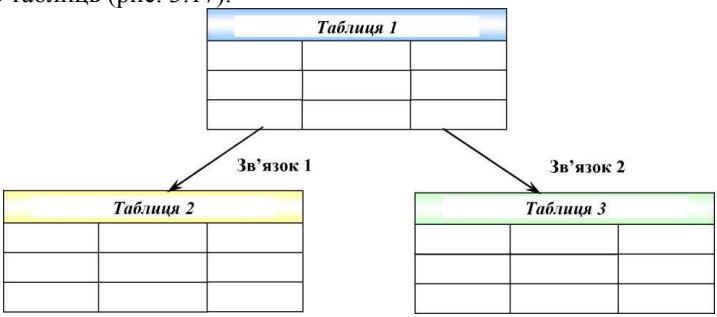


Рис. 3.17. Реляційна модель

Взаємозв'язки також розглядаються як об'єкти. Кожна таблиця подає один об'єкт і складається із рядків (запис, кортеж) і стовпчиків (поле, домен).

У реляційній базі даних кожна таблиця повинна мати первинний ключ-поле або комбінацію полів, які єдиним чином ідентифікують кожний рядок у таблиці. Завдяки своїй простоті і природності подання реляційна модель одержала надзвичайно широке поширення в БД.

Перевагами реляційної моделі даних є простота, зрозумілість і зручність фізичної реалізації.

Основним недоліком реляційної моделі даних є складність опису ієрархічних і мережевих зв'язків, однак це не завадило реляційній моделі широко використовуватись розробниками баз даних.

Сьогодні, крім реляційної моделі в чистому вигляді, СКБД підтримують її розширення, що є змішаними і доповненими моделями, до яких відносяться постреляційні і багатомірні моделі.

Постреляційна модель даних - це розширена реляційна модель, яка знімає обмеження неподільності даних, що зберігаються в кортежах відношень за допомогою використання багатозначних атрибутів, значення яких складаються з підзначень. Набір значень багатозначних атрибутів вважається самостійним відношенням, убудованим в основне відношення. Крім забезпечення вкладеності атрибутів, постреляційна модель підтримує асоційовані багатозначні атрибути.

Недоліком постреляційної моделі даних є складність розв'язку проблеми забезпечення цілісності даних.

Багатовимірні моделі почали активно використовуватись у 90-х рр. XX ст. для розв'язку задач аналізу і прийняття рішень. Основними поняттями, що використовуються в багатомірних СКБД, є агрегованість, історичність і прогнозованість даних, а також вимір і комірка.

Агрегованість даних - це розгляд інформації на різних рівнях її узагальнення.

Історичність даних передбачає забезпечення високого рівня статичності даних і їх взаємозв'язків, а також прив'язки даних до часу.

Прогнозованість даних передбачає задання функцій прогнозування і їх застосування до різних інтервалів часу.

Багатомірність моделі даних виражається в багатомірному логічному подані структури даних, що приводить до більш високої наочності і інформативності.

Вимірювання становить множину однотипних даних, які утворюють одну з граней гіперкуба. Коміркою називають поле, значення якого однозначно визначається фіксованим набором вимірів. На рис. 3.18 наведено приклад багатомірної моделі даних. Кожне значення комірки "Температура" однозначно визначається географічним місцем розташування, датою і часом доби. Для розв'язку практичних задач необхідно будувати гіперкуби з великою кількістю вимірів.

Об'єктно орієнтовані моделі даних. Об'єктно орієнтовані моделі даних виникли порівняно давно. Публікації з'явились у середині 80-х рр. XX ст.

В об'єктно орієнтованій моделі даних між записами бази даних і функціями їх обробки встановлюються взаємозв'язки за допомогою механізмів, подібних відповідним засобам в об'єктно орієнтованих мовах програмування.

Рис. 3.18. Приклад багатомірної моделі даних

Основними поняттями, що використовуються в об'єктно орієнтованих моделях даних, є поліморфізм, спадкування й інкапсуляція.

Інкапсуляція - це механізм, який об'єднує дані та методи, що маніпулюють цими даними, і захищає і те, й інше від зовнішнього втручання або неправильного використання. Коли методи і дані об'єднуються таким способом, створюється об'єкт.

Застосовуючи інкапсуляцію, ми начебто будуємо фортецю, яка захищає дані, що належать об'єкту, від можливих помилок, що можуть виникнути при прямому доступі до цих даних. Крім того, застосування цього принципу дуже часто допомагає локалізувати можливі помилки в коді програми. А це набагато спрощує процес пошуку і виправлення цих помилок.

Можна сказати, що інкапсуляція має на увазі приховування даних (data hiding), що дозволяє таким чином захистити ці дані.

Сутність інкапсуляції полягає в тому, що змінні стану об'єкта приховані від зовнішнього світу. Зміна стану об'єкта (його змінних) можлива тільки за допомогою його методів (операцій).

Успадковування поширює сферу видимості властивостей на усіх нащадків об'єкта.

Поліморфізм означає можливість мати в об'єктах різних типів методи з однаковими іменами. Дані в таких об'єктно орієнтованих СКБД здатні прийняти вид будь-якої структури, яку можна описати на використовуваній мові програмування. Відношення між об'єктами також можуть бути будь- якими складними.

При цьому розв'язується декілька дуже важливих проблем. По- перше, складні інформаційні структури виражаються в них краще, ніж у реляційних базах даних, а по-друге, усувається необхідність транслювати дані з того формату, який підтримується в СКБД.

Недоліком же об'єктно орієнтованих баз даних є їх тісний зв'язок з використовуваною мовою програмування, низька швидкість виконання запитів і складність роботи з даними.

3.4. Фізичні моделі

Фізична модель (рис. 3.19) БД є комп'ютерно орієнтованою і містить структури даних, що зберігаються в пам'яті ЕОМ, включаючи опис форматів даних, порядок їх логічного чи фізичного упорядкування, розміщення за типами пристроїв, а також характеристики і шляхи доступу до даних.

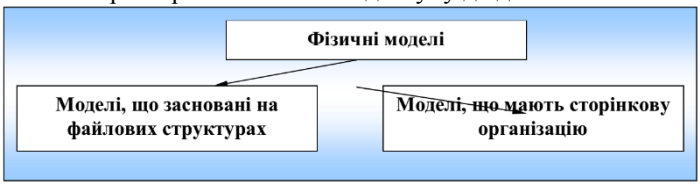


Рис. 3.19. Приклад багатомірноїмоделі даних

Від параметрів фізичної моделі залежать такі характеристики функціонування БД, як обсяг пам'яті і час реакції системи. Фізичні параметри БД можна змінювати у процесі її експлуатації (не змінюючи при цьому опису інших рівнів) з метою підвищення ефективності функціонування системи.

На даний час як фізичні моделі використовуються різні методи розміщення даних, засновані на файлових структурах (рис. 3.20): це організація файлів прямого і послідовного доступу, індексних файлів та інвертованих файлів, що використовують різні методи хешування, взаємопов'язаних файлів. Крім того, сучасні СКБД широко використовують сторінкову організацію даних.

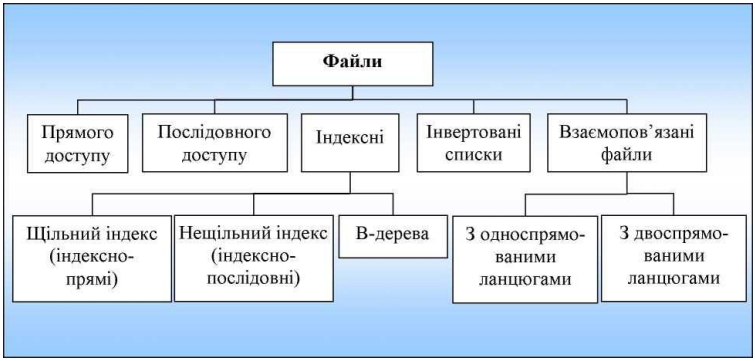


Рис. 3.20. Класифікація файлів, що використовуються в СКБД

[1]Пітер Пін-Шен Чен - американський професор комп'ютерних наук в університеті штату Луїзіана.

[2]Модальність (від лат. modus - розмір, спосіб, образ) у різних предметних сферах - категорія, що характеризує спосіб дії або відношення до дії.

[3]Теги розмітки - спеціальні команди для розміщення на екрані тексту, графіки, відео і аудіо-фрагментів, а також команди, що слугують для зв'язку з іншими HTML-документами та інтернет-ресурсами.

[4]Дескриптор (лат. descriptor - що описує) - лексична одиниця (слово, словосполучення) інформаційно-пошукової мови, яка виражає основний зміст певного тексту. Використовується при інформаційному пошуку документів в інформаційно-пошукових системах.

ЛЕКЦІЯ 4
РЕЛЯЦІЙНІ МОДЕЛІ ТА НОРМАЛІЗАЦІЯ ВІДНОШЕНЬ У НИХ

Як було відзначено раніше, реляційна модель даних була запропонована співробітником фірми IBM Едгаром Коддом, який виклав основні ідеї в статті ”A Relational Model of Data for Large Shared Data Banks” ("Реляційна модель для великих банків спільно використовуваних даних”).

Причина, за якою такі системи називають реляційними, полягає в тому, що англійській термін ”.relation”, по суті, є прийнятою математичною назвою таблиці.

Реляційна модель даних дозволила перебороти недоліки мережевих та ієрархічних моделей, а наочність і чітке теоретичне пророоблення принесли велику популярність моделі серед розробників баз даних.

4.1. Загальні відомості про реляційні моделі баз даних

Принципи реляційної моделі були сформульовані в 1969-1970 рр. Е. Ф. Коддом (E. F. Codd), який на той час працював у корпорації IBM.

Наприкінці 1968 р. Кодд, математик за освітою, вперше прийшов до висновку, що для впровадження в сферу баз даних строгих і точних принципів можна використати математичну теорію, зокрема теорію множин (об’єднання, перетин, різність, декартовий добуток) і логіку предикатів. Свої ідеї він виклав у статті "A Relational Model of Data for Large Shared Data Banks" ("Реляційна модель для великих банків спільно використовуваних даних"). Він показав, що будь-яке подання даних зводиться до сукупності двовимірних таблиць особливого виду, відомих у математиці як відношення (relation).

Запропонувавши реляційну модель даних, Кодд створив і інструментарій для зручної роботи з відношеннями - реляційну алгебру. Кожна операція цієї алгебри використовує одну або декілька таблиць (відношень) як її операндів й отримує в результаті нову таблицю, тобто дозволяє "розрізати" або "склеювати" таблиці (рис. 4.1).

Робота Е. Кодда сприяла появі великої кількості статей і книг, у яких реляційна модель отримала подальший розвиток. Найбільш розповсюджене трактування реляційної моделі даних належить К. Дейту. Згідно з Дейтом, реляційна модель складається з трьох частин:

структурної;
цілісної;
маніпуляційної.

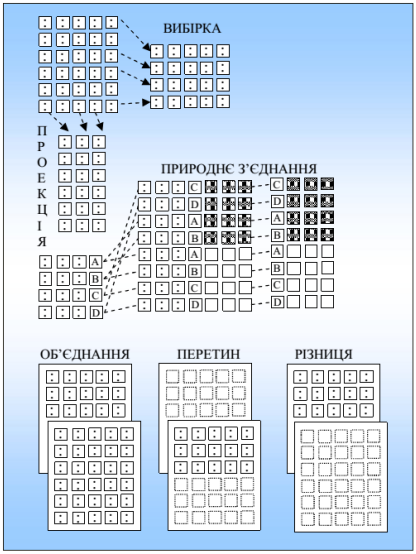


Рис. 4.1. Приклади операцій реляційної алгебри

Структурна частина описує, які саме об'єкти розглядаються реляційною моделлю. Постулюється, що єдиною структурою даних, які використовуються в реляційній моделі, є нормалізовані п-арні відносини.

Цілісна частина описує обмеження спеціального виду, що повинні виконуватися для будь-яких відносин у будь-яких реляційних базах даних. Це цілісність сутностеи і цілісність зовнішніх ключів (посилань).

Маніпуляційна частина описує два еквівалентні способи маніпулювання реляційними даними - реляційну алгебру і реляційне числення.

Реляційна алгебра в явному вигляді становить набір операцій, які можна використовувати, щоб повідомити системі, як в БД з певних відношень реально побудувати необхідне відношення.

Реляційні оператори мають одну важливу властивість: вони замкнені відносно поняття відношення. Це означає, що вирази реляційної алгебри визначаються над відношеннями реляційних БД і результатом обчислень також є відношення. Оскільки результатом будь-якої реляційної операції є певне відношення, можна утворювати реляційні вирази, в яких замість відношення-операнда певної реляційної операції знаходиться вкладений реляційний вираз.

Вирази реляційної алгебри будуються на основі алгебраїчних операцій (високого рівня), і подібно тому, як інтерпретуються арифметичні і логічні вирази, вираз реляційної алгебри також має процедурну інтерпретацію. Інакше кажучи, запит, поданий мовою реляційної алгебри, може бути обчислений на основі елементарних алгебраїчних операцій з урахуванням їх старшинства і можливої наявності дужок.

Набір основних алгебраїчних операцій складається з восьми операцій, які діляться на два класи - теоретико-множинні операції і спеціальні реляційні операції, доповнені певними спеціальними операціями, специфічними для баз даних.

До складу теоретико-множинних операцій входять традиційні операції над множинами:

- об'єднання;
- перетин;
- різність;
- декартовий добуток.

Спеціальні реляційні операції включають:

- вибірку;
- проекцію;
- природне з'єднання;
- ділення.

Ці ідеї стали загальноприйнятими й істотно вплинули на всі аспекти технологій баз даних, а також на інші галузі інформаційних технологій, такі як штучний інтелект, обробка природних мов і проектування програмних засобів тощо.

До Е. Кодда в теорії БД використовувались лише певні, ретельно підібрані визначення. Наприклад, термін відношення в сфері IT практично не використовувався, на відміну від інших галузей. Це було пов'язано з тим, що більшість термінів, що використовувались на той час, були нечіткими і не могли застосовуватись для формальної теорії. Наприклад, якщо розглянути термін запис, то він застосовувався в різних контекстах (як екземпляр запису, як тип запису, як логічний запис, як фізичний запис, запис що підлягав збереженню, як віртуальний запис тощо). Через це в формальній реляційній моделі термін запис взагалі не використовується.

Основними поняттями реляційних баз даних є тип даних, домен, атрибут, кортеж, первинний ключ і відношення (рис. 4.2). Альтернативні варіанти термінів СКБД представлені в табл. 4.1.

Тип даних. Поняття "тип даних" у реляційній моделі даних цілком адекватне поняттю типу даних в мовах програмування. Зазвичай у сучасних реляційних базах даних припускається збереження символьних, числових даних, бітових рядків, спеціалізованих числових даних (таких як "гроші"), а також спеціальних "темпоральних" даних (дата, час, часовий інтервал).

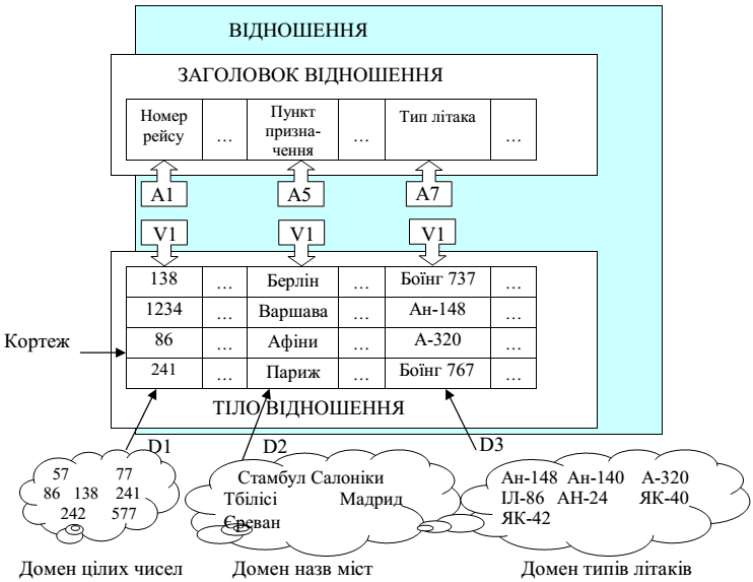


Рис. 4.2. Графічне відображення основних понять реляційних баз даних

Таблиця 4.1

Альтернативні варіанти термінів СКБД

Реляційні терміни	Табличний варіант	Файловий варіант	Об'єктна модель
Відношення	Таблиця	Файл	Клас
Кортеж	Рядок	Запис	Об'єкт
Атрибут	Стовпчик	Поле	Властивість

Домен - обмежена підмножина значень даного типу.

Домен характеризується такими властивостями:

• домен має унікальне ім 'я (в межах бази даних);

• домен є визначеним на певному простому типі даних або на іншому домені;

• домен може мати певну логічну умову, яка дозволяє описати підмножину даних, припустимих для даного домену;

• домен має певне змістове навантаження.

Термін кортеж приблизно відповідає поняттю рядка, аналогічне поняттю відношення, що приблизно відповідає поняттю таблиці.

Кортеж визначається як окрема сутність.

Доменом є набір усіх можливих значень певного атрибута відношення.

Схема відношення є списком імен атрибутів.

Атрибут - це властивість, яка описує сутність. У структурі відношення кожний атрибут має назву і тип.

Реляційна модель передбачає організацію даних тільки у вигляді таблиць. Таблиця складається з рядків і стовпчик (рис. 4.3).

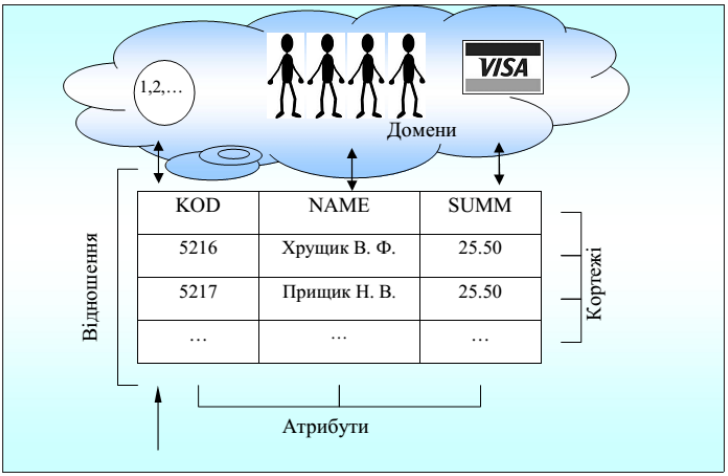


Рис. 4.3. Приклад табличної організації даних

Реляційна модель заснована на математичному понятті відношення, фізичним представленням якого є таблиця.

Будь-яка комп'ютерна реляційна модель, подана у вигляді таблиці, має такі властивості:

1. Кожен елемент таблиці – це один неподільний елемент даних - запис
2. Усі стовпці таблиці однорідні, тобто всі елементи в стовпці містять дані однакового типу і не перевищують визначеної довжини
3. Кожен стовпець має унікальне ім'я
4. Однакові рядки у таблиці відсутні
5. Порядок розміщення рядків і стовпців може бути довільним

Рис. 4.3. Властивості реляційної моделі даних

Сутність - опис об'єкта, дані про який зберігаються в базі даних.

Сутність - це інформація, яку необхідно зберігати. Звідси випливає, що сутності (а значить, і зв'язки) мають певні властивості (properties), які

відповідають тим даних про них, які необхідно зберігати в базі даних. Дані про сутність зберігаються у відношенні.

В загальному випадку властивості можуть бути як простими, так і складними, причому настільки простими або складними, наскільки це буде вимагатись потребами бази даних. Наприклад, властивість "місцезнаходження ділянки" - відносно проста: вона складається тільки з назви населеного пункту і може бути описана як простий символьний рядок. На противагу цьому сутність "ґрунт" може мати властивість "морфологічний опис" з достатньо складною структурою, що включає розчленування на генетичні горизонти, кольори і їх забарвлення, структуру тощо з відповідним текстовим описом.

К. Дейт[1] визначає три основні класи сутностей: стрижневі, асоціативні і характеристичні, а також підклас асоціативних сутностей - позначення.

Стрижнева сутність (стрижень) - це незалежна сутність.

Асоціативна сутність (асоціація) - це зв'язок типу Б:Б між двома або більше сутностями чи екземплярами сутності.

Асоціації розглядаються як повноправні сутності, які або можуть брати участь в інших асоціаціях і позначатись аналогічно стрижневим сутностям, або мати не тільки набір ключових атрибутів, необхідних для вказівки зв'язків, але й будь-яку іншу кількість атрибутів, що характеризують зв'язок.

Характеристична сутність (характеристика) - це зв'язок типу Б:1 або 1:1 між двома сутностями (окремий випадок асоціації). Єдиною метою характеристики в рамках аналізованої предметної сфери є опис або уточнення деякої іншої сутності. Потреба у них виникає в зв'язку з тим, що сутності реального світу мають іноді багатозначні властивості. Наприклад, книга може мати декілька характеристик перевидання (доповнене, перероблене тощо).

В реляційній моделі даних до відношень пред'являються певні вимоги, які виражаються в принципі інформаційної неподільності: кожне значення відношення реляційної моделі може містити тільки одну порцію даних - і принципі інформаційного кодування: неприпустимо, щоб в значенні відношення реляційної моделі містилось більше однієї порції даних.

4.2. Ключі

Зв'язки між кортежами реалізуються у вигляді ключів - певним чином позначених атрибутів відношення.

Ключ - це стовпчик (або декілька стовпчик), що додається до таблиці і дозволяє встановити зв'язок із записами в іншій таблиці.

Існують ключі двох типів: первинні і вторинні (зовнішні).

Первинний ключ - це одне або кілька полів (стовпчик), комбінація значень яких однозначно визначає кожний запис у таблиці. Первинний ключ не допускає значень Null і завжди повинен мати унікальний індекс. Первинний ключ використовується для зв'язування таблиці з зовнішніми ключами в інших таблицях (рис. 4.4).

Зовнішній (вторинний) ключ - це одне або декілька полів (стовпчиків) у таблиці, що містять посилання на поле або поля первинного ключа в іншій таблиці. Зовнішній ключ визначає спосіб об'єднання таблиць.

З двох логічно пов'язаних таблиць одну називають таблицею первинного ключа, або головною таблицею, а іншу - таблицею вторинного (зовнішнього) ключа, або підпорядкованою таблицею.

Таблиця Themes PK

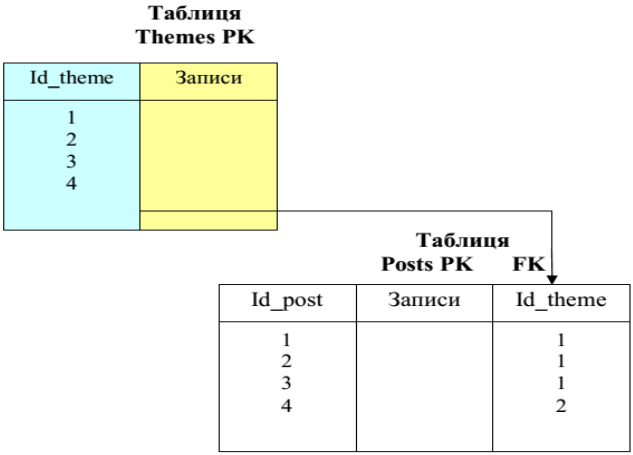


Рис. 4.4. Пов'язування таблиць ключем

СКБД дозволяють зіставити споріднені записи з обох таблиць і спільно вивести їх у формі, звіті або запиті.

Існує три типи первинних ключів: ключові поля лічильника (лічильник), простий ключ і складовий ключ.

Поле лічильника (тип даних "Лічильник"). Для кожного запису цього поля таблиці автоматично заноситься унікальне числове значення.

Простий ключ. Якщо поле містить унікальні значення, такі як коди чи інвентарні номери, то це поле можна визначити як первинний ключ. Як ключ можна визначити всі поля, що містять дані, якщо це поле не містить повторювані значення або значення Null.

Складові ключі. У випадках, коли неможливо гарантувати унікальність значень кожного поля, існує можливість створити ключ, що складається з декількох полів. Найчастіше така ситуація виникає для таблиці, використовуваної для зв'язування двох таблиць відношенням "багато до багатьох" (рис.4.5).

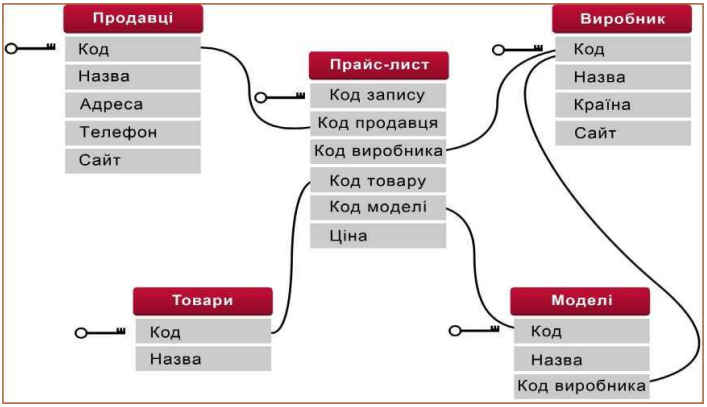


Рис. 4.5. Приклад складового ключа

Необхідно ще раз відзначити, що в полі первинного ключа повинні бути тільки унікальні значення в кожному рядку таблиці, тобто збіг не допускається, а в полі вторинного або зовнішнього ключа збіг значень у рядках таблиці допускається.

Якщо виникають труднощі з вибором потрібного типу первинного ключа, то як ключ доцільно обрати поле лічильника.

4.3. Зв'язування відношень

Структура даних може бути описана формально. Опис глобальної логічної структури бази даних називається схемою. Схема визначає всі типи елементів даних, які зберігаються в базі даних, а також усі зв'язки між ними. Схема бази даних, як правило, дуже складна. Конкретний користувач або прикладний програміст не повинен знати схему в цілому. Така необізнаність часто навіть необхідна з точки зору безпеки даних. Програміст або користувач повинен бути інформований тільки про множину даних і зв'язків, які орієнтовані на його конкретну предметну сферу.

Частина схеми отримала назву підсхеми. По суті, підсхема - це певна організація файлів прикладного програміста. В функції СКБД входить побудова відповідних підсхем із загальної схеми і передача даних користувачам і системним програмістам. При цьому схема даних повинна бути спроектована таким чином, щоб з неї могли бути побудовані всі підсхеми за запитами користувачів або прикладних програм. Ні схема, ні підсхема не визначають методів фізичного зберігання даних.

Схеми і підсхеми подають у вигляді діаграм, на яких зображують типи елементів даних і зв'язки між ними. Розрізняють чотири види зв'язків:

1) необов 'язковий зв 'язок: існування об'єктів не залежить від зв'язку;

2) можливий зв 'язок: існування одного з об'єктів залежить від зв'язку;

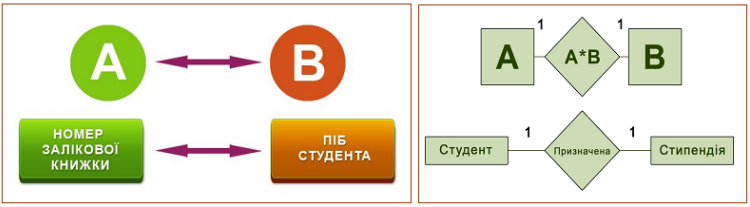
3) умовний зв’язок: частковий вид можливого зв’язку, коли задається умова існування (наприклад, зв’язок між об’єктами СТУДЕНТ, СТИПЕНДІЯ можливий за умови відповідної успішності);

4) обов’язковий зв’язок: існування обох об’єктів залежить від зв’язку. Односторонні зв’язки між парами елементів називаються асоціаціями, а двосторонні - відображеннями.

Між двома сутностями А і В можливі чотири типи зв’язків:

1) перший тип - зв’язок ОДИН-ДО-ОДНОГО (1:1): за допомогою такого відображення подають такий тип зв’язку, коли в кожний момент часу кожний екземпляр елемента, від якого спрямовано зв’язок, ідентифікує один і тільки один екземпляр елемента, до якого спрямовано зв’язок, при цьому ця ідентифікація є унікальною в обох напрямках. Приклад відображення 1:1 наведено на рис. 4.6. Якщо відомо значення А, то однозначно визначається і значення В. І навпаки.

2) другий тип - зв’язок ОДИН ДО БАГАТЬОХ (1 :Б): якщо екземпляр елемента даних, від якого спрямований зв’язок, ідентифікує певну кількість екземплярів елементів даних, до яких спрямовано зв’язок, причому ідентифікація в даному напрямку не обов’язково є унікальною, то таке відображення називається ОДИН ДО БАГАТЬОХ (1:Б). Прикладом такого відношення (рис. 4.7) може бути БУДИНОК - МЕШКАНЦІ. Будинок може пустувати, в ньому може жити один або декілька мешканців.



1 1

Код	Прізвище	Ім’я
1	*Іванов*	*Кузьма*
2	*Петров*	*Василь*
...		

Код	Рік народження	Адреса
1	*1992*	*Шевченка, б.20, кв.6*
2	*1993*	*Щорса вул.,б.30, кв.9*
...		

Рис. 4.6. Приклади відображення 1:1 ("один-до-одного ")

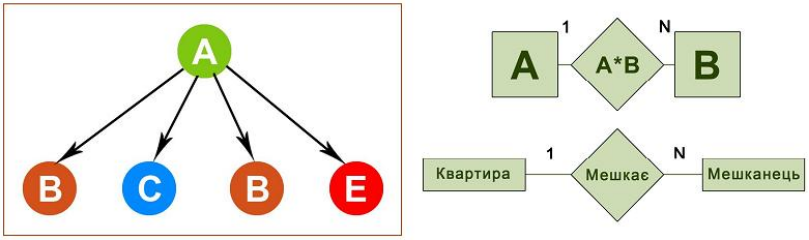


Рис. 4.7. Приклад відображення 1:Б ("один-до-багатьох ")

Оскільки між двома сутностями можливі зв'язки в обох напрямках, то існує ще два типи зв'язків:

3) БАГАТО ДО ОДНОГО (Б:1). Відображення Б:1 є аналогічним відображенням 1:Б;

4) БАГАТО ДО БАГАТЬОХ (Б:Б).

Якщо екземпляр елемента даних, від якого спрямовано зв'язок, ідентифікує певну кількість екземплярів елементів даних, до яких спрямовано зв'язок, і навпаки, тобто ідентифікація не є унікальною в обох напрямках, то таке відображення називається БАГАТО ДО БАГАТЬОХ (Б:Б).

Прикладом такого відношення (рис. 4.8) є відношення ВИКЛАДАЧІ - СТУДЕНТИ. Кожний студент "пов'язаний" з багатьма викладачами, і кожний викладач читає лекції різним групам студентів.

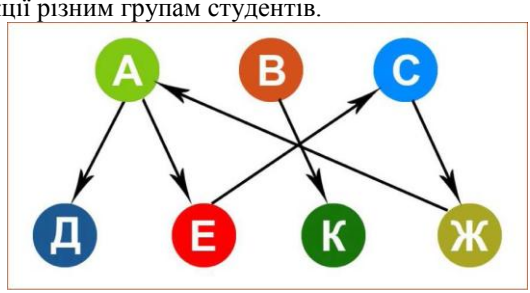


Рис. 4.8. Приклад відображення Б:Б ("багато-до-багатьох ")

Характер зв'язків між сутностями не обмежується переліченими. Існують і більш складні зв'язки:

1) множина зв'язків між одними й тими ж сутностями. Наприклад, пацієнт, маючи одного лікаря, що лікує, може мати також декілька лікарів-консультантів; лікар може бути лікарем, що лікує декількох пацієнтів, та може одночасно консультувати декількох інших пацієнтів;

2) тренарні зв'язки. Наприклад, лікар може призначити декільком пацієнтам декілька аналізів, аналіз може бути призначений декількома лікарями декільком пацієнтам, й пацієнту може бути призначено декілька аналізів декількома лікарями;

3) зв'язки більш високих порядків, семантика (зміст) яких іноді дуже складна.

Існує три типи асоціацій:

• асоціація типу 1 (проста);
• асоціація типу М (складна);
• асоціація типу С (умовна).

В простій асоціації типу 1 (рис. 4.9) екземпляр елемента даних, від якого спрямовано зв'язок, ідентифікує один і лише один екземпляр елемента даних, до якого спрямовано зв'язок. Ця ідентифікація є унікальною й визначає функціональну залежність.

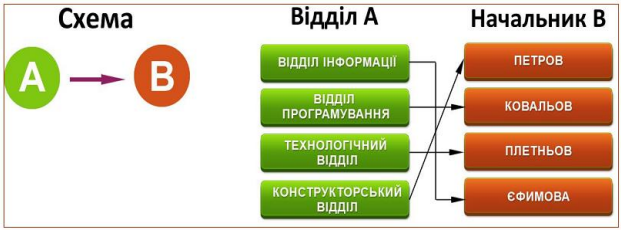


Рис. 4.9. Приклад простої асоціації типу 1

У складній асоціації типу М (рис. 4.10) екземпляр елемента даних, від якого спрямовано зв'язок, ідентифікує деяке число екземплярів елементів даних, до яких спрямовано зв'язок.

Ідентифікація є багатозначною залежністю і не обов'язково унікальною. При цьому зв'язок у зворотному напрямку не розглядається.

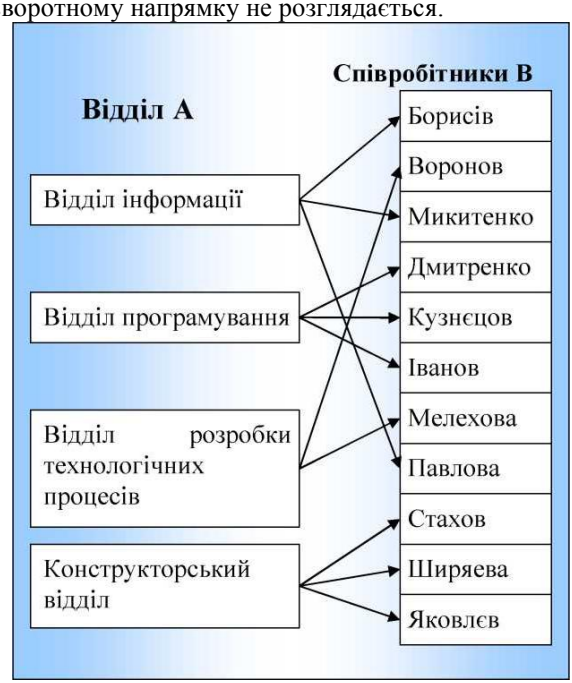


Рис. 4.10. Приклад складної асоціації типу М

В умовній асоціації типу С (рис. 4.11) для даного екземпляра елемента даних, від якого спрямовано зв'язок, може не існувати відповідного екземпляра елемента даних, до якого спрямовано зв'язок. Якщо він існує, то відноситься до єдиного екземпляра елемента даних. Наприклад, ПІБ РОБІТНИКА і ДАТА ЗВІЛЬНЕННЯ.

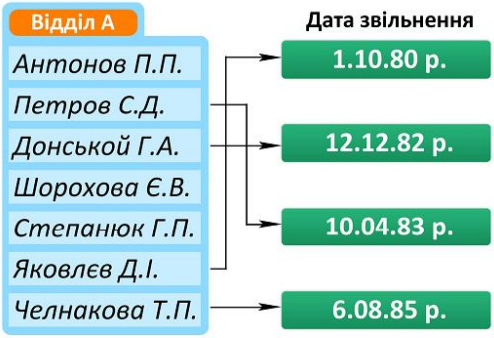


Рис. 4.11. Приклад умовної асоціації типу С

4.4. Реляційні операції

Реляційна модель даних заснована на застосуванні реляційної алгебри Едгара Кодда.

Реляційна алгебра - це теоретична мова операцій, яка дозволяє створювати на основі одного або декількох відношень інше відношення без зміни самих вихідних відношень. Оскільки обидва, операнд і результат, є відношеннями, то результати однієї операції можуть застосовуватись в іншій операції. Це дозволяє створювати вкладені вирази реляційної алгебри (за аналогією з тим, як створюються вкладені арифметичні вирази), але при будь-якій глибині вкладеності результатом є відношення. Така властивість називається замкнутістю. Вона підкреслює те, що застосування будь-якої кількості операцій реляційної алгебри до відношень не призводить до створення інших об'єктів, крім відношень, аналогічно тому, як результатами арифметичних операцій з числами є тільки числа.

Реляційна алгебра є мовою послідовного використання відношень, у якому всі кортежі, навіть взяті з різних відношень, обробляються однією командою, без організації циклів. Для команд реляційної алгебри запропоновано декілька варіантів синтаксису.

Існує декілька варіантів вибору операцій, які включаються в реля- ційну алгебру. Первісно Кодд запропонував вісім операцій, але згодом до них були додані ще декілька. П'ять основних операцій реляційної алгебри, а саме вибірка (selection), проекція (projection), декартовий добуток (cartesian product), об'єднання (union) і різність множин (set difference), які виконують більшість дій з добування даних. На підставі п'яти основних операцій можна також винести додаткові операції, такі як операції з'єднання (join), перетинання (intersection) і

ділення (division), які можуть бути виражені в термінах п’яти основних операцій (рис. 4.12, табл.4.1).

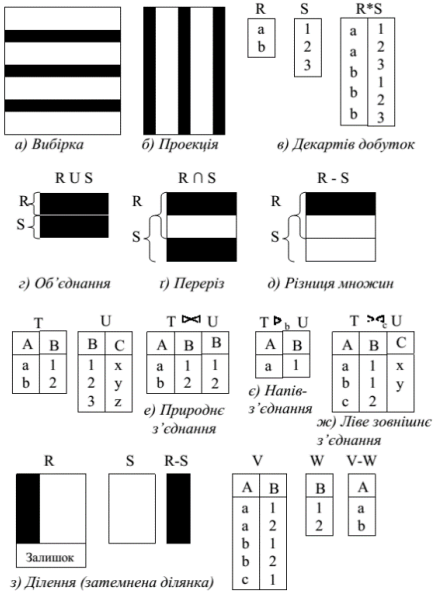


Рис. 4.12. Найпростіші операції над множинами

Реляційна алгебра Кодда має декілька недоліків:

По-перше, вісім перерахованих операцій за охопленням своїх функцій, з одного боку, надлишкові, оскільки мінімально необхідний набір складають п’ять операцій: об’єднання, віднімання, твір, проекція і вибірка. Три інші операції (перетин, з’єднання і поділ) можна визначити через п’ять мінімально необхідних. Так, наприклад, з’єднання - це проекція вибірки.

Таблиця 4.1

Основні операції над множинами

Операція	Позначення	Сфера застосування
Вибірка	$\sigma_{\text{предикат}}(R)$	Визначає результуюче відношення, яке містить тільки ті кортежі (рядки) з відношення R, що задовольняють задану умову (предикат)
Проекція	$\prod a_1, a_{2,} \dots a_n(R)$	Визначає нове відношення, що містить вертикальну підмножину відношення R, яка створюється за допомогою витягання значень зазначених атрибутів і виключення з результату рядків- дублікатів

Операція	Позначення	Сфера застосування
Декартів добуток	$R \times S$	Визначає нове відношення, яке є результатом конкатенації (зчеплення) кожного кортежу з відношення R з кожним кортежем з відношення S
Об'єднання	$R \cup S$	Визначає нове відношення, що містить усі кортежі, що містились тільки в R, тільки в S, одночасно в R і S, причому всі дублікати кортежів виключені. При цьому відношення R і S повинні бути сумісними по об'єднанню
Перетин	$R \cap S$	Визначає відношення, що містить кортежі, які присутні як у відношенні R, так і в відношенні S. Відношення R і S повинні бути сумісними по об'єднанню
Різність	$R - S$	Різність двох відношень R і S складається з кортежів, які є у відношенні R, але відсутні у відношенні S. При цьому відношення R і S повинні бути сумісними по об'єднанню
Тета-з'єднання	$R\ \ _f\ S$	Визначає відношення, що містить кортежі з декартового добутку відношень R і S, які задовольняють предикат F
З'єднання по еквівалентності	$R\ \ _f\ S$	Визначає відношення, яке містить кортежі з декартового добутку відношень R і S, що задовольняють предикат F (предикат повинен передбачати тільки порівняння на рівність)
Природне з'єднання	$R\ \ S$	Природним з'єднанням називається з'єднання по еквівалентності двох відношень R і S, виконане по усіх загальних атрибутах x, з результатів якого виключається по одному екземпляру кожного загального атрибута
(Ліве) зовнішнє з'єднання	$R \supset\ \ S$	З'єднання, при якому кортежі відношення R, які не мають співпадінь значень в загальних стовпчиках відношення S, також включаються в результуюче відношення
Напів-з'єднання	$R\ _f S$	Визначає відношення, що містить ті кортежі відношення R, які входять у з'єднання відношень R і S

Операція	Позначення	Сфера застосування
Ділення	$R \div S$	Визначає відношення, що складається з множини кортежів відношення R, які визначені на атрибуті С, який відповідає комбінації всіх кортежів відношення S, де С - множина атрибутів, які є у відношенні R, проте відсутні у

По-друге, цих восьми операцій недостатньо для побудови реальної СКБД на принципах реляційної алгебри. Необхідні розширення, що включають операції: перейменування атрибутів, утворення нових атрибутів, що обчислюються, обчислення підсумкових функцій, побудова складних алгебраїчних виразів, присвоєння, порівняння тощо.

Операції вибірки і проекції є унарними, оскільки працюють з одним відношенням. Інші операції працюють з парами відношень, і тому їх називають бінарними операціями. В наведених нижче визначеннях R і S це два відношення, що визначені на атрибутах $A = (a_1,a_2,...a_N)$ і $B = (bi, b_2,..b_M)$ відповідно.

За допомогою цих операцій можна будувати більш складні операції, які отримали назву об'єднань, що в мові структурованих запитів SQL реалізуються за допомогою інструкції SELECT. Результатом операції над відношенням є представлення, яке не зберігається в базі даних, протягом певного часу є доступним для читання.

Перераховані вище реляційні операції окремо застосовуються рідко, однак їх комбінації дозволяють одержувати вибірки даних практично будь- якої складності.

4.5. Правила Кодда

Для визначення того, чи є база даних реляційною чи ні, Едгар Кодд розробив дванадцять правил. Якщо база даних задовольняє всі ці правила, то її з упевненістю можна назвати реляційною.

Перше правило Кодда - дані в відношеннях повинні мати явне представлення. Це правило свідчить про те, що в реляційній базі даних повинен виконуватись принцип інформаційної неподільності.

Друге правило Кодда - до даних повинен бути забезпечений гарантований доступ. У цьому разі в будь-якій реляційній базі даних користувач, знаючи ім'я відношення, значення первинного ключа та ім'я атрибута, повинен отримати доступ до необхідних даних.

Третє правило Кодда - в реляційній базі даних повинна бути реалізована обробка невизначених значень. Це положення говорить про те, що в реляційних базах даних для всіх типів даних повинна підтримуватись робота з невизначеними значеннями NULL.

Четверте правило Кодда - опис бази даних повинен здійснюватись у термінах реляційної моделі. Це правило висуває додаткову вимогу до опису бази даних, яка повинна зберігатись у вигляді окремих зв'язаних між собою

відношень, а доступ до його вмісту повинен здійснюватись за допомогою описаних вище реляційних операцій.

П'яте правило Кодда - в реляційній базі даних повинна бути забезпечена повнота підмножини мови. При цьому мова маніпулювання даними і мова визначення даних повинні бути необхідними і достатніми засобами роботи з даними.

Шосте правило Кодда - реляційна база даних повинна підтримувати оновлення представлень. У цьому разі створені за допомогою реляційних операцій представлення при зміні даних у вихідних відношеннях повинні також оновлюватися.

Сьоме правило Кодда - в реляційній базі даних для маніпулювання даними повинна використовуватись високорівнева мова. Синтаксис операцій, що застосовуються до відношення в цілому або до кортежів зокрема, повинен бути зрозумілим будь-якому користувачеві.

Восьме правило Кодда - в реляційній базі даних повинна бути забезпечена фізична незалежність даних. Виконання цього правила приводить базу даних у відповідність моделі ANSI/SPARC, в якій прикладні програми не повинні залежати від особливостей представлення даних на фізичному рівні.

Дев'яте правило Кодда - в реляційній базі даних повинна бути реалізована логічна незалежність даних. Це правило свідчить про те, що додаток реляційних баз даних не повинен залежати від логічних обмежень, що накладаються на відношення.

Десяте правило Кодда - в реляційній базі даних повинна бути забезпечена незалежність контролю цілісності. Дотримання цього правила призводить до необхідності зберігання засобів підтримки цілісності даних, таких як обмеження на первинні і зовнішні ключі, межі допустимого діапазону значень даних в окремих відносинах бази даних.

Одинадцяте правило Кодда - реляційна база даних повинна мати дистрибутивну незалежність даних. У цьому разі до бази даних висувається вимога можливості перенесення на інші платформи і здатність до поширення, в тому числі в локальних та глобальних мережах.

Дванадцяте правило Кодда - реляційна база даних повинна забезпечувати узгодження мовних рівнів. Це правило Кодда головним чином спрямовано на узгодження виконання низькорівневих операцій, таких як резервне копіювання й архівація даних, та високорівневих операцій, наприклад, вибірка даних, розділення прав користувачів і привілеїв доступу.

Правила Кодда виступають переважно теоретичними вимогами, що висуваються до побудови реляційних баз даних. На практиці жодна база даних повною мірою не реалізує реляційну модель. Розробники баз даних керуються принципом розумної достатності, розробляючи систему, яка не суперечить необхідним вимогам реляційних баз даних і відповідає достатнім вимогам замовника.

4.6. Нормалізація реляційних баз даних

Після статті, присвяченої реляційній алгебрі, Едгар Кодд у 1972 р. опублікував працю під назвою "Подальша нормалізація реляційної моделі баз даних", у якій ввів термін "нормалізація" і сформулював три нормальні форми.

Нормалізація - це метод організації реляційної бази даних з метою скорочення надлишковості.

При розробці і проектуванні будь-якої реляційної бази даних однією з перших розв'язується задача нормалізації її відношень, причому цей процес є ітераційним і полягає в послідовному переведенні відношень з першої нормальної форми в нормальні форми більш високого ступеня за певними правилами. Кожна наступна нормальна форма обмежує функціональну залежність відношень при збереженні властивостей попередніх нормальних форм.

Нормалізація дозволяє повною мірою реалізувати переваги реляцій- ної моделі, змушує розробника створювати більшу кількість відношень, рівномірніше розподіляючи в них інформацію, що приводить до зниження надлишковості і підвищення цілісності даних.

Усього виділяють п'ять нормальних форм.

4.6.1. Перша нормальна форма

Перше правило реляційної моделі полягає в тому, що жоден стовпчик не повинен містити два або більше значень.

Відношення представлено в першій нормальній формі (1НФ) тоді і тільки тоді, коли всі його атрибути є простими, тобто містять тільки неподільні порції даних.

Наведемо приклад перетворення відношення на першу нормальну форму, використовуючи таблицю, в якій зберігаються дані про персональні комп'ютери (рис. 4.13).

information
$600 500 10 $6000
$850 750 5 $4250
$700 500 3 $2100
$450 600 8 $3600
$670 600 10 $6700

1НФ →

price	speed	quantity	total
600	500	10	6000
850	750	5	4250
700	500	3	2100
450	600	8	3600
670	600	18	6700

Рис. 4.13. Перетворення відношення на першу нормальну форму

Як видно з рис. 4.11 усі дані про комп'ютери, що містяться у вихідному відношенні, розбиті на чотири прості атрибути, які містять неподільні порції даних.

Приведення відношень до першої нормальної форми спрощує розробку запитів і створення представлень, однак не виключає дублювання даних, що призводить до втрати цілісності і збільшення об'єму займаної пам'яті. Для вирішення цих проблем необхідно використовувати другу нормальну форму.

4.6.2. Друга нормальна форма

Для проведення подальшої нормалізації відношень необхідно використовувати поняття первинного ключа, оскільки друга нормальна форма вимагає, щоб усі атрибути залежали від первинного ключа.

Відношення представлено в другій нормальній формі (2НФ) тоді і тільки тоді, коли воно представлено в першій нормальній формі і кожний неключовий атрибут повністю визначається первинним ключем (рис. 4.14).

price	speed	quanti	total
600	500	10	6000
850	750	5	4250
700	500	3	2100
450	600	8	3600
670	600	18	6700

2НФ →

model	price	speed	quanti	total
1234	600	500	10	6000
1456	850	750	5	4250
1333	700	500	3	2100
1782	450	600	8	3600
1121	670	600	18	

Рис. 4.14. Перетворення відношення на другу нормальну форму

Надлишковість першої нормальної форми може слугувати причиною виникнення аномалій вставки і видалення. Аномалії вставки виникають тоді, коли дані неможливо додати в відношення доти, поки вони неповні, або для цього буде потрібний додатковий перегляд відношення. Наприклад, немає сенсу зберігати абстрактну інформацію про комп'ютери, якщо вона не відповідає певній моделі.

Застосування другої нормальної форми дозволяє усунути аномалії вставки і видалення.

4.6.3. Третя нормальна форма

Застосування першої і другої нормальних форм дозволяє позбутися явного дублювання даних, однак неявне дублювання даних все ж таки має місце. Вирішити цю проблему допомагає використання третьої нормальної форми (рис. 4.15).

Відношення представлено в третій нормальній формі (3НФ) тоді і тільки тоді, коли воно представлено в другій нормальній формі і всі неключові атрибути відношення взаємно незалежні та цілковито залежать від первинного ключа.

Якщо у відношенні є залежність атрибутів складного ключа від неключових атрибутів, то необхідно перейти до посиленої третьої нормальної форми, яка називається нормальна форма Бойса-Кодда.

model	price	speed	quanti	total
1234	600	500	10	6000
1456	850	750	5	4250
1333	700	500	3	2100
1782	450	600	8	3600
1121	670	600	18	6700

3НФ →

model	price	speed	quanti
1234	600	500	10
1456	850	750	5
1333	700	500	3
1782	450	600	8
1121	670	600	18

Рис. 4.15. Перетворення відношення на третю нормальну форму

Відношення представлено в нормальній формі Бойса-Кодда (БКНФ) тоді і тільки тоді, коли воно знаходиться в третій нормальній формі і в ньому відсутні залежності атрибутів складеного ключа від неключових атрибутів.

Приведення відношень у третю нормальну форму є достатньою умовою для розв'язку практичних задач і на цьому в більшості випадків процес проектування реляційної бази даних закінчується.

4.6.4. Четверта нормальна форма

В деяких випадках у відношеннях з'являється багатозначна залежність даних, яка полягає в залежності атрибутів складеного ключа. Видалення або додавання кортежем призводить до обробки всім кортежем відношення, що веде до надлишковості. Для виключення багатозначних залежностей необхідно використовувати четверту нормальну форму.

Відношення представлено в четвертій нормальній формі (4НФ) тоді і тільки тоді, коли воно знаходиться в нормальній формі Бойса- Кодда й існує багатозначна залежність первинного ключа від іншого атрибута, а решта атрибутів функціонально залежать від первинного ключа.

4.6.5. П'ята нормальна форма

Результатом нормалізації всіх попередніх операцій були два нові відношення. Іноді це виконати не вдається або отримувані відношення явно мають небажані властивості. В цьому разі виконують декомпозицію вихідного відношення на відношення, кількість яких перевищує два. Однак така декомпозиція не повинна приводити до втрати даних і появи додаткових фіктивних кортежів. Саме для цього і розроблена п'ята нормальна форма.

Відношення знаходиться в п 'ятій нормальній формі (5НФ) тоді і тільки тоді, коли воно представлено в четвертій нормальній формі і не містить залежностей з'єднання.

Треба відзначити, що глибоке розуміння процесів, що відбуваються при перетворенні на четверту і п'яту нормальні форми, вимагає залучення математичного апарата теорії множин. В цілому ж потрібно відзначити, що четверта і п'ята нормальні форми становлять більше теоретичний інтерес, ніж практичний.

4.7. Денормалізація баз даних

При проектуванні бази даних, яка задовольняє всі нормальні форми, з'являється надлишок відношень, оскільки перехід до чергової нормальної форми призводить до розділу відношень. При цьому усувається надлиш- ковість даних, однак з'являється надлишок внутрішніх зв'язків між відношеннями. Крім того, виникають додаткові витрати на виконання операцій об'єднання, що істотно впливає на продуктивність.

Усунення надлишковості не обов'язково означає підвищення продуктивності. Накладні витрати на виконання операцій об'єднання суттєві, тому розробники іноді свідомо йдуть на порушення правил нормалізації.

Процес порушення нормальних форм називається денормалізацією.

Денормалізація викликана декількома причинами, основними з яких є час виконання запитів, час проведення оновлень, загальний об'єм сховища даних, аномалії видалення, що викликають втрату цілісності даних.

4.8. Переваги та недоліки реляційного підходу у створенні баз даних

Реляціинии підхід є найбільш поширеним у даний час, хоча поряд із загальновизнаними перевагами має і багато недоліків. До переваг реляційного підходу можна віднести:

- наявність невеликого набору абстракцій, які дозволяють порівняно просто моделювати значну частину найбільш поширених предметних сфер і припускають точні формальні визначення, залишаючись при цьому інтуїтивно зрозумілими;
- наявність простого й одночасно потужного математичного апарату, що спирається головним чином на теорію множин і математичну логіку та забезпечує теоретичний базис реляційного підходу до організації баз даних;
- можливість ненавігаційного маніпулювання даними без необхідності знання конкретної фізичної організації баз даних у зовнішній пам'яті.

В даний час основним предметом критики реляційних СКБД є не їхня недостатня ефективність, а властива їм деяка обмеженість (прямий наслідок простоти) при використанні в галузях, у яких вимагаються гранично складні структури даних (наприклад, САПР). Ще одним часто відмічуваним недоліком реляційних баз даних є неможливість адекватного відображення семантики предметної сфери. Інакше кажучи, можливості представлення знань про семантичну специфіку предметної сфери в реляційних системах дуже обмежені. Сучасні дослідження у сфері постреляційних систем головним чином присвячені саме усуненню цих недоліків.

[1]Кристофер Дейт (Christopher J. Date) (1941 р.) - відомий фахівець у галузі баз даних, незалежний автор, лектор і консультант. Автор класичного посібника "Введення в системи баз даних", який як стандартний текст у системах баз даних використовується в багатьох університетах світу.

ЛЕКЦІЯ 5
ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ РОБОТИ БАЗ ДАНИХ

5.1. Паралельні обчислення

Сьогодні паралельні обчислення є надзвичайно популярними, наприклад як для інтернет-магазинів, так і для бухгалтерських багатокорис- тувацьких систем. Сутність роботи таких полягає в наступному. Певний вид товару надходить на склад підприємства, після чого він продається покупцеві, причому продаж не повинен відбутись раніше від надходження і один і той же товар не повинен бути проданим різним покупцям. Такий процес називається паралелізмом. Від

успішності реалізації паралельних обчислень залежить ефективність роботи системи в цілому.

Паралелізм є доволі складною проблемою для СКБД, які повинні справлятись не тільки з численними запитами на підключення, а й планувати одночасний доступ до даних. Для розв'язку таких задач використовуються транзакції.

5.1.1. Поняття транзакції

Транзакцією називається послідовність операцій, що здійснюються над базою даних і які переводять базу даних з одного несуперечливого стану в інший несуперечливий стан.

В кінці транзакції відбувається або її відміна, або завершення. Відміна транзакції називається відкатом, завершення транзакції називається фіксацією.

Транзакція повинна мати властивості, які представлені на рис. 5.1.

Рис. 5.1. Властивості транзакції

Атомарність означає дотримання принципу неподільності транзак- цій. Даний принцип полягає в тому, що всі інструкції, що складають транзакцію, обов'язково виконуються, інакше не виконується жодна з них. Будь-яких проміжних станів не існує.

Атрибут є атомарним, якщо його значення втрачає зміст при будь- якому розбитті на частини або перевпорядкуванні. І навпаки, якщо будь- який спосіб розбивки на частини не позбавляє атрибут сенсу, то атрибут є неатомарним. Одне й те ж значення може бути атомарним або неатомарним залежно від змісту цього значення.

Узгодженість гарантує, що у міру виконання транзакції дані переходять з одного узгодженого стану в інший, при цьому не порушується взаємна узгодженість даних у БД.

Ізоляція полягає в організації послідовної обробки транзакцій при доступі до бази даних, при цьому в користувача виникає враження, що всі транзакції виконуються паралельно.

Стійкість означає, що при успішному завершенні транзакції в БД вносяться постійні, невідмінювані зміни. БД, що підтримує виконання стійких транзакцій, здатна витримати раптову аварію, наприклад, збій живлення, і при цьому залишатися узгодженою.

Транзакції реалізуються шляхом ведення журналу всіх змін, що вносяться в базу даних у ході виконання кожної транзакції. За журналом можна відновити узгоджений стан БД у випадку збою.

Існує декілька видів збою, класифікацію яких наведено на рис. 5.2.

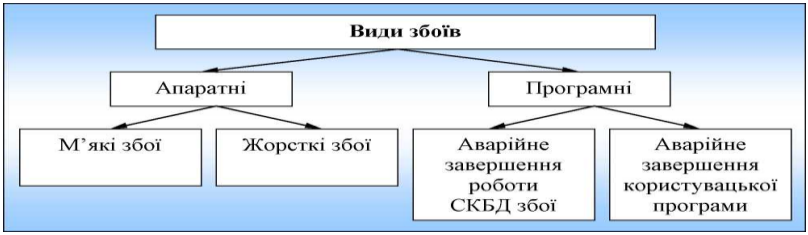


Рис. 5.2. Класифікація видів збою баз даних

Виділяють два види апаратних збоїв. М'які збої виникають при раптовій зупинці роботи комп'ютера, наприклад, при аварійному вимкненні живлення. Жорсткі збої проявляють себе при виході комп'ютера з ладу і спричиняють втрату інформації на носіях зовнішньої пам'яті.

Серед програмних видів збоїв треба відзначити аварійне завершення роботи СКБД і аварійне завершення користувацької програми, в результаті чого певна транзакція залишається незавершеною і потрібно ліквідувати її наслідки.

Підтримка надійності збереження даних є пріоритетним завданням роботи СКБД. Для цього в БД необхідно зберігати надлишкову інформацію для відновлення, причому ця інформація повинна зберігатись дуже надійно. Найбільш поширеним методом підтримки такої надлишковості є ведення журналу змін БД.

Журналом транзакцій називається механізм збереження проміжних станів, підтвердження або відкату транзакції, що реалізується в СКБД.

У багатьох випадках дотримуються стратегії "упереджувального" запису в журнал транзакцій (так званого протоколу Write Ahead Log - WAL), який полягає в тому, що запис про зміну будь-якого об'єкта БД повинен потрапити до зовнішньої пам'яті журналу раніше, ніж змінений об'єкт потрапить до зовнішньої пам'яті основної частини БД.

Треба відзначити, що використання транзакції при оптимізації БД знижує продуктивність роботи, однак підвищує цілісність БД. Відсутність транзакцій приводить до істотного зростання продуктивності, проте страждає цілісність. Компроміс полягає в тому, що існують проміжні варіанти, які називаються рівнями ізоляції транзакцій, що дозволяють знайти компроміс між цими двома суперечливими вимогами.

5.1.2. Рівні ізоляції

Прийнято виділяти чотири рівні виконання транзакцій: рівень серіалізації транзакцій (SERIALIZIBLE), рівень достовірного читання (REPEATABLE READ), рівень підтвердженого читання (READ COMMITTED), рівень непідтвердженого читання (READ UNCOMMITTED).

Найвищий рівень виконання транзакцій спрямований на збереження цілісності даних і розв'язок проблем, пов'язаних з появою проміжних даних, неузгоджених даних і рядків-примар, а найнижчий рівень націлений лише на підвищення швидкості виконання запитів.

Проміжні дані з’являються тоді, коли незавершена транзакція модифікує кортеж відношення, а інша транзакція зчитує цей кортеж. Якщо перша транзакція буде відмінена, то виявиться, що друга транзакція отримала дані, які ніколи не існували. Така ситуація з’являється на рівні непідтвердженого читання.

На рівні підтвердженого читання транзакції можуть читати тільки підтверджені дані. Однак це не вирішує проблему неузгоджених даних.

Наприклад, у ході транзакції виконується запит, який визначає середнє значення за певним числовим атрибутом відношення. По завершенні цього запиту інша транзакція, що працює з тим же відношенням, видаляє або додає кортежі. Якщо перша транзакція буде повторно виконувати свій запит, то вона отримає інші результати.

Проблема неузгоджених даних розв’язується на рівні достовірного читання. В цьому режимі кортежі, до яких транзакція звертається для читання або запису, блокуються. В даному разі виникає проблема, пов’язана з появою кортежів-примар.

Транзакція може заблокувати всі кортежі, з якими ведеться робота, проте вона не може завадити іншій транзакції додати кортеж в той же час відношення. Якщо в ході транзакції вводяться два запити на вибірку, а між ними друга транзакція додає в відношення новий кортеж, то цей кортеж стане "примарою", оскільки вона з’являється в ході однієї і тієї ж транзакції.

На рівні серіалізації транзакції примусово виконуються одна за одною в порядку пріоритету. Якщо дві транзакції будуть прагнути оновити один і той же кортеж, то одна з транзакцій буде оголошена такою, що програла в тупиковій ситуації і відмінена.

На будь-якому рівні нижче рівня серіалізації з’являється ризик порушення цілісності даних. Однак, якщо в рамках поставлених завдань потрібно збільшити швидкість виконання запитів, то цілісність даних відходить на другий план і можна застосовувати один з трьох інших рівнів виконання транзакцій.

5.1.3. Виконання транзакцій

Багато СКБД за замовчуванням працюють у режимі автоматичного завершення транзакцій. В цьому разі кожен запит є самостійною транзакцією, яка після отримання результатів запиту негайно завершується.

Однак для забезпечення одночасного доступу до даних використовуються механізми блокування БД. СКБД може блокувати, в разі потреби, відносини цілком, кортежі, атрибути або сторінки, які є довільним блоком даних, пов’язаних з відношенням.

Також для виконання транзакцій у деяких СКБД використовується механізм складання послідовності виконання транзакцій, що являє собою лічильник для створення унікальних числових ідентифікаторів. У цьому разі в БД жодні потоки виконання транзакцій не отримають однакові ідентифікатори, отже, не завадять роботі один одного.

Проектувальники БД можуть створювати послідовності і самі за допомогою первинних ключів з автоматичним збільшенням значення при додаванні кортежу.

5.2. Обробка транзакцій

Відслідковувати рівень виконання транзакції розробнику необхідно при проектуванні спеціалізованої БД, у більшості ж інших випадків доцільно використовувати вже готові і пророблені системи обробки транзакцій, наприклад, OLTP-системи, OLAP-системи або монітори транзакцій.

5.2.1. OLTP-системи

У разі використання в БД сильно нормалізованих моделей даних прийнято застосовувати так звані OLTP-додатки (On-Line Transaction Processing - оперативна обробка транзакцій).

Основною функцією таких систем є виконання більшої кількості коротких транзакцій, при цьому транзакції виглядають відносно просто, наприклад, операція читання або запису, і час очікування запитів не перевищує декілька секунд.

Однак OLTP-системи мають низку недоліків. У таких системах транзакцій дуже багато, виконуються вони одночасно, і в разі помилки транзакція повинна повність відкотитись.

Практично всі запити до БД в OLTP-додатках складаються з команд вставки, оновлення та видалення. Запити на вибірку головним чином призначені для надання користувачам можливості вибору даних з різних відношень. При цьому більша частина запитів відома заздалегідь ще на етапі проектування системи. Таким чином, критичними для OLTP- додатків є швидкість і надійність виконання коротких операцій оновлення даних. OLTP-системи вимагають захисту від несанкціонованого доступу, порушення цілісності, апаратних і програмних збоїв.

5.2.2. OLAP-системи

В БД, що використовують принципи побудови систем підтримки прийняття рішень, сховищ даних, систем інтелектуального аналізу даних, застосовуються OLAP-додатки (On-Line Analitical Processing - оперативна аналітична обробка даних).

Такі системи характеризуються такими ознаками. Додавання даних відбувається рідко і головним чином великими блоками. Дані, додані в систему, зазвичай ніколи не видаляються. В таких системах перед завантаженням дані проходять різні процедури обробки. Запити до бази даних є достатньо складними. При цьому швидкість виконання запитів є важливою, але не критичною.

Дані OLAP-додатків зазвичай представлені у вигляді одного або декількох гіперкубів і використовують багатомірні моделі даних. Оперативність обробки

великих об’ємів даних в OLAP досягається за рахунок застосування потужної багатопроцесорної обчислювальної техніки, складних методів аналізу і спеціальних сховищ даних, що накопичують інформацію з різних джерел за великий період часу та забезпечують швидкий доступ до неї. В більшості випадків складний аналітичний запит неможливо сформулювати в термінах стандартної мови запитів SQL, і доводиться застосовувати спеціалізовані мови, орієнтовані на аналітичну обробку даних.

5.2.3. Монітори транзакцій

Зі зростанням складності розподілених обчислювальних систем виникають проблеми ефективного використання їх ресурсів. Для розв’язку цих проблем до складу розподілених OLTP-систем вводять додатковий компонент - монітор транзакцій.

Монітори транзакцій виконують дві основні функції в СКБД: динамічний розподіл запитів у системі й оптимізацію кількості виконуваних серверних додатків. Перша функція спрямована на оптимальний розподіл обчислювальних ресурсів між декількома серверами при обробці великої кількості запитів, друга функція спрямована на розв’язок завдання запуску необхідної і достатньої кількості серверних додатків для обробки запитів із забезпеченням максимальної швидкості обробки транзакцій.

Якщо в системі використовується монітор транзакцій, то з боку розробника не потрібно додаткових витрат для підтримки контролю цілісності даних, і він може зосередитись на розв’язку інших завдань.

5.3. Оптимізація баз даних

Оптимізація БД - це процес налаштування системи, спрямований на підвищення швидкості її роботи або скорочення об’єму використовуваної пам’яті. У більшості випадків успіх при оптимізації баз даних досягається тоді, коли накопичено достатній досвід проектування і супроводу таких систем. Врешті-решт оптимізація спрямована на економію грошових коштів, пов’язаних з експлуатацією і супроводом БД, отже, фінансові витрати на оптимізацію не повинні перевищувати економічної ефективності від впровадження засобів оптимізації.

Оптимізація починається з організаційних заходів, спрямованих на оновлення апаратної частини, оновлення операційної системи та оновлення безпосередньо СКБД (рис. 5.3).

Рис. 5.3. Організаційні заходи при оптимізації бази даних

Найбільший ефект приносить оновлення апаратної частини, при цьому саме цей організаційний захід може виявитися найдешевшим варіантом.

Гордон Мур встановив, що обчислювальні потужності здвоюються кожні півтора роки. Незважаючи на таке стрімке зростання продуктивності, питома вартість обчислювальних засобів невпинно знижується, що й гарантує досягнення необхідного результату.

Наступним етапом організаційних заходів, спрямованих на оптимі- зацію бази даних, є оновлення операційної системи. Залежно від цілей і завдань, які ставляться перед БД, можна використовувати або безплатні облегшені операційні системи, або комерційні операційні системи.

Оновленню підлягає і сама СКБД. Постійно випускаються нові версії програмних продуктів такого типу, які розв'язують або нові завдання, або старі, на новому рівні. При цьому в нових версіях СКБД зазвичай виправлено похибки, які також можуть істотно впливати на ефективність роботи.

Однак, оскільки не завжди вдається досягти потрібного ефекту за рахунок організаційних заходів, іноді доводиться застосовувати безпосередньо методи оптимізації БД. Класифікацію методів оптимізації баз даних наведено на рис. 5.4.

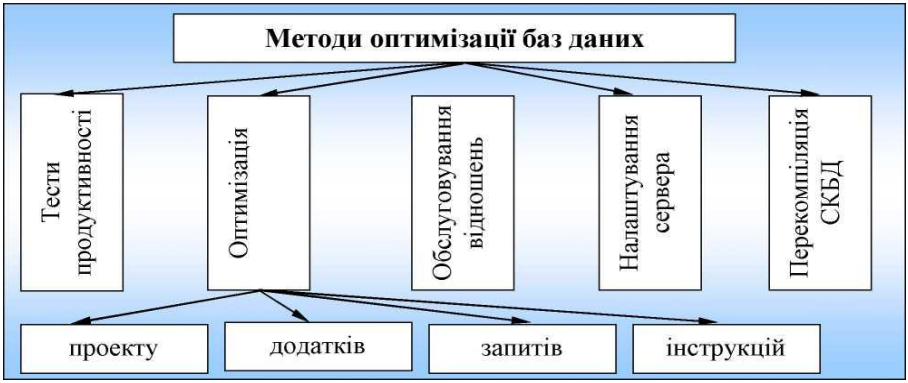


Рис. 5.4. Методи оптимізації баз даних

Попереднім етапом оптимізації БД є оцінка продуктивності. Практично всі розробники СКБД надають комплекс тестів продуктивності, що включають в себе перевірку декількох стандартних аспектів функціонування БД, наприклад, швидкості виконання запитів, операцій зчитування і запису, а також поради з усунення недоліків, виявлених у процесі тестування системи.

Такі тести відбивають тільки відносну продуктивність сервера. За їх допомогою можна дізнатись, наскільки зросте швидкість його роботи при зміні рекомендованих налаштувань, вони лише непрямо допомагають в оптимізації БД.

Оптимізація БД починається з оптимізації проекту. Основною метою даного методу є визначення прийнятного рівня надлишковості даних і денормалізація деяких відношень.

Для цього процесу характерним є створення підсумкових даних. Наприклад, можна додати до відношення атрибут, що зберігає результати обчислень по інших атрибутах, іноді застосовують технологію створення не тільки підсумкових атрибутів, а й цілих відношень, причому, якщо такі відношення зберігають часто змінювану інформацію, можна зробити їх резидентними.

Також особливу увагу при оптимізації проекту приділяють визначенню типів даних. Наприклад, іноді доцільно не використовувати тип даних змінної довжини, що підтримується багатьма сучасними СКБД. Як правило, в них зберігаються відеофільми, звукозаписи, зображення. З точки зору оптимізації проекту більш ефективним є запам'ятовування лише шляху до таких файлів, при окремому збереженні їх від БД.

Оптимізація додатків, що працюють з БД, головним чином полягає в використанні технології кешування. Для швидкого підключення до БД можна кешувати з'єднання з БД, можна кешувати дані з відношень, що використовуються найбільш часто, або перед виконанням вставки кортежем у відношення, щоб потім здійснити виконання цієї операції в пакетному режимі.

У разі використання мови запитів SQL оптимізувати запит доволі складно, хоча різноманітні підходи до реалізації одного й того ж завдання можуть заощадити процесорний час. При проектуванні запитів необхідно особливу увагу приділяти роботі з індексами, оскільки вони дозволяють істотно підвищити швидкість роботи додатка. Якщо подібний індекс ще не існував, то за потреби його слід створити, оскільки в багатьох проектах пріоритетним завданням є збільшення швидкості роботи додатка, а не економія дискового простору.

Крім оптимізації запитів, можна застосовувати методи оптимізації інших інструкцій, наприклад, інструкцій вставки і видалення. Щоб уникнути витрат часу на аналіз інструкції, можна скористатися перевагами значень за замовчуванням: замість того щоб вказувати значення всіх атрибутів, можна задати лише ті, які відрізняються від стандартних установок. Для того щоб не дублювати синтаксис однотипних операцій, необхідно використовувати багаторядкові інструкції. Також можна скористатися командами, що дозволяють поміщати інструкції в чергу без блокування клієнтського додатка, що підвищує його оперативність.

Періодично для прискорення виконання запитів необхідно здійснювати оптимізацію відношень. По-перше, можна видаляти порожні проміжки, що залишилися після видалення кортежів. По-друге, можна з'єднувати розподілені фрагменти відношень з динамічними кортежами. По-третє, можна сортувати відношення й індекси. У разі якщо відношення протягом часу практично не змінюється і дисковий простір обмежений, можна архівувати відношення, однак після цього воно буде доступним лише для зчитування.

Розподіл ресурсів сервера, а головним чином оперативної пам'яті, також впливає на продуктивність баз даних. Збільшення об'єму пам'яті сприяє прискоренню роботи СКБД, оскільки в оперативній пам'яті вона зберігає представлення і часові відношення. Існує безліч зразків конфігураційних файлів з різними варіантами налаштувань, що стосуються використання пам'яті сервера, практично для всіх баз даних і операційних систем. Підбір найбільш придатних конфігурацій сервера в сукупності з тестуванням системи може принести позитивні результати при проведенні оптимізації баз даних.

Необхідність у компіляції при оптимізації баз даних виникає в тому разі, коли для використовуваної платформи неможливо знайти скомпільовану версію СКБД або використовувані бібліотеки чи компілятори морально застаріли. Інструкцій з перекомпіляції на інформаційних ресурсах розробників БД доволі багато, однак цей метод оптимізації є крайнім і застосовується в тому разі, коли всі інші варіанти оптимізації не принесли прийнятного результату.

5.4. Організація безпеки баз даних

Для організації безпеки БД недостатньо організувати охорону приміщень і встановити програмні засоби захисту. Потрібен комплексний підхід, який передбачає організацію захисту СКБД відповідно до її структури, завдань і можливостей захисту.

Систему безпеки БД потрібно планувати з початку її розробки і виконувати на всіх етапах життєвого циклу.

На практиці створення системи захисту часто починається в процесі експлуатації БД. Система безпеки повинна бути багаторівневою, адаптованою до нових умов функціонування.

Для ефективної побудови системи безпеки необхідно, по-перше, виділити уразливі елементи з СКБД, по-друге, виявити загрози для виділених елементів, по-третє, сформувати вимоги до системи безпеки і, нарешті, вибрати методи та засоби задоволення висунутим вимогам.

Безпека СКБД може бути порушена внаслідок реалізації загроз, які являють собою можливість навмисної або випадкової дії, що може призвести до порушення безпеки інформації, яка зберігається та оброблюється.

Основними видами загроз у СКБД є несанкціоноване або некоректне використання ресурсів, поява похибок у програмних і апаратних засобах, викрадення носіїв інформації, виведення з ладу засобів збереження або передачі даних.

Існуючі методи забезпечення безпеки БД можна розділити на чотири основні класи: фізичні, апаратні, програмні й організаційні (рис. 5.5).

Рис. 5.5. Методи забезпечення безпеки баз даних

Фізичний захист спрямований на обмеження доступу сторонніх осіб у приміщення, де розташований сервер з встановленою СКБД. Фізичні методи забезпечення безпеки включають засоби охоронної сигналізації, що засновані на різних принципах.

Основними апаратними засобами забезпечення безпеки є засоби захисту процесорів, наприклад, перевірка припустимості команд, що видаються, пристроїв зовнішньої пам'яті, каналів зв'язку і систем електроживлення.

До програмних методів забезпечення безпеки БД відносять застосування вбудованих засобів операційних систем і антивірусних програм, використання спеціалізованих комерційних додатків і утиліт.

Організаційний захист передбачає сукупність дій, пов'язаних з розробкою і прийняттям законодавчих документів, що стосуються питань захисту інформації, а також з проведенням пропагандистських заходів щодо роботи з БД в межах правового поля.

Однак найбільший інтерес становить синтез двох підходів, які реалізують програмно-апаратні методи забезпечення захисту інформації в БД. Програмно-апаратні засоби забезпечення безпеки допомагають як захистити СКБД від зовнішніх впливів, так і запобігти виникненню похибок у самих додатках, що працюють з БД.

5.5. Захист баз даних від несанкціонованого доступу

Для захисту від несанкціонованого доступу за допомогою програмно-апаратних засобів використовують технологію реєстрації спроб доступу в систему з боку користувачів і програм, а також гнучку політику швидких повідомлень про такі спроби.

Захист від несанкціонованого доступу з боку користувачів у сучасних системах реалізується двома способами: парольним захистом і шляхом ідентифікації й автентифікації[1].

Найпростіший парольний захист є доволі слабким засобом забезпечення безпеки БД, особливо якщо пароль не шифрується. Основний її недолік полягає в тому, що всі користувачі, що використовують однаковий пароль, з точки зору СКБД, ідентичні. Незручність парольного захисту для користувача полягає в тому, що при використанні простого і короткого пароля з'являється загроза його підбору зловмисниками, а якщо пароль складний, то користувач повинен додатково його зберігати на іншому носієві і відповідно забезпечувати йому додатковий захист.

Більш серйозний контроль доступу в систему відбувається, якщо кожного користувача, що підключається до БД, спочатку ідентифікувати, а потім автентифікувати - переконатися, що це саме він, і при запиті ресурсів контролювати його повноваження.

Для автентифікації можна використовувати особисту інформацію користувача, електронні ключі, електронні жетони, магнітні картки, активні засоби розпізнавання або біометричні засоби. Найбільш перспективними є

біометричні засоби захисту баз даних. До них відносяться контроль за відбитками пальців, зчитування сітчатки ока, розпізнавання голосу, визначення ДНК людини. Однак багато з цих методів проходять етапи доопрацювання та апробації.

Одним із різновидів несанкціонованих програм є комп'ютерні віруси. Кількість комп ютерних вірусів постійно зростає, отже, і потенційний збиток від їх роботи також пропорційно зростає. Тому проблема захисту БД від комп'ютерних вірусів на всіх стадіях їх розвитку є надзвичайно актуальною. Для цього в СКБД розробники включають засоби діагностування стану програмно-апаратних засобів, локалізації і видалення вірусів, усунення наслідків їх дії.

5.6. Захист баз даних від несанкціонованого використання ресурсів

Забезпечення захисту від несанкціонованого використання ресурсів вимагає застосування засобів моніторингу запитів ресурсів, що підлягають захисту, і сигналізації при спробі їх незаконного використання. До таких засобів відносяться варіанти захисту від копіювання даних, дослідження додатків, перегляду, модифікації і видалення даних.

Рис. 4.6. Приклади несанкціонованого використання ресурсів

Для захисту СКБД від несанкціонованого копіювання в додатках можна використовувати прив'язку до апаратної частини, щоб створена копія була непрацездатною на іншому комп'ютері.

При несанкціонованому дослідженні додатків застосовуються такі засоби, які не дозволяють або ускладнюють вивчення системи захисту БД. Наприклад, після декількох невдалих спроб підключення до додатка, який має парольний захист, необхідно блокувати подальші спроби підключення до неї або передбачити засоби самоліквідації.

Найбільш ефективним засобом захисту даних від перегляду є їх шифрування. Несанкціонований перегляд даних при цьому потребує використання ключа шифрування, підбір якого навіть при сучасному рівні розвитку інформаційних технологій становить складне завдання.

Шифрування незамінне для захисту інформації від розкриття її змісту в СКБД, а також при її передачі лініями зв'язку. Шифрування даних здійснюється в темпі надходження інформації і в автономному режимі. Перший спосіб

застосовується в системах прийому-передачі інформації, а другий - для засекречування інформації.

Поширеним підходом до розв'язку задачі захисту даних від модифікації є обчислення контрольних сум і їх порівняння з еталоном.

Захистити дані від видалення можна шляхом запобігання несанкціонованим операціям видалення в СКБД. Для цього цілей необхідно розробляти додаткові програмні засоби, які паралельно з виконанням користувацького додатку відслідковує запити на видалення, що надходять, і в разі їх несанкціонованого виконання відмовляють у доступі до даних.

5.7. Захист баз даних від некоректного використання ресурсів

Захист від некоректного використання ресурсів виконується операційною системою і передбачає виділення клієнтським додаткам ізольованої частини оперативної пам'яті та захист системних ділянок зовнішньої пам'яті. Однак більшість розробників ідуть далі і впроваджують різноманітні інструменти коректного використання прикладних ресурсів, наприклад, документів, зображень, звукових файлів.

5.8. Захист баз даних за допомогою внесення надлишковості

Дуже ефективним способом відновлення БД після збоїв є надмірність. Надмірність буває функціональною, інформаційною або структурною. Залежно від цілей захисту БД можна використовувати одну з представлених видів надмірності або їх композицію.

Функціональна надмірність означає організацію обчислювального процесу, при якому дублюються функції управління, зберігання й обробки інформації.

Наприклад, запуск декількох однакових клієнтських додатків у багатозадачній операційній системі є ознакою функціональної надмірності.

Інформаційна надмірність використовується для запобігання повній втраті інформації і реалізується шляхом одноразового або періодичного копіювання та архівування найбільш цінної інформації. У разі збою або втрати інформації можна, використовуючи резервну копію, відновити необхідну інформацію, важливим аспектом в даному разі є періодичність. Якщо резервування проводиться часто, то може постати проблема неефективного використання дискового простору, якщо резервування проводиться рідко, то можуть зрости витрати на відновлення інформації, яка накопичилася в БД між моментом останнього збереження і моментом збою.

Структурна надмірність означає резервування апаратних компонентів обчислювальної системи на різних рівнях: дублювання серверів обробки інформації, дублювання накопичувачів інформації.

При резервуванні необхідно в першу чергу забезпечити стабільне та безперебійне живлення обчислювальної системи.

[1]Автентифікація (з грец. - реальний або істинний) - процедура встановлення справжності користувача.

ЛЕКЦІЯ 6
МОВА СТРУКТУРОВАНИХ ЗАПИТІВ SQL

Мова маніпулювання даними (ММД) дозволяє виконувати передбачені в системі операції над даними з бази даних, тобто містить набір операторів маніпулювання даними, що дозволяє вибирати, заносити дані, видаляти, модифікувати (редагувати) тощо.

В даний час існують численні приклади мов СКБД, що поєднують можливості опису даних і маніпулювання даними в єдиних синтаксичних рамках. Більш того, сучасні СКБД підтримують єдину інтегровану мову, що містить усі необхідні засоби для роботи з базою даних. Це мова SQL (Structured Query Language - структурована мова запитів) і QBE (Query- By-Example - запити за зразком).

6.1. Загальні відомості про структуровану мову запитів

Зростання кількості даних, необхідність їх збереження та обробки призвели до того, що виникла потреба створення мови маніпулювання даними (ММД), яка б могла функціонувати в численних комп'ютерних системах різних видів.

Будь-яка ММД повинна надавати користувачу можливості:

- створювати БД і таблиці з повним описом їх структури;
- виконати основні операції маніпулювання даними, зокрема, вставку, модифікацію і видалення даних з таблиць;
- виконувати прості і складні запити, які здійснюють перетворення даних.

Усі ММД, що були створені до появи реляційних баз даних і розроблені для багатьох СКБД, орієнтувались на операції з даними, які були представлені у вигляді логічних записів файлів. Це вимагало від користувачів ретельного знання організації збереження даних і певних зусиль для вказівки не тільки того, які дані потрібні, але й того, де вони розміщені і як крок за кроком отримати їх. Тому актуальною стала проблема створення універсальної ММД, за допомогою якої користувачі могли б маніпулювати даними незалежно від того, де вони працюють (на персональному комп'ютері, мережевій робочій станції або універсальній ЕОМ.

Саме такою мовою, яка з'явилась у результаті розробки реляційної моделі даних, стала мова SQL (Structured Query Language - Мова струк- турованих запитів), яка на даний час отримала широке поширення і фактично перетворилася на стандартну мову реляційних баз даних.

SQL веде свою історію з початку 1970-х рр., коли в дослідній лабораторії компанії IBM у штаті Каліфорнія була розроблена його перша версія. Перша публікація опису мови відноситься до 1974 р. На той час її назвали SEQUEL (Structured English Query Language - структурована англійська мова запитів). Спочатку вона була реалізована в експериментальній реляційній СКБД System/R, проект якої ініціювала в середині 70-х рр. XX ст. компанія IBM у дослідній лабораторії в Сан-Хосе. При реалізації наступної версії System/R мова

була перейменована в SQL. Дослідний проект System/R був завершений у 1979 р. і підтвердив можливість створення ефективних промислових реляційних СКБД.

У 1977 р. невеличка, щойно створена фірма Relational Software розпочала до створення промислової реляційної СКБД на основі SQL. Постачання цієї СКБД, яка отримала назву Oracle, розпочалось у 1979 р. Незабаром і сама фірма була перейменована в Oracle. З того часу вона є найбільшим постачальником реляційних СКБД на базі SQL.

У середині 70-х рр. XX ст. в дослідній лабораторії Каліфорнійського університету в Берклі був відкритий проект зі створення реляцій- ної СКБД. У цій лабораторії, як і в компанії IBM, створили експериментальну СКБД, що отримала назву Ingress, на якій відпрацьовувались результати наукових досліджень в галузі реляційних баз даних.

У 1980 р. частина співробітників цієї лабораторії організували фірму Relational Technology, яка в 1981 р. випустила промислову СКБД Ingress. Первісно ця СКБД використовувала реляційну мову QUEL, однак у 1986 р. була переведена на мову SQL.

У 1980 р. компанія IBM на основі досвіду, одержаного при розробці експериментальної System/R, розпочала до створення власної промислової СКБД реляційного типу, яка почала постачатись у 1982 р. під назвою SQL/DS. Згодом в компанії був розроблений більш досконалий продукт - DB2, постачання якого розпочалось у 1985 р. Ця СКБД стала стратегічним програмним продуктом компанії IBM.

Таким чином, до середини 80-х рр. XX ст. SQL стала загальновизнаною мовою реляційних СКБД, а її діалект, що підтримувався СКБД DB2, фактично став стандартом для керування реляційними БД.

Стандарт на мову SQL був випущений Американським національним інститутом стандартів (ANSI) у 1986 р., а в 1987 р. Міжнародна організація стандартів (ISO) прийняла його як міжнародний. Нинішній стандарт SQL відомий під назвою SQL/92. На сучасному етапі ведеться робота над більш сучасним стандартом SQL3.

Мова SQL (Structured Query Language - структурована мова запитів) орієнтована на операції з даними, представленими у вигляді логічно взаємопов’язаних сукупностей таблиць.

Особливість пропозицій цієї мови полягає в тому, що вони орієнтовані здебільшого на кінцевий результат обробки даних, а не на процедуру цієї обробки. SQL сама визначає, де знаходяться дані, які індекси і навіть найефективніші послідовності операцій треба використати для їх одержання, при цьому не потрібно вказувати ці деталі в запиті до БД.

SQL - мова програмування четвертого покоління, яка максимально наближена до людської мови.

Одним із головних завдань SQL є легкість синтаксису мови, вона зрозуміла всім, починаючи від користувачів і закінчуючи адміністраторами, а її запити читаються як звичайні речення.

Словник SQL відносно невеликий, а його команди є слова англійською мовою. Зазвичай ключові слова SQL записують прописними літерами, щоб відрізнити їх від назв таблиць і стовпчик. Часто для зручності сприйняття інструкції SQL записують у декількох рядках, що допускається синтаксичним аналізатором.

Для зручності подальших міркувань при опису реляційної моделі даних будемо використовувати табличний варіант термінів: відношення - таблиця, кортеж - рядок, атрибут - стовпчик, оскільки саме така термінологія використовуються при програмуванні баз даних.

Реалізація в SQL концепції операцій, орієнтованих на табличне подання даних, дозволила створити компактну мову з невеликим набором пропозицій. Мова SQL може використовуватися як для виконання запитів до даних, так і для побудови прикладних програм.

6.2. Категорії команд SQL

Основні категорії команд мови SQL призначені для виконання різноманітних функцій, включаючи побудову об'єктів бази даних і маніпулювання ними, початкове завантаження даних у таблиці, оновлення і видалення існуючої інформації, виконання запитів до бази даних, управління доступом до неї і її загальне адміністрування.

Серед категорій команд мови SQL можна виділити:

- мову визначення структур БД;
- мову маніпулювання даними;
- мову запитів;
- мову керування даними;
- команди адміністрування даних;
- команди керування транзакціями.

Мова визначення структур бази даних (Data Definition Language DDL) дозволяє створити і змінювати структуру об'єктів бази даних, наприклад, створювати і видаляти таблиці. Основними командами мови DDL є: CREATE TABLE, ALTER TABLE, DROP TABLE, CREATE INDEX, ALTER INDEX, DROP INDEX.

Мова маніпулювання даними (Data Manipulation Language — DML) використовується для маніпулювання інформацією всередині об'єктів реляційної бази даних за допомогою трьох основних команд: INSERT, UPDATE, DELETE.

Мова запитів (Doctrine Query Language - DQL) є найбільш відомою користувачам реляційних баз даних, незважаючи на те, що вона включає тільки одну команду SELECT. Ця команда разом зі своїми численними опціями і

пропозиціями використовується для формування запитів до реляційної бази даних.

Мова керування даними (Data Control Language - DCL) дозволяє керувати доступом до інформації, що знаходиться всередині БД. Зазвичай, вона використовується для створення об'єктів, пов'язаних з доступом до даних, а також слугує для контролю над розподілом пріоритетів між користувачами. Команди керування даними - GRANT, REVOKE.

Команди адміністрування даних дозволяють користувачу здійснювати контроль за виконанням дій та аналізувати операції БД; вони також можуть виявитися корисними при аналізі продуктивності системи.

Примітка. Не треба плутати адміністрування даних з адмініструванням, БД, яке є загальним керуванням БД і має на меті використання команд усіх рівнів.

Команди керування транзакціями - COMMIT, ROLLBACK, SAVEPOINT, SET TRANSACTION.

6.3. Переваги мови SQL

Мова SQL є основою багатьох СКБД, оскільки відповідає за фізичне структурування і запис даних на диск, а також за зчитування даних з диска, дозволяє приймати SQL-запити від інших компонентів СКБД і додатків користувачів.

SQL - потужний інструмент, який забезпечує користувачів, програми і обчислювальні системи доступом до інформації, що міститься в реляційних БД.

Основні переваги мови SQL полягають у наступному:

- стандартизованість - використання мови SQL у програмах стандартизовано міжнародними організаціями;
- незалежність від конкретних СКБД - усі поширені СКБД використовують SQL, оскільки реляційну базу даних можна перенести з однієї СКБД на іншу з мінімальними доробками;
- можливість переносу з однієї обчислювальної системи на іншу - СКБД може бути орієнтована на різні обчислювальні системи, однак додатки, створені за допомогою SQL, припускають використання як для локальних БД, так і для великих багатокористувацьких систем;
- реляційна основа мови SQL є мовою реляційних БД, тому вона набула популярності тоді, коли одержала широке поширення реляційна модель подання даних. Таблична структура реляційної БД добре зрозуміла, а мова SQL проста для вивчення;
- можливість створення інтерактивних запитів - SQL забезпечує користувачам практично миттєвий доступ до даних, при цьому в інтерактивному режимі можна одержати результат запиту за дуже короткий час без напису складної програми;
- можливість програмного доступу до БД - мова SQL легко використовується в додатках, яким необхідно звертатися до БД. Одні й ті ж оператори SQL використовуються як для інтерактивного, так і програмного

доступу, тому частини програм, що містять звернення до БД, можна спочатку перевірити в інтерактивному режимі, а потім убудовувати в програму;

- забезпечення різного представлення даних - за допомогою SQL можна представити таку структуру даних, що той або інший користувач буде бачити різні їх представлення. Крім того, дані з різних частин БД можуть бути скомбіновані і представлені у вигляді однієї простої таблиці, а значить, представлення придатні для посилення захисту БД і її настроювання під конкретні вимоги окремих користувачів;
- можливість динамічної зміни і розширення структури БД - мова SQL дозволяє маніпулювати структурою БД, тим самим забезпечуючи гнучкість з точки зору пристосованості БД до вимог предметної сфери, що змінюються;
- підтримка архітектури клієнт-сервер. SQL - один з найкращих засобів для реалізації додатків на платформі клієнт-сервер. SQL слугує сполучною ланкою між клієнтською системою, що взаємодіє з користувачем, і серверною системою, що керує БД, дозволяючи кожній з них зосереджуватись на виконанні своїх функцій.

Мова SQL використовується в інших стандартах і навіть здійснює вплив на розробку інших стандартів як інструмент визначення (наприклад, стандарт Remote Data Access, RDA).

Створення мови сприяло не тільки розробці необхідних теоретичних основ, а й підготовці успішно реалізованих технічних рішень. Це безпосередньо стосується оптимізації запитів, методів розподілу даних і реалізації засобів захисту. Почали з'являтися спеціалізовані реалізації мови, призначені для нових ринків: системи керування обробки транзакцій (OnLine Transaction Processing, OLTP) і системи оперативної аналітичної обробки або системи підтримки прийняття рішень (On Line Analytical Processing, OLAP). Вже відомі плани подальших розширень стандарту, які включають підтримку розподіленої обробки, об'єктно орієнтованого програмування, розширень користувачів і мультимедіа.

6.4. Базові поняття реляційних баз даних

Основними поняттями реляційної БД (РБД) є: *відношення*, *домен, кор-теж, тип даних, атрибут, схема відношення, первинний ключ* (рис. 6.1).

Рис. 6.1. Схема реляційної бази даних

Відношення – фундаментальне поняття реляційної моделі даних. Звичайним представленням відношень є таблиця, заголовком якої є схема відношень, а рядками – кортежі відношення-екземпляра (рис. 6.2)

Рис.6.2. Представлення відношень

Імена атрибутів іменують стовпчики цієї таблиці. Тому іноді кажуть "стовпчик таблиці", маючи на увазі атрибут відношення.

Відношенням R називають підмножину декартового добутку D1 × D2 ×...× Dn множин D1, D2 ,..., Dn(n ≥ 1) не обов'язково різних. Вихідні множини n D1, D2...,D називаються доменами. Найбільш правильним трактуванням поняття домену є розуміння його як припустимого потенційної безлічі значень даного типу.

Домен – обмежена підмножина значень даного типу.

Домен має таке семантичне навантаження: дані вважаються порівнюваними тільки у тому разі, коли вони відносяться до одного домену. Кортеж – це

множина пар {ім'я атрибута, значення}, яка містить одне входження кожного імені атрибута, що належить схемі відношення.

Кортеж – набір іменованих значень даного типу.

Типи даних – у сучасних реляційних БД припускається збереження символьних, числових даних, бітових рядків, спеціалізованих числових даних, спеціальних темпоральних і абстрактних даних. Атрибут – це входження домену у відношення.

Будь-який об'єкт реального світу характеризується певною множиною характеристик (атрибутів (A1, A2, ..., An)). Ця характеристика має ім'я атрибута (A1, A2, ..., An) і множину допустимих значень – доменів. Тоді таблиця являє собою відношення, в якому кожний рядок є множиною значень, взятих по одному з домену кожного імені атрибута.

Рядки відношень називаються кортежами і мають кратність, яка дорівнює кількості атрибутів.

Кортежі відношень утворюють множину, оскільки рядки не дублюються.

Схемою відношення SR називається скінченна множина імен атрибутів (A1,A2,...,An):

SR = (A1, A2,..., An), $Ai \subseteq Dj$.

Якщо атрибути приймають значення з одного і того ж домену, то вони називаються θ - порівнюваними, де θ – множина припустимих операцій порівняння, заданих для даного домену. Наприклад, якщо домен містить числові дані, то для нього припустимими є всі операції порівняння. Отже, можна записати:

θ = {=, <, >, > =, = <, < >}.

Однак і для доменів, що містять символьні дані, можуть бути задані не тільки операції порівняння за рівністю і нерівністю значень.

Якщо для даного домену задано графічну упорядкованість, то він має також повний спектр операцій порівнянь. Схеми двох відношень називаються еквівалентними, якщо вони мають однакову степінь і таку впорядкованість імен атрибутів у схемах, що на однакових місцях будуть знаходитись порівнювані атрибути, тобто атрибути, що приймають значення з одного домену.

SR1 = (A1, A2,..., An) – схема відношення R1,

SR2 = (B1, B2,..., Bn)– схема відношення R2 після впорядкування.

У реляційній моделі, як і в теоретико-графовій, підтримуються ієрархічні зв'язки між відношеннями. В кожному зв'язку одне відношення може виступати як основне, а інше виступає в ролі підпорядкованого. Це означає, що один кортеж основного відношення може бути пов'язаний з декількома кортежами підпорядкованого відношення. Для підтримки цих зв'язків, обидва відношення повинні містити набори атрибутів, за якими вони пов'язуються. В основному відношенні це первинний ключ відношень, який являє собою набір атрибутів, значення яких однозначно визначають кортеж основного відношення. Для кожного відношення принаймні повний набір його атрибутів має цю властивість.

Однак при формальному визначенні первинного ключа вимагається забезпечення його мінімальності, тобто в набір атрибутів первинного ключа не повинні входити такі атрибути, які можна відкинути без завдання шкоди для основної властивості – однозначно визначати кортеж.

У підпорядкованому відношенні для моделювання зв'язку повинен бути присутній набір атрибутів, який відповідає первинному ключу основного відношення, однак тут цей набір атрибутів уже є вторинним ключем, тобто він визначає множину кортежів підпорядкованого відношення, які пов'язані тільки з одним кортежем основного відношення.

Даний набір атрибутів у підпорядкованому відношенні називається зовнішнім ключем

6.5. Фундаментальні властивості відношень

1. Відсутність кортежів-дублікатів. Випливає з визначення відношень як множини кортежів.

2. Відсутність упорядкованості кортежів. Дає додаткову гнучкість при збереженні БД у зовнішній пам'яті і при виконанні запитів до БД. 3. Відсутність упорядкованості атрибутів.

4. Атомарність значень атрибутів. Серед значень домену не може міститись множина значень.

Основні правила відношень:

1. Кожне з відношень складається з кортежів і має унікальне ім'я.

2. Рядки мають фіксовану кількість полів та значень, тобто в кожній позиції відношення на перетинанні кортежу й атрибута завжди є одне значення або нічого немає.

3. Кортежі обов'язково відрізняються один від одного хоча б одним значенням, що дозволяє однозначно ідентифікувати будь-який кортеж такого відношення.

4. Атрибутам однозначно привласнюються імена, і в кожному з них розміщуються однорідні значення даних.

5. Повний інформаційний вміст БД подається у вигляді явних значень даних, і такий метод подання є єдиним. Зокрема, не існує будь-яких спеціальних зв'язків або показників, що з'єднують одну таблицю з іншою.

6. При виконанні операцій з відношенням його кортежі й атрибути можна обробляти в будь-якому порядку безвідносно до їх інформаційного вмісту

6.6. Базові засоби маніпулювання реляційними даними

Базові засоби маніпулювання реляційними даними визначають два механізми:

1) реляційна алгебра, яка заснована на теорії множин;

2) реляційне числення, яке ґрунтується на математичній логіці, точніше на обчисленні предикатів першого порядку.

Зазвичай розглядають два види обчислень: обчислення доменів і обчислення предикатів (предикат – це вислів, у який можна підставляти аргументи).

Ці механізми мають важливу властивість – вони замкнені відносно поняття відношення. Це означає, що вираз реляційної алгебри і формули реляційного числення визначають над відношенням реляційної БД і результатом обчислення також є відношення.

В результаті будь-який вираз або формула можуть інтерпретуватись як відношення, що дозволяє їх використовувати в інших виразах або формулах.

Реляційна алгебра має велику вразливу потужність, тобто дуже складні запити до БД можуть бути виражені за допомогою одного виразу реляційної алгебри або однієї формули реляційного числення.

Конкретна мова маніпулювання реляційною БД називається реляційно повною, якщо будь-який запит, що здійснюється за допомогою одного виразу реляційної алгебри або однієї формули реляційного числення, може бути виражений за допомогою одного оператора цієї мови.

Механізми реляційної алгебри і реляційного числення різняться рівнем процедурності. Вирази реляційної алгебри будуються на основі алгебраїчних операцій високого рівня, і подібно до того, як інтерпретуються арифметичні і логічні вирази реляційної алгебри, вони так само мають процедурну інтерпретацію.

Для формули реляційного числення однозначна інтерпретація, зазвичай, відсутня. Формула ж тільки встановлює умови, які повинні задовольняти кортежі результуючого відношення. Тому мови реляційного числення є менш процедурними або декларативними.

Оскільки механізми реляційної алгебри і реляційного числення є еквівалентними, то для перевірки ступеня реляційності певної мови БД можна скористатися будь-яким з цих механізмів. Зазвичай мова ґрунтується на суміші алгебраїчних і логічних конструкцій, наприклад SQL.

6.7. Запис SQL-операторів

Для успішного вивчення мови SQL необхідно навести короткий опис структури SQL-операторів і нотації, що використовуються для визначення формату різних конструкцій мови.

Оператор SQL складається із зарезервованих слів, а також зі слів, що визначаються користувачем. Зарезервовані слова є постійною частиною мови SQL і мають фіксоване значення. Їх треба записувати точнісінько так, як це встановлено, не можна розбивати на частини для переносу з одного рядка на інший. Слова, що визначаються користувачем, задаються ним самим (відповідно до синтаксичних правил) і становлять ідентифікатори або імена різних об’єктів БД. Слова в операторі також розташовуються відповідно до встановлених синтаксичних правил.

Ідентифікатори мови SQL призначені для позначення об’єктів у БД і є іменами таблиць, представлень, стовпчик та інших об’єктів бази даних.

Символи, які можуть використовуватись у створюваних користувачем ідентифікаторах мови SQL, повинні бути визначені як набір символів.

Стандарт SQL задає набір символів, який використовуються за замовчуванням. Він включає рядкові і прописні літери латинського алфавіту (A–Z, a–z), цифри (0–9) і символ підкреслювання (_). На формат ідентифікатора накладаються такі обмеження: – ідентифікатор може мати довжину до 128 символів; – ідентифікатор повинен починатись з букви; – ідентифікатор не може містити проміжки. Приклад запису:

<ідентифікатор>::=<буква> {<буква>|<цифра>}[...n]

Більшість компонентів мови не чутливі до регістру. Оскільки в мові SQL вільний формат, окремі SQL-оператори та їх послідовності будуть мати більшу читабельність при використанні відступів і вирівнювання. Мова, в термінах якої даються описи мови SQL, називається метамовою.

Синтаксичні визначення зазвичай задають за допомогою спеціальної металінгвістичної символіки, яка називається формулами Бекуса–Науера (БНФ). Великі літери використовуються для запису зарезервованих слів. Малі літери використовуються для запису слів, що визначаються користувачем. Використовувані в нотації БНФ символи і їх позначення представлено в табл. 6.1.

Таблиця 6.1

Використовувані в нотації Бекуса–Науера

Символ	Позначення
::=	Дорівнює за визначенням
|	Необхідність вибору одного з декількох наведених значень
<…>	Опис за допомогою метамови структури мови
{…}	Обов’язковий вибір певної конструкції зі списку
[…]	Необов’язковий вибір певної конструкції зі списку
[,…n]	Необов’язкова можливість повторення конструкції від нуля до декількох разів

6.8. Засоби маніпулювання відношеннями

Основна ідея реляційної алгебри – засоби маніпулювання відношеннями можуть ґрунтуватись на традиційних теоретико-множинних операціях, доповнених певними спеціальними операціями, які є специфічними для БД.

Існує декілька підходів до визначення реляційної алгебри. В початковому варіанті набір основних алгебраїчних операцій дорівнює восьми, які поділяються на теоретико-множинні операції та спеціальні реляційні операції.

До першого класу входять:

1) об’єднання відношень;
2) перетинання відношень;
3) різність відношень;
4) прямий добуток відношень.

До другого класу входять:

5) обмеження відношень;
6) проекція відношень;
7) з’єднання відношень;
8) ділення відношень.

Крім того, до складу алгебри включається операція присвоювання, яка дозволяє зберегти в БД результати обчислень алгебраїчних виразів, і операція перейменування атрибутів, яка надає можливість коректно сформулювати заголовок результуючого відношення.

6.9. Загальна інтерпретація реляційних операцій

Майже всі операції набору мають наступну інтерпретацію:

1. При виконанні операцій об’єднання 2-х відношень здійснюється відношення, яке включає всі кортежі, що входять в одне з відношень операндів.

2. Операція перетинання 2-х відношень здійснює відношення, які включає всі кортежі, що входять в обидва відношення-операнди.

3. Відношення, які є різницею 2-х відношень включають всі кортежі, що входять у відношення – перший операнд – такі, що жоден з них не входить у відношення, які є другим операндом.

4. При виконанні прямого добутку 2-х відношень здійснюється відношення, кортежі якого є конкатенацією (зчепленням) кортежів першого і другого операндів.

5. Результатом обмеження відношень за певною умовою є відношення, що включає кортежі відношення-операнда, які задовольняють цю умову.

6. При виконанні проекції відношення на заданий набір його атрибутів здійснюється відношення, кортежі якого виконуються шляхом взяття відповідних значень з кортежів відношення-операнда.

7. При з'єднанні 2-х відношень за певною умовою утворюється результуюче відношення, кортежі якого є конкатенацією 1-го і 2-го відношень і задовольняють цю умову.

8. В операції реляційного ділення два операнди: бінарне й унарне відношення. Результуюче відношення складається з атрибутивних кортежів, що включають значення першого атрибута кортежів 1-го операнда таких, що множина значень 2-го атрибута (при фіксованому значенні 1-го атрибута) збігається з множиною відношень 2-го операнда.

9. Операція перейменування здійснює відношення, тіло якого співпадає з тілом операнда, але імена атрибутів змінені.

10. Операція присвоювання дозволяє зберегти результат обчислення реляційного виразу в існуючому відношенні БД.

Оскільки результатом будь-якої операції, крім операції присвоювання, є певні відношення, можна утворювати реляційні вирази, в яких замість відношення-операнда деякі реляційні операції перебувають вкладеними в реляційний вираз: $A \cup B, A \cup (B \cap C)$. Кожне відношення характеризується схемою і набором кортежів. Заголовок відношення представляє собою множину пар (ім'я атрибута і домену), отже, домени атрибутів результуючого відношення однозначно визначаються доменами відношень операндів.

Однак, іноді виникає конфлікт імен атрибутів. Для розв'язання цього конфлікту до складу операцій реляційної алгебри і введено операцію перейменування. Її слід застосовувати в будь-якому випадку, коли виникає конфлікт іменування атрибутів у відношеннях-операндах однієї реляційної операції.

Тоді до одного з операндів спочатку застосовується операція перейменування, а потім виконується основна операція.

Об'єднання відношень – називається відношення, яке містить множину кортежів, що належать або 1-му, або 2-му вихідним відношенням, або двом відношенням одночасно.

Нехай задані 2 відношення:

$R_1=\{r_1\}$ і $R_2=\{r_2\}$, де r_1 і r_2 відповідно кортежі відношень R_1 і R_2, тоді об'єднання:

$$R_3 = R_1 \bigcup R_2 = \{r \mid r \in R_1 \vee r \in R_2\},$$

де r – кортеж нового відношення, $\vee$ – операція логічного додавання АБО (OR)

Перетинання відношень – відношення, яке містить множину кортежів, що належать одночасно і 1-му, і 2-му відношенням.

$$R_4 = R_1 \bigcup R_2 = \{r \mid r \in R_1 \wedge r \in R_2\},$$

де $\wedge$– операція логічного множення І (AND).

Різність відношень R_1 і R_2 – відношення, що містить множину кортежів, належить R_1 і не належить R_2.

$$R_5 = R_1 / R_2 = \{r \mid r \in R_1 \wedge r \notin R_2\},$$

$$R_6 = R_2 / R_1 = \{r \mid r \notin R_1 \wedge r \in R_2\},$$

Перші дві операції є комутативними, тобто результат операції не залежить від порядку аргументів в операції.

Операція ж різниці є принципово несиметричною, тобто результат буде різним для різного порядку аргументів.

Прямим добутком відношення R_1 в ступені n зі схеми $S_{R_1} = (A_1, A_2, ...A_n)$ і відношення R_2 ступеня m зі схеми $S_{R_2} = (B_1, B_2, ...B_m)$ називається відношення R_3 ступеня n+m зі схеми $S_{R_3} = (A_1, A_2, ...A_n, B_1, B_2, ...B_m)$, що містить кортежі, отримані зчепленням (конкатенацією) кожного кортежу r відношення R_1 з кожним кортежем r відношення R_2, тобто R_1= {r}, R_2 = {q}, тоді

$$R_1 \otimes R_2 = \{(r;q) \mid r \in R_1 \wedge q \in R_2\}$$

Зчепленням, або конкатенацією, кортежів c = < c_1, c_2,..., c_n > і q = < q_1, q_2,...,q_m> називається кортеж, отриманий додаванням значень другого в кінець першого.

Зчеплення позначається (c, q).

Операцію добутку з урахуванням можливості перестановки атрибутів у відношенні можна вважати симетричною. Дуже часто операція добутку використовується для одержання певного універсуму, тобто відношення, яке характеризує всі можливі комбінації між елементами окремих множин, однак самостійного значення результат виконання операцій не має, він бере участь у подальшій обробці.

ЛЕКЦІЯ 7
ОРГАНІЗАЦІЯ ЗБЕРЕЖЕННЯ ДАНИХ У ГІС

7.1. Загальні відомості про збереження даних у ГІС

Організація даних – процес зведення різнорідних даних і моделей у єдину логічно несуперечливу модель, яку в подальшому можна буде ефективно застосовувати в різних технологіях аналізу та управління [19].

Зібрані дані можуть зберігатися у вигляді наборів або файлів. Крім того, при зведенні дані можуть організовувати пов'язані сукупності, які називають моделями. Для того щоб різнорідні дані й моделі можна було опрацьовувати в одній системі, вони повинні бути упорядковані й зведені до єдиної інформаційної моделі, де будуть доповнювати одне одного.

Результатом організації даних є створення такої інформаційної моделі, яка дозволяє організувати ефективне збереження в базі даних і ефективну обробку в інформаційних системах та різних технологіях.

Організація даних надає геоданим якісно нові властивості. Саме організація даних дозволяє використовувати геодані при розв'язку широкого кола прикладних задач управління, аналізу, логістики, планування, проектування, прогнозування, використання ресурсів, моніторингу тощо. Вихідна та попередньо опрацьована інформація включає велику кількість параметрів, деякі з яких можуть дублювати один одного. Зменшення кількості геоданих про реальні об'єкти досягається застосуванням моделей, що зберігають основні властивості об'єктів дослідження і які не містять другорядних властивостей.

Однією з особливостей збору даних у геоінформатиці є те, що вихідні дані можуть мати не тільки різні розмірності, але й вимірюватися в різних шкалах. Організація даних у геоінформатиці створює умови для зведення даних різної розмірності та шкал вимірів у єдине середовище та їх спільний аналіз.

У процесі організації даних усе розмаїття вхідної інформації – про об'єкти, їх характеристики, про форми і зв'язки між об'єктами, різні описові відомості – перетворюються на набори моделей. При обробці даних в ГІТ використовують інформаційні моделі.

Інтегрована інформаційна модель становить сукупність простих моделей. Для ефективної обробки даних ця сукупність повинна певним чином бути оптимізована. Це завдання розв'язується шляхом організації даних. Вибір того або іншого способу організації даних в ГІС, в першу чергу саме тієї або іншої моделі даних, має ключове значення.

Вибір моделі даних напряму визначає більшість функціональних можливостей створюваної ГІС, оскільки деякі функції або просто неможливо реалізувати для певних типів організації даних, або вони забезпечуються досить складними маніпуляціями.

Організація даних в ГІС напряму визначає і застосовність тих або інших технологій введення даних. Тією ж мірою від неї залежить досяжна просторова

точність подання графічної частини інформації, можливість отримання якісного картографічного матеріалу й організації контролю якості карт.

Значною мірою спосіб організації даних у ГІС визначає також досяжну швидкодію системи, наприклад, при виконанні запиту або візуалізації на екрані. Можливість працювати з великими об'ємами даних або з точними даними по великих територіях також пов'язана зі способами та формами організації даних. Зручність редагування й оновлення даних, можливості організації багатокористувацької роботи в режимі редагування, створення розподілених по мережі баз даних – це все також пов'язане,

в першу чергу, з організацією даних і вже у другу – з конкретним програмним забезпеченням [25].

На підставі викладеного можна виділити наступні завдання системної організації даних у ГІС:

1) перетворення інформації як описових відомостей;

2) зведення множини геоданих до єдиної інтегрованої інформаційної моделі;

3) класифікація вихідних даних і моделей при перетворенні їх на інтегровану модель;

4) ідентифікація даних у процесі перетворення даних на інтегровану модель, чим забезпечується збереження їх індивідуальності;

5) встановлення додаткових зв'язків між геоданими на основі їх інтеграції;

6) уніфікація вихідних даних і створення можливості обробки й аналізу даних, виміряних у різних шкалах і з різними розмірностями, в єдиній системі;

7) створення бази для розв'язання основного завдання геоінформатики – встановлення просторових відношень між просторовими процесами, об'єктами, явищами та їх характеристиками.

Просторова (картографічна) інформація – основа інформаційного блоку ГІС, тому способи її формалізації є найважливішою складовою частиною ГІС. Як відомо, просторова інформація ГІС складається з метричної частини, яка описує позиційні властивості об'єктів (процесів, явищ), та атрибутивної частини (змістові (семантичні, тематичні)) характеристики. Робочим середовищем при роботі з просторовою інформацією є проект.

Проект у ГІС – спеціальний файл, де знаходиться організована певним чином просторова інформація.

Банк даних (БнД, databank, data bank) – інформаційна система централізованого збереження і колективного використання даних, що містить сукупність баз даних, СКБД та комплекс прикладних програм.

БнД називають локальним (local databank), якщо він міститься в одному обчислювальному центрі (ОЦ) або на одному комп'ютері, і розподіленим (distributed databank), коли сукупність локальних БнД поєднується за допомогою мережевих серверів у систему.

Сукупність просторової й атрибутивної інформації про конкретну територію утворює картографічний банк даних. Картографічні банки даних іменуються також банками цифрових карт (БЦК).

7.2 Типи файлів бази даних

Організована у вигляді бази даних просторова інформація в ГІС, складається з файлів у таких можливих формах: прості списки (невпорядковані файли), упорядковані послідовні файли та індексовані файли.

7.2.1. Невпорядковані файли

Невпорядковані файли (списки) є найпростішою структурою, яка реалізує неупорядкований масив записів. Прикладом невпорядкованих файлів можуть слугувати записи у записній книжці, сукупність карт у картотеці, у тому разі, коли записи занотовуються до записної книжки або картотеки у певній послідовності (рис. 15.1) [15].

Єдиною перевагою такої структури файлу даних є те, що новий запис занотовується в кінець файлу. Наявність такої картотеки дозволяє здійснити пошук будь-якого імені (атрибута), однак відсутність упорядкування карток робить процес пошуку досить тривалим, особливо у випадках, коли база даних включає велику кількість записів.

Наприклад, якщо для вибірки однієї картки потрібна одна секунда, а кількість карток становить 200 000, то пошук у середньому займе:

$$\tau = \frac{(n+1)}{2} = \frac{(200000+1)}{2} = 27.7(\textit{годин})$$

де τ – кількість операцій;

n – кількість карток, що підлягає перебиранню.

Таким чином, для наведеного прикладу і даного розміру файлу (картотеки) пошук потрібної інформації може складати майже 28 годин.

7.2.2. Послідовно впорядковані файли

Послідовно впорядковані файли *(ordered sequential files*) мають певний *ключ* (літери або цифри), згідно з яким відбувається пошук даних (рис. 7.2).

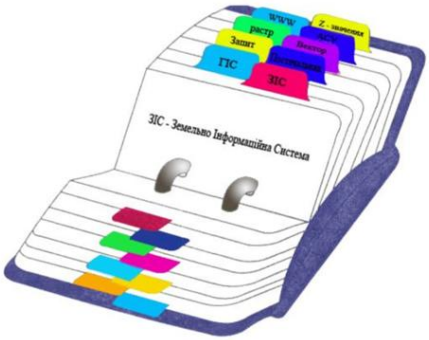


Рис. 7.1. Приклад невпорядкованого масиву даних

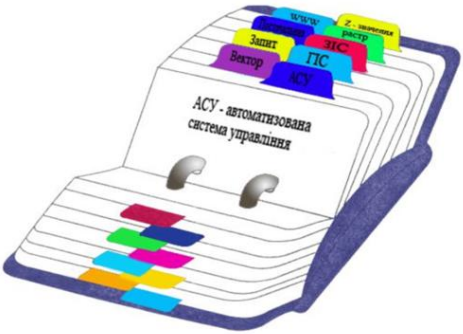


Рис. 7.2. Приклад послідовно впорядкованого масиву даних, де сортування виконується з використанням алфавіту

Зазвичай, для пошуку використовується *дихотомія* (пошук даних поділом файлу навпіл). Пошук починається поділом усього масиву записів на дві частини і вибіркою запису в середині. Якщо точка поділу виявляється тією, що потрібна, то процедура пошуку вважається закінченою. Якщо необхідний запис знаходиться раніше обраної точки, то спочатку виконується операція з першою половиною, якщо після – то з другою половиною.

Використання дихотомії не передбачає перегляду значної частини файлу. Середня кількість операцій у цій стратегії становить:

$$\tau = log2(n+1).$$

Для попереднього прикладу час пошуку становить трохи більше 2 годин, замість попередніх 28.

7.2.3. Індексовані файли

У вищенаведених прикладах записи ідентифікувались і порівнювались за ключовим атрибутом – літерою, словом або ознакою. Стратегія пошуку була заснована на значеннях самих атрибутів.

Оскільки в ГІС кожному об’єкту привласнюються певні семантичні характеристики (атрибути), то й пошук елементів здійснюється з певним набором атрибутів.

Кожний просторовий об’єкт ГІС може описуватись великою кількістю атрибутів, а сортування або пошук можна виконувати тільки одним способом.

Тобто, якщо за одним атрибутом можна застосувати швидкий пошук поділу навпіл, то для всіх інших атрибутів доведеться виконувати виснажливий послідовний пошук. Потрібно знайти якийсь вихід, бо інакше доведеться пересортовувати файл для кожного запиту.

Вихід існує при застосуванні зовнішнього індексу, який будується таким чином: з вихідного файлу в новий копіюються значення одного атрибута для всіх записів разом із розташуванням цих записів.

Тобто кожний запис у новому файлі складається зі значення атрибута й адреси запису у вихідному файлі, з якого це значення було взяте. Потім упорядковуються записи нового файлу, згідно зі значенням атрибута. Щоб знайти записи із заданим значенням атрибута у новому файлі, можна використати поділ навпіл. Знайшовши потрібні записи в індексному файлі, отримаємо адреси вихідного файлу, за яким можна отримати всі атрибути об'єктів.

Таким чином, для пошуку певних даних в основному файлі використовується додатковий індексний файл, що називається *зовнішнім індексом*, а сам вихідний файл стає *індексованим*. Це надає можливість вносити до індексованого файлу декілька атрибутів, значення яких дозволяють організувати пошук.

Використання зовнішнього індексу передбачає три умови:

– по-перше, потрібно знати заздалегідь критерії, за якими буде виконуватись пошук;

– по-друге, посилання на всі додавання у вихідний файл повинні поміщатись у відповідні місця індексних файлів, не порушуючи їх упорядкованості;

– по-третє, якщо не буде передбачений з якихось причин певний критерій пошуку, то доведеться використовувати послідовний перебір для отримання потрібної інформації [15].

7.3. Принципи організації даних у ГІС

Просторова організація інформації в ГІС застосовується на практиці у різних модифікаціях і поєднаннях.

Існує два підходи до організації просторових даних у ГІС – *пошаровий* (*layer*), який іноді називають класичним, та *об'єктно орієнтований*.

Зважаючи на те, що сучасні ГІС, як правило, є динамічними системами, виникає потреба коректної організації інформації в часовому інтервалі.

Характеристика і визначення часу спричиняють найбільші проблеми. Часто важливим є відносний, а не абсолютний час спостереження (тобто послідовність, у якій виникають явища). У цьому разі необхідний результат досягається впорядкуванням даних, а не фіксацією абсолютного часу.

ГІС дає змогу реалізувати ретроспективне відновлення динаміки просторових процесів у вигляді створення динамічних рядів і карт розподілу. Для цього необхідне визначення раціональної глибини ретроспекції і вибору оптимального кроку дискретизації даних (часової розрізненості системи). Крок дискретизації в часі може відрізнятися на багато порядків (в десятки і сотні разів) для різних типів даних і мати обмеження, пов'язані як з інструментальною частотою спостережень, так і з природним ходом процесів. Тому необхідна така просторова організація даних, яка б передбачала можливість різної розрізненості

(реалізованої в побудові регулярних ієрархічних мереж), – ієрархічна система часової розрізненості.

7.3.1. Пошаровий принцип організації даних

Пошарова організація збереження просторових даних є найбільш поширеною технологією в ГІС. Сутність цієї організації полягає в тому, що однорідна просторова й атрибутивна інформація про певну територію подається у вигляді тематичних шарів (over lay – покриттів, тем, пластів).

Поділ інформації на шари є інтуїтивно зрозумілим і легко співвідноситься із загальноприйнятими принципами використання прозорих кальок-накладок при роботі з паперовими картами.

Шар (layer, theme, coverage, overlay) – сукупність однотипних (однієї мірності) просторових об'єктів, що відносяться до однієї теми (класу об'єктів) у межах певної території і в системі координат, яка є загальною для цього набору шарів.

Покриття (coverage) – цифрова модель одиниці збереження бази векторних даних ГІС, яка зберігає у вигляді записів усі об'єкти первинного (точки, дуги, полігони) і вторинного (координати опорних точок, анотації тощо) рівнів певного просторового об'єкта, а також структуру відношень між ними, зокрема топологічну.

Покриття визначається в контексті його змістової визначеності (рослинність, рельєф, адміністративний поділ тощо) або його статусу в середовищі редактора (активний шар, пасивний шар). Покриття можебути порожнім.

Порожнє покриття – покриття, в якому відсутні будь-які просторові об'єкти.

Шари (покриття) поєднуються в цифрові карти. Карти можуть не підтримувати у своїй структурі покриття, однак у цьому разі беруть частину або всі функції покриттів на себе.

Шар однорідний не тільки за тематикою, але й за типам об'єктів. Кожний тематичний шар містить об'єкти певного виду, що поєднуються загальними характеристиками. Шар даних (тема) відповідає логічній сукупності просторових об'єктів із загальними характеристиками. Кожний шар / тема визначається наступними умовами: об'єкти одного класу, однаковий набір полів. Геометричний тип об'єкта визначає 6 класів шарів або тем (рис. 7.3).

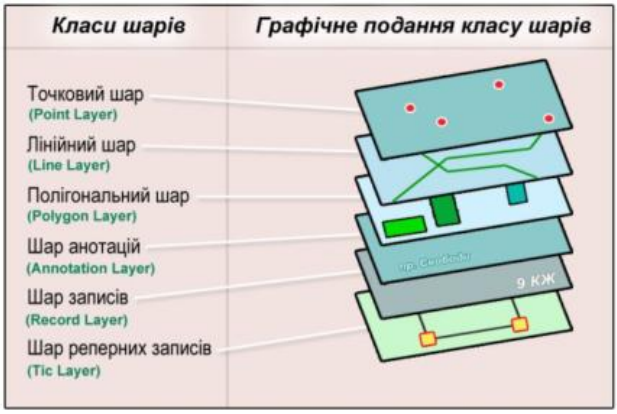


Рис.7.3. Визначення тематичних шарів

Концепція пошарового подання графічної інформації запозичена із систем автоматизованого проектування (CAD-систем), однак у ГІС вона набула якісно нового розвитку. Принципова відмінність полягає в тому, що шари в ГІС можуть бути як векторними, так і растровими, причому векторні шари обов'язково повинні мати одну з трьох характеристик векторних даних, тобто векторний шар повинен бути визначений як точковий, лінійний або полігональний додатково до його тематичної спрямованості.

В табл. 7.1 наведено приклади деяких шарів даних із типової ГІС муніципального управління.

Інша важлива відмінність пошарового подання векторних даних полягає в тому, що вони є об'єктними, тобто містять інформацію про об'єкти, а не про окремі елементи об'єкта, як у САПР.

В ГІС виділяють *вертикальну* і *горизонтальну* пошарову організацію даних.

Вертикальна пошарова організація даних.. Інструментом просторового взаємозв'язку шарів по вертикалі є єдина для усіх шарів система координат для визначення просторових об'єктів.

Сукупність шарів утворює інтегровану основу графічної частини ГІС (рис. 7.4).

Належність об'єкта або частини об'єкта до шару дозволяє використовувати і додавати групові властивості об'єктам даного шару. Аяк відомо з теорії обробки даних, саме їх *групова обробка є основою підвищення продуктивності автоматизованих систем*.

Пошарова організація електронної карти за наявності гнучкого механізму управління тематичними шарами дозволяє об'єднати та відобразити не тільки велику кількість інформації, на відміну від звичайної карти, але й істотно спростити аналіз картографічних даних за допомогою селекції, необхідної для візуалізації та механізму "прозорості" цифрової карти. Поділ інформації на тематичні шари дозволяє розв'язувати завдання типізації і розбивки даних на типи, підвищувати ефективність інтерактивної обробки і групової

автоматизованої обробки, спрощувати процес збереження інформації в базах даних, включати автоматизовані методи просторового аналізу на стадії збору даних і при моделюванні, спрощувати розв'язання експертних задач.

Таблиця 7.1

Приклади шарів даних

Назва шару	**Об'єкти реального світу**	**Геометричний тип об'єкта**	**Атрибути користувача**
Осі вулиць	Міські вулиці	Лінійний	Назва, клас вулиці
Дорога	Проїжджі частини вулиць	Полігональний	№, площа, ширина
Квартали	Квартали міста	Полігональний	Щі забудови льність
Будівлі житлові	Будівлі	Полігональний	Кількість поверхів, кількість квартир, кількість мешканців
Будівлі промислові	Будівлі	Полігональний	Висота, об'єм
Земельні ділянки	Зонування	Полігональний	Код ділянки, площа, тип землекористування
Залізниця	Основні залізничні лінії	Лінійний	Назва залізниці
Лінії газо забезпечення	Система газозабезпечення	Лінійний	Діаметр, тиск, довжина
Лінії водо забезпечення	Система водо забезпечення	Лінійний	матеріал, довжина Діаметр,
Колодязі водо забезпечення	Система водо забезпечення	Точковий	Позначка верху, глибина, запірний пристрій
Скарги на каналізацію	Місця розташування аварійних ділянок каналізації	Точковий	Адреса, дата, код аварії

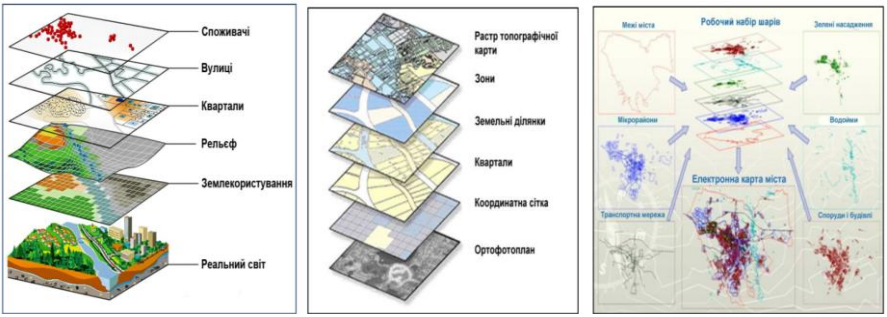


Рис. 7.4. Пошарова організація даних у ГІС

Введення топологічних властивостей у графічні дані ГІС дозволяє розв'язувати задачі, які методами програмного забезпечення САПР взагалі не реалізуються.

Наприклад, це можливість накладення шарів для отримання нового шару, який є не просто результатом накладання, а містить нові об'єкти, отримані на основі методів просторового аналізу з використанням логічних операцій. Крім того, пошаровий спосіб подання інформації дозволяє дрібні території, які зазвичай відповідають аркушам карти, комбінувати у більші одиниці (досліджувані ділянки).

В деяких ГІС у шарі можуть міститися об'єкти одного типу, а не однієї теми, наприклад, шари точкових, лінійних чи полігональних об'єктів. Інколи в шарі можуть бути об'єкти різні і за типом, і за темою. Однак найпоширенішим випадком все ж залишається логічний поділ на шари, коли кожен шар визначається як за тематикою, так і за типом, що допомагає організувати споріднені об'єкти шляхом мінімізації числа зв'язків.

Працюючи в ГІС, користувач може підключати і відключати тематичні шари, змінювати порядок їх відображення, згідно зі своїми потребами. Кожний шар (пласт) карти містить інформацію, яка відноситься до однієї або декількох тем.

Наприклад, для задач планування розвитку міської території такий набір може включати дані про вулиці, міські інженерні мережі, об'єкти транспортної інфраструктури, різні типи зонування міської території, землеволодіння, нерухомості тощо.

На рівні шарів здійснюються пошук, завантаження і вивантаження даних у середовище ГІС. До об'єктів шару застосовуються функції пошуку, форматування, зміни графічних змінних.

Кожний тематичний шар характеризується такими властивостями:

– видимістю (visible) – підключається чи відключається відображення певного шару на екрані (при цьому шар залишається в оперативній пам'яті та бере участь в усіх інших дозволених операціях). Крім цього, є функції відображення шару залежно від масштабу екранного подання, задаються найменший і найбільший масштаби, за яких шар є видимим на екрані;

включається чи виключається відображення службової інформації для окремих об’єктів шару – опорних точок, центроїдів полігонів, напрямків ліній тощо;

– редагованістю (editable) – у шар, що редагується, дозволено вносити зміни за допомогою всіх доступних інструментів створення і редагування форми об’єктів, а також змінювати графічні змінні об’єкти. Зазвичай, можна редагувати тільки один шар;

– участю в запитах (selectable) – з шару можна отримувати атрибутивну інформацію за допомогою різних засобів побудови запитів, в іншому разі всі запити ігноруються;

– можливістю автоматичного друку пояснювальних написів (auto label, labeled) – у відповідному шарі включається режим автоматичного друку пояснювальних написів для картографічних об’єктів, наприклад, назв країн, міст, вулиць.

За замовчуванням для напису береться вміст першого текстового поля з атрибутивної бази даних, є можливість налаштування на будь-яке інше поле бази даних або використання як напису результату обчислень (злиття фрагментів тексту) у декількох полях. Так само може задаватися формат відображення напису – шрифт, розмір і колір шрифту, прив’язування до центра точки, лінії чи полігона (у центрі або збоку, зі зсувом вниз чи вгору). Задається метод контролю накладання і дублювання написів (наприклад, написи не можуть накладатися один на одного за певного масштабу, не може бути двох однакових написів тощо).

Для відображення службової інформації поверх усіх відкритих шарів даних може створюватись "*косметичний*" шар. Вміст "косметичного" шару існує до того часу, поки залишається відкритим базовий шар, відносно якого виводиться службова інформація. За потреби "косметичний" шар може бути збережений у вигляді окремого файлу даних.

Спеціальні шари утворюються при створенні тематичних карт, вони прив’язані до шару, на основі якого була створена тематична карта. Переміщення за списком накладення шарів базового шару зумовлює переміщення і похідного тематичного шару.

Інформаційні компоненти тематичного шару можуть бути *зовнішніми* (векторні та растрові шари, таблиці, бібліотеки символів) або *внутрішніми* (спеціальні типи шарів, запити, карти, макети друку тощо).

Сучасні технології введення просторових даних у ГІС, їх інтерпретації та збереження передбачають поелементний поділ змісту існуючих карт (дані про рельєф, гідрографічну мережу, населені пункти, дорожню мережу, адміністративні межі тощо). Для забезпечення зручності збереження й обробки великих наборів даних кожний із тематичних шарів додатково може бути поділений на *фрагменти*. Зазвичай поділ на фрагменти успадковує прийняту схему розграфлення карт (за окремими аркушами топокарт, градусною сіткою тощо). Логічна нерозривність отриманого фрагментованого шару забезпечується засобами, що підтримують безшовні бази даних.

Після операції "фрагментування" (tiling) при наступному відображенні на екрані комп'ютера ГІС надають можливість виконувати зворотні операції – "зшивання".

З урахуванням того, що банк картографічних даних у ГІС може включати сотні шарів однорідної просторової інформації, це відкриває широкі можливості для побудови первинних оригіналів поелементних карт на основі шарів однорідних картографічних даних.

У разі збігу систем координат можливе багаторазове накладення картографічних шарів як у векторному, так і растровому форматі.

Оскільки растрові карти непрозорі, то вони використовуються як основа на задньому плані комбінованого векторно-растрового зображення.

Кількість одночасно виведених на екран картографічних шарів обмежується тільки ресурсами комп'ютера.

Для забезпечення контролю й управління візуалізацією картографічних шарів у ГІС існують спеціальні інструменти (функції відкриття і закриття одного шару, відкриття і закриття групи шарів, закриття усіх раніше відкритих шарів тощо). За одночасного відкриття й перегляду декількох шарів необхідно впорядковувати їх взаємне розташування і перекриття. В екранних вікнах, що керують відображенням шарів, можна побачити розміщення окремого шару порівняно з іншими шарами, а також покроково перемістити обраний шар нагору або вниз усієї сукупності шарів.

Карта в ГІС – сукупність різних шарів, визначених на спільній території і в загальній системі координат.

При створенні нового проекту необхідно створити нові або підключити раніше створені тематичні шари. Векторні шари (що містять точкові, лінійні або площинні об'єкти) можуть створюватись безпосередньо в середовищі ГІС або в інших програмних середовищах (наприклад, це може бути креслення в обмінному або двійковому форматі AutoCAD).

У шари можуть бути завантажені растрові зображення різних форматів (які використовуються в цифровій картографії). З кожним векторним шаром може пов'язуватись таблиця характеристик, що зберігається з векторним шаром, і набір таблиць з атрибутивними (тематичними) даними, які зберігаються у зовнішній СКБД (рис. 7.5).

Shape	ID	PIN	Area	Addr	Code
	1	334-1626-001	7,342	341 Cherry Ct.	SFR
	2	334-1626-002	8,020	343 Cherry Ct.	UND
	3	334-1626-003	10,031	345 Cherry Ct.	SFR
	4	334-1626-004	9,254	347 Cherry Ct.	SFR
	5	334-1626-005	8,856	348 Cherry Ct.	UND
	6	334-1626-006	9,975	346 Cherry Ct.	SFR
	7	334-1626-007	8,230	344 Cherry Ct.	SFR
	8	334-1626-008	8,045	342 Cherry Ct.	SFR

Пов'язана таблиця власників

PIN	Owner	Acq. Date	Assessed	Tax Stat
334-1626-001	G. Hall	1995/10/20	$ 115,500.00	02
334-1626-002	H.L. Hoines	1993/10/06	$ 24,375.00	01
334-1626-003	W. Rodgers	1980/09/24	$ 175,500.00	02
334-1626-004	J. Wiliamson	1974/09/20	$ 135,750.00	02
334-1626-005	P. Goodman	1966/06/06	$ 30,350.00	02
334-1626-006	K. Stakly	1942/10/24	$ 120,750.00	02
334-1626-007	J. Dormandy	1996/01/27	$ 110,650.00	01
334-1626-008	S. Gooley	2000/05/31	$ 145,750.00	02

Рис. 7.5. Зв'язок таблиці шару з внутрішньою таблицею СКБД по полю PIN

Для кожного шару можна визначити такі об'єкти бази даних:

– запити до атрибутивних таблиць;

– теми (варіанти тематичного картографування шару);

– форми подання довідкової інформації про об'єкти;

– діаграми (подання результатів у вигляді різноманітних графіків);

– макроси – зовнішньо виконувані програми або внутрішні функції

ГІС (задаються користувачем для карти в цілому, для тематичного шару або для окремих об'єктів).

Пошарова організація даних припускає, що тематичні шари в просторі по-перше, не мають розривів, по-друге, що скрізь існує певна інформація – навіть "відсутність об'єкта" або "немає даних про наявність чи відсутність об'єкта".

Тематика шару необов'язково повинна відповідати тематичному поділу. Наприклад, хоч шосе і річка є лінійними об'єктами карти, немає ніякого сенсу зберігати їх в одному й тому ж тематичному шарі. Атрибути річки можуть містити назву, класифікацію витрат води, тоді як атрибути дороги – найменування і тип покриття. Отже, атрибути доріг і річок різні. Оскільки атрибути різні, вони мають зберігатись у різних шарах, створених для однієї і тієї ж ділянки.

Тематичні шари для однієї ділянки є "вертикальними" шарами (рис. 7.6).

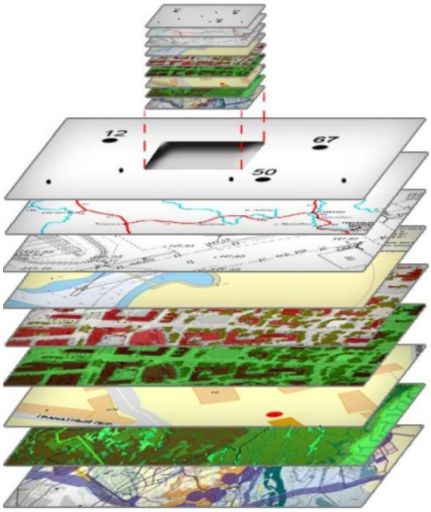


Рис. 7.6. Вертикальне подання тематичної інформації

Кількість, тематику та зміст шарів, як правило, визначають на етапі формування проекту. Визначаючи об'єктний зміст шарів, ураховують склад завдань, які буде розв'язувати ГІС, тому що сконцентрована інформація для одного випадку не буде обов'язково корисною для іншого. Наприклад, використовуючи технологію, орієнтовану на якісний та кількісний облік земель, доцільно всю контурну топографічну інформацію зібрати в одному шарі (темі), а розв'язуючи завдання, пов'язані з плануванням доріг або управлінням руху автотранспорту, інформацію про мережу доріг і населені пункти слід розміщувати в окремих шарах.

Також повинен бути визначений пріоритет введення або цифрування просторових даних шарів. Це є обов'язковим, оскільки часто один шар даних містить просторові об'єкти, які співпадають з іншими. Наприклад, озера можуть бути використані для визначення полігонів у шарі даних лісового кадастру. Шари, зазвичай, визначаються розробником, виходячи з потреб користувачів й доступності даних. Визначення шарів може відрізнятися залежно від потреб ГІС. Наприклад, кількість шарів у діючій регіональній ГІС може вимірятися декількома сотнями.

Доцільно створювати таку кількість шарів, яку можна виділити з елементарних тематичних груп інформації. Однак такий підхід не завжди виправданий, оскільки для того щоб в подальшому була можливість отримання інтегрованої інформації щодо перетину декількох шарів, виникає потреба в топологічних оверлеях, і їх буде тим більше, чим більше шарів було заведено.

З іншого боку, внесення всієї інформації в один шар також є не найкращим рішенням, тому що, по-перше, зміна інформації в одній тематичній групі може викликати потребу корегування усього шару, по-друге, під час розв'язання

окремих завдань не тільки немає потреби використовувати всю наявну інформацію, але й навпаки, її надмірна кількість може суттєво ускладнити роботу і спричинить потребу вичленовування із шару певної надлишкової частини інформації та перенесення її в інший шар.

Сукупність шарів (мультишарів) утворює інтегровану основу графічної частини ГІС.

Для створення баз даних суміжні покриття мають бути помічені однаковими реєстраційними точками, так званими реперами (benchmark), уздовж спільної межі. Ці спільні реєстраційні точки повинні мати однакові значення ідентифікаторів.

Репери – це реєстраційні точки, які визначають положення точок на земній поверхні, для яких відомі їх реальні координати.

Найкраще розробити і використовувати номенклатурну систему, яку варто застосовувати для всієї графічної бази.

Шари, які являють собою логічні набори об'єктів для одних і тих же ділянок, мають реєструватися один за одним по вертикалі. Два шари будуть зареєстровані, якщо можна позитивно відповісти на такі питання:

– чи вимірюються шари в однакових одиницях?

– чи знаходяться шари в одній координатній системі?

– чи мають спільні реєстраційні точки однакові ідентифікатори?

– чи викреслюються шари один над одним?

Шарова організація карти дозволяє:

• спростити аналіз картографічної інформації;

• виконувати тематичні або просторові вибірки;

• виконувати аналіз тощо.

У результаті аналізу тематичних шарів, що відображає "сирі" дані про об'єкти, будуються нові шари, згідно із завданням дослідження. Оскільки шар містить графічні образи, таблиці, числа, рядки, то така неоднорідність істотно ускладнює застосування звичайних СКБД, розрахованих на жорстку логічну структуру інформації, що зберігається у вигляді записів.

Зрозуміло, що цю умову для ГІС виконати неможливо.

Обхідний шлях передбачає поділ даних на однорідні групи з подальшим розміщенням у різних файлах, що негативно впливає на ефективність роботи. Крім того, ускладнюється введення нових типів даних і модернізації існуючих.

Поділ інформації на тематичні шари дає змогу:

– розв'язувати завдання типізації і розподілу даних на типи;

– підвищувати ефективність інтерактивного і групового автоматизованого опрацювань;

– спрощувати процес збереження інформації в базах даних;

– включати автоматизовані методи просторового аналізу на стадії збору даних і під час моделювання;

– спрощувати розв'язання експертних завдань тощо.

Підсумовуючи наведене, можна зробити висновок, що пошарове подання картографічної інформації надає широкі можливості аналізу картографічних даних.

Зазвичай у ГІС за один раз уводиться один шар даних. Шар даних буде повністю завантажений, коли будуть виконані графічні перетворення, редагування, топологічні побудови, трансформації атрибутів, зв'язування, а також перевірки перед тим, як буде запущений наступний шар даних.

Тому існує декілька етапів повного завантаження шару даних.

Більшість геоінформаціних проектів інтегрують шари даних для створення похідних тем або шарів, які становлять результат якогось обчислення або географічної моделі, наприклад, вартості лісів, придатності використання земель тощо.

Похідні шари цілковито залежать від цілі проекту. Кожний шар даних, інтегрований в індивідуальному порядку, топологічно буде вихідним для створення комбінованих даних шарів. Окремі функції аналізу даних можуть бути здійснені, ґрунтуючись на моделі даних, наприклад, растровій або векторно-топологічній структурі.

Важливо відзначити, що у векторній ГІС топологічна структура визначається тільки за допомогою унікальних міток для кожного просторового об'єкта.

Залежно від конкретної реалізації ГІС кількість шарів при пошаровій організації даних може обмежуватись, а може й не обмежуватись.

При пошаровій організації даних дуже зручно маніпулювати великими групами об'єктів, представлених шарами, як єдиним цілим, наприклад, включаючи або виключаючи шари для візуалізації, визначати операції, що ґрунтуються на взаємодії шарів. В цілому ж можна стверджувати, що *пошарова організація даних має великі аналітичні можливості.*

Об'єкти, віднесені до одного шару, утворюють певну фізично окрему одиницю даних. Вони збираються в один файл або в одну директорію, вони мають єдину і відмінну від інших шарів систему ідентифікаторів, до них можна звертатися як до якоїсь множини. З однієї теми може бути передбачено декілька шарів різного масштабу і відповідно різної точності або різних часових інтервалів. Ця ідея також використовується для логічного упорядкування даних у більшості геоінформаційних програм.

Горизонтальна пошарова організація даних.

При розробці логічної моделі даних із використанням реляційної моделі може знадобитися горизонтальний поділ одного тематичного шару, який отримав назву "*бібліотека карт*". Це робиться, головним чином, через зручність адміністрування баз даних і щоб уникнути роботи з занадто великими файлами. Внутрішня організація шарів даних у горизонтальній формі в ГІС отримала назву *просторового індексування*.

Аркуші (теми) в бібліотеці карт дають змогу легко оновлювати й зберігати дані шляхом просторового поділу. Одночасно можна аналізувати, запитувати та відображати операції на базі даних усього регіону.

Кожен з аркушів бібліотеки має починатися з порожнього стандартизованого шару-основи, який може включати кордон досліджуваної ділянки (екстент, extent) та розташування реєстраційних точок. Ці шари основи важливі для автоматизації. Вони гарантують, що суміжні аркуші будуть збігатися, а не перекриватись при їх поєднанні, або відображатися один біля одного. Реєстраційні точки, які мають один і той же ідентифікатор, будуть мати одне й те ж місцезнаходження. Якщо реєстраційні точки відсутні, то необхідно, щоб координати об'єкта в місці розриву суміжних аркушів збігалися, бо їх невідповідність призведе до неможливості автоматичного стикування, а це стане причиною збільшення ручних робіт.

Векторні ГІС використовують просторові індекси для більш швидкого доступу до об'єктів на певній ділянці карти. Індексування просторових об'єктів дозволяє зменшити обчислювальну складність процедур пошуку вкладених об'єктів або об'єктів, що перетинаються, тому індекси є важливою частиною алгоритмів накладення полігонів.

Просторове індексування – це метод використання програмного забезпечення для збереження й отримання просторових даних.

Існують різні підходи для прискорення процесу пошуку просторових об'єктів у межах програмного продукту ГІС. Більшість із них пов'язані з розподілом географічної ділянки на керовані підмножини або блоки. Ці блоки потім індексуються математично, наприклад, методом розбивки квадродерева або побудови R-дерева, щоб забезпечити швидкий пошук і витяг при запиті за ініціативою користувача.

Просторове індексування аналогічне визначенню аркушів карти, за винятком того, що конкретні методи індексування використовуються для доступу до даних через карту меж аркушів (блоків).

Це робиться для підвищення продуктивності запитів для великих наборів даних, які охоплюють декілька аркушів карти, а також для забезпечення цілісності даних за допомогою меж аркушів карти.

Метод і процес просторового індексування зазвичай є прозорим для користувача. Незважаючи на це, він є надзвичайно корисним при використанні особливо великих об'ємів даних. Поняття просторової індексації відіграє все більш важливу роль у розробці великомасштабних додатків із використанням ГІС.

Наприклад, якщо інформацію про дорожню мережу певного району можна виразити у вигляді таблиць з номерами доріг за зазначеним класифікатором і множиною числових характеристик або представити цю ж інформацію у вигляді карти, де дороги відображені у вигляді ліній, а пов'язані з ними числові характеристики (ширина, кількість смуг руху, тип покриття тощо) виражені у вигляді умовних знаків (тип лінії, колір, товщина).

Поділ інформації на фізичному рівні може повністю не відповідати її поділу на концептуальному рівні, наприклад, фізично згрупована в один шар інформація може представляти декілька різноманітних тематичних прошарків

для користувача. Такий розподіл різних рівнів організації даних досить зручний для практики.

Тематичні шари карт можуть поєднуватись, утворюючи базу даних (бібліотеку карт).

Важливо відзначити, що растрові системи за характером своєї структури даних зазвичай не вимагають методу просторового індексування.

Растровий підхід накладає регулярні, легко адресовані розділи за даними навколишнього світу нерозривно з їх структурою даних. Таким чином, просторова індексація, як правило, не потрібна. Однак у більш складних векторних ГІС потрібен спосіб швидкого отримання просторових об'єктів.

7.3.2. Об'єктно орієнтований принцип організації даних

Складність реального світу та його пізнання вимагають розвитку відповідних методів і засобів, у тому числі й у ГІС. У свою чергу розвиток ГІС пов'язаний із потребою спільної обробки зростаючих обсягів просторової й непросторової інформації, розробки більш складних процесів обробки взаємопов'язаної різнопланової інформації, інтеграції цієї інформації у взаємодії з іншими, різними за призначенням системами. Додаткові вимоги в частині знаходження оптимальних (раціональних) рішень, зручності, продуктивності, надійності та вартості також потребують розробки і розвитку адекватних моделей. Поширення потужних персональних комп'ютерів створило у 90-х рр. XX ст. основу для широкого застосування об'єктно орієнтованого підходу в практиці проектування та програмування інформаційних систем.

Нова методологія орієнтована на подолання складності, пов'язаної з розробкою програмних засобів, на створення великих складних систем, колективну їх розробку, наступний активний супровід при експлуатації і регулярні модифікації.

Об'єктно орієнтована модель розширює можливості застосування ГІС для максимального спрощення створення інтелектуальних просторових об'єктів, які відображають взаємодію і поведінку об'єктів реального світу. За цим принципом організації даних групування об'єктів відповідає їх логічним взаємозв'язкам. Об'єктно орієнтований принцип організації даних в ГІС фокусує увагу не стільки на загальних властивостях об'єктів (що моделюються через поділ на тематичні шари в попередньому підході), а на їх положенні у певній складній ієрархічній схемі класифікації і на взаємовідношеннях між об'єктами.

В силу цього зручно відображаються різні родинні і генетичні відношення між об'єктами, відношення співпідпорядкованості, функціональні зв'язки між об'єктами.

Об'єктно орієнтований підхід у ГІС базується на моделях, підходах і термінах об'єктно орієнтованих баз даних та об'єктно орієнтованого програмування, але має свої специфічні риси: зберігання просторових і семантичних даних в одній базі даних; зберігання форми просторового об'єкта; топологічно зберігає інтегровані набори класів об'єктів тощо. У ньому відсутній

поділ інформації на тематичні шари, точніше групування об’єктів у тематичні шари, з якими можна маніпулювати як з тематичними картами.

Групування здійснюється більш складним чином, згідно з логічними взаємозв’язками між ними, з побудовою ієрархій, які відповідають їх найбільшим загальним властивостям. Такий підхід наближений до структури людського мислення. Він ефективний, коли необхідне використання логічних взаємозв’язків об’єктів, але малоефективний при безперервному розподілі у просторі ознак (побудові рельєфу, оцінці питомого вмісту корисної копалини, забрудненні ґрунту важкими металами тощо).

В основі об’єктно орієнтованої методології лежить об’єктний підхід, при якому предметна прикладна сфера подається у вигляді сукупності об’єктів, які взаємодіють між собою за допомогою передачі повідомлень.

Об’єкт – сутність (реальна або абстрактна), що має стан,

поведінку та індивідуальність.

Стан об’єкта характеризується його структурою, переліком усіх можливих властивостей та значеннями кожної з цих властивостей.

Поведінка об’єкта (функціональність) – характеристика того, як об’єкт взаємодіє з іншими об’єктами або піддається взаємодії інших об’єктів, виявляючи свою індивідуальність.

Поведінка об’єкта реалізується у вигляді функцій, які називають методами. При цьому структура об’єкта є доступною тільки через його методи, які в сукупності формують інтерфейс об’єкта.

Індивідуальність об’єкта – властивості об’єкта, що відрізняють його від усіх інших об’єктів.

Для об’єктно орієнтованої методології особливий інтерес становлять два типи ієрархічних співвідношень об’єктів:

– зв’язки, що позначають рівноправні відношення між об’єктами; об’єкт співпрацює з іншими об’єктами через зв’язки, що з’єднують його з

ними;

– агрегація, яка описує відношення цілого і частини, що приводять до відповідної ієрархії об’єктів. Об’єкти утворюють мінімальні одиниці інкапсуляції.

Інкапсуляцією називається спосіб об’єднання структури і поведінки в одному місці (начебто в "капсулі") та приховання усіх даних усередині об’єкта, що робить їх невидимими для усіх, за винятком методів самого об’єкта.

В об’єктно орієнтованій моделі об’єкти інкапсулюють атрибути і лінії поведінки. Доступ до даних, що містяться в об’єкті, можливий тільки відповідно до ліній поведінки об’єкта.

Інкапсуляція захищає дані від ушкодження іншими об’єктами, а також закриває внутрішні деталі об’єктів від іншої системи.

Інкапсуляція також забезпечує ступінь незалежності даних, щоб не виникало потреби змінювати об’єкти-відправники або об’єкти, що отримують повідомлення, при їх взаємодії з об’єктом, поведінка якого змінилася.

Інкапсуляція – це сутність об’єктно орієнтованої моделі.

Одне з ключових понять об’єктно орієнтованого підходу – поняття класу.

Клас – множина об’єктів, що мають спільну структуру і спільну поведінку. Саме клас спочатку описує змінні і методи об’єкта, тобто структуру і поведінку об’єкта. Будь-який конкретний об’єкт – це екземпляр класу. Об’єкти, не пов’язані спільністю структури і поведінки, не можна об’єднати в клас, оскільки за визначенням вони нічим не пов’язані між собою.

Істотними є такі типи відношень між класами:

– відношення "узагальнення спеціалізація" (загальне і часткове) відбиває ступінь загальності, відношення "ціле-частина" відбиває агрегування об’єктів;

– відношення "асоціація" відбиває змістовий зв’язок між класами, які не пов’язані ніякими іншими типами відношень;

– спадкування – це таке відношення між класами, коли один клас повторює структуру і поведінку іншого класу (одиночне спадкування) або інших (множинне спадкування) класів.

Клас, структура і поведінка якого успадковуються, називається суперкласом. Похідний від суперкласу клас називається підкласом.

Це означає, що спадкування встановлює між класами ієрархію загального і часткового.

Відношення описують те, як об’єкти асоційовані один з одним. Вони визначають правила створення, зміни та видалення об’єктів. Існує декілька видів відношень, які можуть використовуватися в об’єктно орієнтованій моделі даних.

Спадкування – дозволяє одному класу успадковувати атрибути та лінії поведінки одного або декількох інших класів. Клас, що успадковує атрибути і лінії поведінки, відомий як підклас. Батьківський клас називається суперкласом. Крім успадкованої ними лінії поведінки, підкласи можуть додавати або перевизначити успадковані атрибути і лінії поведінки.

Суперклас – це генералізація його підкласів, а підклас – це уточнення свого суперкласу.

Наприклад, будинок – це уточнення будівлі, а будівля – це генералізація будинку. Клас будинків може успадковувати атрибути і лінії поведінки класу будівель, такі як кількість поверхів, кімнат та тип споруди.

Асоціація – загальні відношення між об’єктами.

Кожна асоціація може також мати асоційовану з нею множинність, яка визначає кількість об’єктів, асоційованих з іншим об’єктом. Наприклад, асоціація може свідчити, що "власник" об’єкта може мати один або багато будинків.

Об’єднання (агрегування) і композиція – це особливі типи асоціацій.

Об’єднання (агрегування) – визначений тип асоціації.

Об’єкти можуть містити інші об’єкти, тому об’єднання – це просто набір різних класів об’єктів, зібраних в один клас, який стає новим об’єктом. Класи об’єктів можуть збиратись в об’єднаний клас. Наприклад, об’єднаний клас "Земельномайновий комплекс" може бути створений шляхом об’єднання класів

"Земельна ділянка" і "Будинки". Ці нові складені об'єкти є важливими, оскільки вони, на відміну від простих об'єктів, спроможні представляти більш складні структури.

Композиція – ще одна спеціальна форма асоціації.

Це більш сильний

асоціативний взаємозв'язок, при якому життя "вмісту" класів об'єктів керує життям "змістового" класу об'єктів (контейнера). Наприклад, будинок складається з основи, стін та даху. Якщо видалити будинок, то автоматично видалиться його основа, стіни і дах, але не його символ.

Об'єктно орієнтована технологія ґрунтується на об'єктній моделі. Основними принципами, на яких будуються об'єктні моделі, є: абстрагування, інкапсуляція, модульність, ієрархічність, типізація, паралелізм та збережуваність.

Абстракція виділяє істотні характеристики якогось об'єкта, які відрізняють його від усіх інших видів об'єктів і, таким чином, чітко визначає його концептуальні межі з точки зору спостерігача.

Інкапсуляція – це процес відділення один від одного елементів об'єкта, які визначають його устрій і поведінку; інкапсуляція слугує для того, щоб ізолювати контрактні зобов'язання абстракції від їх реалізації.

Модульність – це властивість системи, яка була розкладена на слабко внутрішньо пов'язані між собою модулі.

Ієрархія – це ранжування абстракцій, розташування їх за рівнями.

Типізація – це спосіб захисту від використання об'єктів одного класу замість іншого або, у крайньому разі, керування таким використанням.

Паралелізм – це властивість, яка відрізняє активні об'єкти від пасивних.

Збережуваність – здатність об'єкта існувати у часі, переживаючи процес, що його породив, і (або) у просторі, переміщаючись зі свого первісного адресного простору.

Об'єктно орієнтована модель бази геоданих має низку переваг:

– забезпечує комплексне подання реального світу;

– модель є інтуїтивною, оскільки в ній використовуються об'єкти,

що існують у реальному світі;

– інкапсуляція, що об'єднує атрибути і лінії поведінки об'єкта, робить можливим доступ до об'єкта за допомогою чітко визначеного набору методів та атрибутів без знання змісту об'єкта;

– забезпечує високий рівень цілісності даних (нові дані повинні відповідати правилам поведінки);

– забезпечує моделювання складних відношень між даними;

– підтримує множинні рівні генералізації, об'єднання і асоціації;

– добре інтегрується з методами імітаційного моделювання;

– має функцію підтримки версій для одночасного множинного оновлення;

– вимагає меншого кодування в ГІС-програмах, що означає меншого внесення помилок і більш низьку вартість підтримки.

В об'єктно орієнтованій моделі даних є також і деякі недоліки:

• комплексні моделі реального світу складніше розробляти і будувати;

• великі та комплексні моделі виконуються повільніше;

• ця модель залежить від ретельності опису явищ реального світу (що особливо складно у світі природи);

• аналіз об'єктно орієнтованих баз даних вимагає використання об'єктно орієнтованих мов програмування;

• деякі бізнес-додатки можуть не мати доступу до об'єктно орієнтованої бази даних або можливості передачі до неї даних.

Аналіз сучасного розвитку ГІС свідчить про те, що:

– системи з єдиним способом організації даних є радше винятком, ніж правилом;

– не існує єдиного й універсального способу ефективної просторової організації даних; у будь-якому разі необхідна повнота подає і зручність операцій досягаються використанням великої розмаїтості підходів;

– множинність підходів забезпечує гнучкість подання і використання даних;

– гнучкість системи досягається, крім того, моделюванням як самих просторових структур, так і відносин між ними та можливістю переходу від одних способів подання до інших.

Об'єктно орієнтована модель будується на концепції цифрового опису об'єктів місцевості в термінах сутностей і відношень між ними, де під об'єктом розуміють об'єкт місцевості або карти, метричний і семантичний опис якого не змінюється впродовж усього об'єкта. В об'єктно орієнтованих моделях топологічні відносини формуються на рівні об'єкта, а не на рівні графічних примітивів, однак це цілком можливе.

В даних моделях поняття тематичного шару не є головним, а одиницею управління даними слугує об'єкт, який описується в термінах системи класифікації і кодування об'єктів карти. Атрибутивна інформація зберігається у вигляді таблиць, пов'язаних з просторовою інформацією.

Як приклад на рис. 7.7 представлено цифрову модель фрагмента в об'єктно орієнтованому вигляді, яка містить опис полігональних об'єктів (ліс, ялина заввишки 25 м, діаметром 0,3 м, відстань між деревами 7 м; полігони посадженої берези (висота 20 м, діаметр 0,25 м, відстань 4 м); луг і лінійних об'єктів (лісова дорога та польова дорога по контуру лісу, лугу, посадки).

Треба відзначити, що багато ділянок метричного опису повторюються і неодноразово: AB тричі (ліс, луг, дорога), BC – тричі (посадка, луг, дорога), CD – двічі (посадка, дорога), BD – двічі (ліс, посадка).

Метричні описи повторюваних ділянок необов'язково однакові, оскільки при роздільному цифруванні кожного об'єкта вони можуть не збігатися.

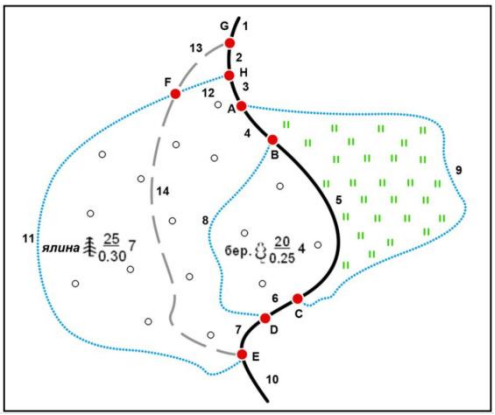


Рис. 7.7. Фрагмент картографічних даних

Крім того, кожний об’єкт описується поза зв’язком із сусідніми і при роботі з ним невідомо, що розташовано по сусідству. Це можна з’ясувати, "переглянувши" усю базу даних і виконавши аналіз за певним критерієм. Саме ця обставина змушує уводити правила шифрування (об’єкт ліворуч або праворуч) і просторово-логічні зв’язки (ПЛЗ), за допомогою яких можна з’ясувати, що знаходиться ліворуч (праворуч).

Об’єктно орієнтовані моделі, що ґрунтуються на принципах класифікації та кодування об’єктів, є менш універсальними в частині графічного опису зображень, але усувають усі проблеми в частині об’єднання

тематичної і картографічної інформації. Крім того, об’єктно орієнтовані моделі є більш перспективними, оскільки в практику все більше впроваджуються технології об’єктно орієнтованого програмування та об’єктних СКБД.

Підсумовуючи вищенаведене, слід зазначити, що основними мотивами вибору об’єктного підходу до побудови моделі збереження даних можна назвати природність подання у вигляді об’єктів моделі реального

світу, керованість розробкою проекту і відкритість його до модернізації, логічність і наочність у маніпуляціях із даними, а також необхідність застосування об’єктної бази даних для збереження неоднорідної інформації, що використовується в ГІС.

ЛЕКЦІЯ 8
МОДЕЛІ ОРГАНІЗАЦІЇ ДАНИХ

8.1. Моделі організації даних

Принципи організації даних визначають відповідні моделі організації даних:
1) георeляційна модель організації даних;
2) об’єктно орієнтована модель організації даних;
3) об’єктно реляційна модель організації даних.
Широкого поширення набули моделі даних ESRI:
а) георeляційна модель організації даних у вигляді моделі даних "Шейп-файл" і моделі даних "Покриття";
б) об’єктно орієнтована модель організації даних у вигляді моделі даних "База геоданих".
На рис. 8.1 представлено ієрархію моделей даних ESRI від загального верхнього рівня моделей географічних об’єктів до нижнього спеціального рівня організації даних.

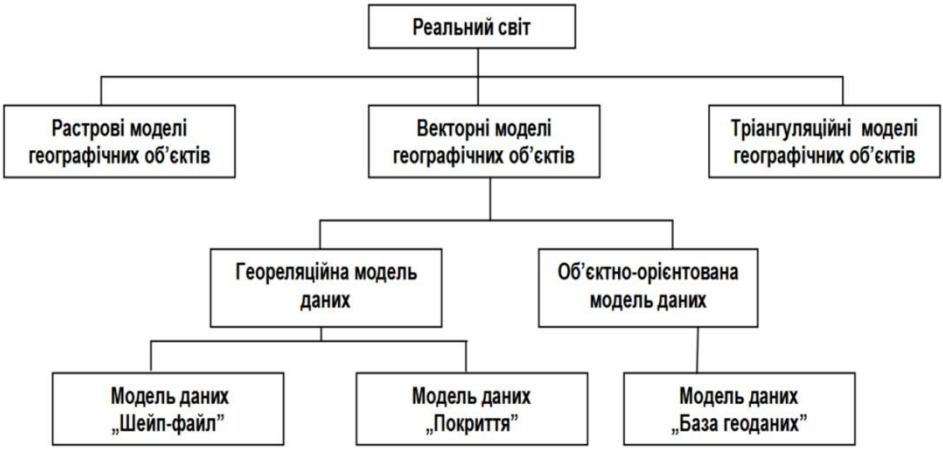


Рис. 8.1. Моделі даних ESRI

У реляційній моделі дані зберігаються як асоційовані набори таблиць, які логічно пов’язані за допомогою спільних атрибутів. Записи зберігаються як рядки таблиць, атрибути – у вигляді колонок. Кожна колонка містить атрибутивні дані тільки одного типу: дату, текстовий рядок, числові дані тощо. Таблиці стандартизуються для мінімізації дублювання.

ГІС містить два типи даних – просторові і семантичні (атрибутивні).

Просторові дані географічних об’єктів зберігаються в окремих таблицях просторових даних у вигляді послідовності координатних пар *x, y*.

Атрибутивні дані географічних об’єктів організовуються у таблиці атрибутивних даних. Кількість записів у таблицях атрибутів дорівнює кількості графічних об’єктів у двійкових файлах.

Відношення між географічними об’єктами здійснюється за допомогою топології, яка також подається відповідними таблицями.

У ГІС, орієнтованих на роботу з базами даних (зокрема, в ARC/INFO), успішно застосовується георелаційна модель даних (Georelational Data Model).

Сутність цієї моделі полягає у роздільному зберіганні значень координат і атрибутивних даних. Зв’язок між просторовою й атрибутивною інформацією здійснюється за допомогою ідентифікаторів. Ця модель заснована на геометричному типі об’єкта і відображає світу вигляді наборів точок, ліній і полігонів. Координати кожного об’єкта з унікальним ідентифікаційним номером зберігаються в двійкових "arc"-файлах. Атрибутивні значення й опис топології зберігаються в таблицях

реляційної СКБД (первісно в INFO таблиці). Записи пов’язані з геометрією за допомогою ідентифікаційного номера об’єкта (Identifier – ID). Модель географічних даних подає географічні об’єкти як набір взаємопов’язаних просторових і атрибутивних даних. При цьому ГІС здійснює узгоджене управління цілісною інформацією об’єктів, яка розподіляється між файловою системою і базою даних.

Таким чином, георелаційна модель даних визначається наступними

умовами:

1) записами в таблицях просторових даних, які відображають моделі географічних об’єктів (точками, лініями, полігонами) і записами в таблиці атрибутів;

2) зв’язок між географічним об’єктом і записом у таблиці атрибутів підтримується через унікальний номер – ідентифікатор об’єкта;

3) ідентифікатор зберігається у двох місцях: у файлах географічних об’єктів, що містить пари координат *X, Y,* і у відповідних записах таблиці атрибутів географічних об’єктів.

Важливою вимогою цієї моделі є так звана планарність виконання, яка означає, що плоска поверхня покривається безперервно об’єктами без перетинання.

Перевагою реляційних таблиць є те, що вони дозволяють спростити реальний світ і надають швидкі й достовірні відповіді на запити, які вони опрацьовують.

У георелаційних моделях використовується фіксований набір вбудованих типів даних, таких як числа, дати та текстові рядки.

Георелаційні моделі можуть вимагати складного прикладного програмування для результативного моделювання комплексних ситуацій реального світу. Складні змінні потребують великої кількості взаємозалежних таблиць. Однак георелаційна модель неспроможна моделювати усе розмаїття географічних об’єктів для конкретної області користувача.

В моделі об’єктів можуть бути геометричні структури, в яких один і той же об’єкт (наприклад, полігон) може відображати будинок або земельну ділянку, якщо обидві форми співпадають. Єдина відмінність полягає в їх атрибутах.

Геореляційна модель має наступні переваги:

– проста структура таблиць, які легко зчитувати;

– інтуїтивно простий інтерфейс користувача;

– наявність множини інструментів для кінцевих користувачів (наприклад, макросів і скриптів);

– простота зміни й додавання нових прив’язок, даних і записів;

– простота використання таблиць, що описують географічні елементи зі спільними атрибутами;

– можливість прив’язки таблиць, що описують топологію, необхідну для просторового аналізу;

– прямий доступ до даних, які забезпечують їх швидку й ефективну обробку;

– незалежність даних від додатка;

– наявність великих об’ємів просторових даних у цьому форматі.

До недоліків геореляційної моделі можна віднести:

• обмежене подання реального світу;

• обмежену гнучкість управління запитами і даними;

• повільний послідовний доступ;

• складність моделювання складних відносин даних, оскільки для цього часто необхідні кваліфіковані прикладні програмісти баз даних;

• необхідність подання складних відношень у вигляді процедур у кожній програмі, яка звертається до бази даних.

8.2. Збереження даних у моделі "Шейп-файл"

Модель даних "Шейп-файл" представляється цифровим форматом Shapefile. Формат "Шейп-файл" (Shapefile) – це цифровий векторний формат ESRI для збереження просторової і пов’язаної семантичної (атрибутивної) інформації про географічні об’єкти. Цей формат є непридатним для збереження топологічної інформації. Формат Shapefiles створений для ArcView GIS; він використовується в ARC/INFO, ArcGIS.

Формат Shapefile містить набір файлів з однаковою назвою, але з різним розширенням. Наприклад:

1. Forest.shp.
2. Forest.dbf.
3. Forest.shx.
4. Forest.sbn.
5. Forest.sbx.
6. Forest.ain.
7. Forest.aih.

Ці файли поділяються на обов’язкові та факультативні (додаткові).

Обов'язковими файлами є три файли з розширенням .shp, .shx, .dbf, оскільки вони містять базові дані. Файл форми з розширенням .shp (shape file) – це головний файл, який зберігає географічні об'єкти в його власному запису як список координатних пар *x, y*. Оскільки кожний географічний об'єкт (за винятком точкових) може містити різну кількість координатних пар, кожний запис у .shp файлі може бути різної довжини.

Записи змінної довжини вимагають певного часу для обробки і відображення на екрані, тому *.shp* файл індексується з *.shx* файлом. Файл індексу форми з розширенням *.shx* (shape index file) містить один запис фіксованої довжини для кожного запису в .shp файлі. Кожний запис в *.shx* файлі містить номер запису і довжину в байтах відповідного запису в *.shp* файлі. Оскільки усі записи в *.shx* файлі мають рівну довжину, вони можуть швидко зчитуватись і оброблятися аналогічно табличному пошуку для звітів у *.shp* файлі. Файл-індекс форми *(.shx*) прискорює креслення усіх просторових об'єктів у шейп-файлі. Можна прискорити запит і відбір індивідуальних просторових об'єктів створюючи просторовий і атрибутивний індекси.

Файл атрибутів із розширенням .dbf (dBASE file) зберігає атрибутивну інформацію про просторові об'єкти у *.shp* файлі як таблицю атрибутів у форматі dBASE. Таблиця атрибутів містить один запис для кожного просторового об'єкта у *.shp* файлі. Кожний запис атрибутів має відношення один до одного з відповідним записом форми.

Додаткові 2 файли просторових індексів ArcView GIS створює кожного разу, коли виконується просторове з'єднання або вибір теми темою. Файл просторового лотка з розширенням *.sbn* (spatial bin) розділяє ділянку, що містить географічні об'єкти *.shp* файлу на прямокутні ділянки, що отримали назву лотків. Кожний лоток містить запис кількості просторових об'єктів у .shp файлі, які потрапляють у його сферу. Коли виконується просторовий запит, записи в .sbn файлі зчитуються першими і розглядаються тільки просторові об'єкти, які перетинають лотки, зазначені запитом. Це значно прискорює обробку. Оскільки у лотки потрапляє різна кількість просторових об'єктів, то записи лотка змінюються за довжиною і потребують власного індексу.

Файл просторового індексу лотка з розширенням *.sbx* (spatial bin index) файл містить один запис фіксованої довжини для кожного запису у *.sbn* файлі. Кожний запис у *.sbx* файлі містить запис числа і довжини в байтах відповідного запису лотка у *.sbn* файлі. Оскільки записи у *.sbx* однакової довжини, вони можуть зчитуватись доволі швидко та опрацьовуватись аналогічно табличному пошуку у записах змінної довжини у *.sbn*.

Додаткові 2 файли індексу атрибуту ArcView GIS створює кожного разу, коли виконується зв'язок із таблицею. Файл індексу атрибута з розширенням *.ain* (attribute index) – файл містить один індекс для кожного поля, включеного у зв'язок, або такого, для якого створений індекс методом, описаним вище. Ці індекси покращують операції з даними типу з'єднання і зв'язку та прискорюють прості запити типу: [Назва] = "Луцьк".

Індекс атрибута не буде покращувати операції на запитах, що включають рядки відповідності типу [Назва] або порівняння [Кількість] <100000.

Файл заголовка індексу атрибута з розширенням *.aih* (attribute index header) – містить назву кожного поля, яке було індексоване. Він слугує як каталог до значень, що містяться в *.ain* файлі. До додаткових файлів відносяться ще три файли.

Файл системи координат із розширенням *.prj* (map projection) зберігає інформацію про систему координат і картографічні проекції.

Файл метаданих із розширенням *.xml* (extensible markup language) зберігає дані про дані в ArcInfo та ArcView 8.0 і більш пізніх версій для використання в Internet.

Файл легенди з розширенням (ArcView legend) зберігає символіку графічного відображення об'єктів.

Шейп-файли є простими, оскільки зберігають примітивні геометричні типи даних: точкові, лінійні та полігональні. Ці примітиви мають обмежене використання без будь-яких ознак для вказівки того, що вони представляють.

Таким чином, таблиця записів буде зберігати просторові об'єкти та атрибути для кожної примітивної форми в шейп-файлі. Форми (точкові, лінійні, полігональні), а також дані атрибутів можуть створювати нескінченну множину уявлень (представлень) про географічні дані. Представлення надає можливості для потужного і точного обчислення.

8.3. Збереження даних у моделі "Покриття"

Покриття (Coverage) – це геореляційна модель, яка має векторний топологічний формат даних. Інститутом ESRI розроблені дві версії файлів покриття: для PC ARC\INFO у 1981 р., для версій ArcGIS з 2000 р.

В математиці термін "покриття множини *X*" – це сімейство підмножин цієї множини, об'єднання яких є X, або сімейство підмножин простору, в якому розташоване X і яке містить X. Елементи покриття містять у собі повну інформацію про локальну будову простору. Поняття "покриття" розглядається у контексті загальної топології.

Покриття містить просторові й атрибутивні дані географічних об'єктів. Покриття використовує набір класів просторових об'єктів для подання географічних об'єктів (рис. 8.2).

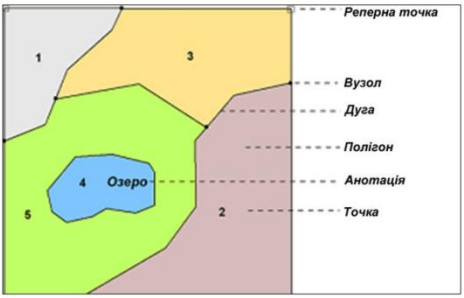


Рис. 8.2. Класи просторових об’єктів у покритті

Модель даних "Покриття" використовує наступні класи просторових об’єктів:

Точка (*Point*) – використовується для подання точкових просторових об’єктів або користувацьких ідентифікаторів ID полігонів. Точка визначається координатною парою *x,y*.

Дуга (*Arc*) – використовується для подання лінійних просторових

об’єктів або меж полігонів. Дуга визначається послідовністю координатних пар x,y початкового вузла, проміжних вершин, кінцевого вузла.

Дуги топологічно зв’язуються через їх кінцеві точки (вузли). Один лінійний об’єкт може бути утворений багатьма дугами.

Вузол (Node) – представляє кінцеві точки дуг або перетинання лінійних об’єктів. Вузол має унікальний ідентифікатор. Вузол може топологічно пов’язуватись із набором дуг, які з’єднані між собою.

Шлях (Route) – лінійний просторовий об’єкт, що складає одну чи декілька дуг або частин дуг.

Секція (Section) – дуга або частина дуги, яка використовується для визначення шляху чи створення шляхових блоків.

Полігон (Polygon) – представляє площинні об’єкти. Полігони топологічно визначаються серією дуг, які формують їх межі, включаючи дуги, що визначають острови усередині. Користувацькі ідентифікатори ID полігонів подаються точками усередині меж.

Регіон (Region) – сукупність полігонів, що представляють географічний об’єкт.

Анотація (Annotation) – текст, що використовується для позначення об’єктів. Анотації не мають топологічних зв’язків з іншими об’єктами та не використовуються для аналітичних цілей.

Реперна точка (Tic) – реєстраційна точка, яка визначає положення відомої точки на земній поверхні, для якої відомі координати

реального земного простору. Реперні точки дозволяють реєструвати та трансформувати координати покриття. Рекомендована кількість реперних точок – 4 і більше.

Охоплення покриття (Coverage Extent) – мінімальний прямокутник, що обмежує покриття, яке представляє територіальне охоплення покриття. Охоплення покриття визначається граничними координатами Xmax, Xmin, Ymax, Ymin його елементів.

Описові дані для класів просторових об’єктів зберігаються у відповідних таблицях атрибутів. Пов’язування просторових об’єктів й атрибутів забезпечується наступними базовими принципами:

• просторові об’єкти у покритті існують у відношенні "один-до-одного" з відповідними записами у таблиці атрибутів просторових об’єктів;

• ArcGIS підтримує зв’язок між просторовими об’єктами й атрибутами за допомогою ідентифікатора, призначеного кожному об’єктові;

• порядковий номер просторового об’єкта фізично зберігається у двох місцях покриття:

• у файлах, що містять просторові дані для кожного просторового об’єкта (координатні пари);

• відповідним записом у таблиці атрибутів просторових об’єктів.

ArcGIS автоматично створює та підтримує ці зв’язки. Набір класів у покритті варіюється залежно від географічних об’єктів, які воно представляє. Покриття може містити:

1) набір точок, які представляють географічні об’єкти, й асоційовані таблиці атрибутів, які описують ці точкові об’єкти;

2) набір вузлів і дуг, які представляють лінійні просторові об’єкти, та асоційовані таблиці атрибутів, які описують ці лінійні об’єкти;

3) набір вузлів і дуг, які оточують географічні області (полігони), та асоційовані таблиці атрибутів, які описують ці області;

4) комбінацію наборів 1) і 2);

5) комбінацію наборів 2) і 3).

Покриття також містить інші елементи (анотації та реперні точки).

Наведене свідчить, що дуга і вузол є головними "будівельними" елементами покриття. Ключове значення дуги Arc використовується у назві програмних продуктів ArcInfo, ArcGIS.

Покриття зберігається як директорія, у якій кожний клас просторових об’єктів зберігається як набір файлів. Наприклад, покриття "дороги" є лінійним покриттям, що містить файл дуг (Arc), файл анотацій (Annotation)

для дуг, файл реперних точок (Tic).

Просторові дані зберігаються у двійкових файлах, а атрибутивні та топологічні дані – в таблицях INFO. Покриття також може мати асоційовані файли.

Топологія має відношення до способу, згідно з яким лінійні дані зберігаються і співвідносяться. Топологія реєструє просторові відношення між дугами й полігонами в покритті. Кожна дуга має "від вузла", "до вузла", полігон ліворуч і праворуч, унікальний ідентифікатор ID-дуги та внутрішній порядковий номер. Групи дуг, які формують замкнені форми (полігони), асоціюються з унікальною міткою. Збереження даних таким способом дозволяє системі визначати, які полігони суміжні, які дуги формують полігони, як далеко стоять центри дуг і полігонів один від одного тощо.

Треба мати на увазі, що топологія не створюється автоматично. Користувач використовує спеціальні команди для створення топології.

8.4. Об’єктно орієнтована модель даних "База геоданих"

8.4.1. Визначення бази геоданих

Починаючи з 2000 р., у ArcGIS представлено новий підхід до збереження й подання географічних даних – об’єктно орієнтована модель даних, що отримала назву "База геоданих" – БГД (Geographic Database –

GDB).

База геоданих – це сукупність географічних наборів даних різних типів, що використовуються в ArcGIS і які зберігаються в загальних папках файлової системи або в реляційній базі даних.

Модель "База геоданих" ґрунтується на принципах реляційних таблиць і використовує персональну базу даних Microsoft Access або багатокористувальницьку реляційну базу даних, таку як Oracle, Microsoft SQL Server, PostgreSQL, Informix або IBM DB2.

Ключовою концепцією бази геоданих є набір даних. База геоданих містить три основні типи наборів даних:

– класи просторових об’єктів (Feature classes);

– растрові набори даних (Raster datasets);

– непросторові таблиці (Tables).

Ці набори даних додають до бази геоданих розширені можливості (шляхом створення топології мережі або підтипів) для моделювання поведінки, збереження цілісності даних і роботи з одним з найважливіших наборів просторових відношень.

База геоданих забезпечує:

• доступ і управління великими обсягами просторових даних, що зберігаються у файлах і базі даних;

• обробку великих і різноманітних типів даних та інших об’єктів;

• застосування складних правил і відношень в "інтелектуальних" ГІС, що безпосередньо моделюють реальність.

8.4.2. Об’єктно орієнтована векторна модель даних

Модель бази геоданих підтримує об’єктно орієнтовану векторну модель даних. У цій моделі сутності представлені як об’єкти з властивостями, поведінкою та відношеннями. Підтримка різних типів географічних об’єктів вбудована в систему. До цих типів об’єктів відносяться прості об’єкти, географічні об’єкти, мережеві просторові об’єкти, анотації просторових об’єктів тощо, більш спеціалізовані типи просторових об’єктів.

Модель дозволяє визначити взаємовідношення між об’єктами, а також правила для підтримки цілісності посилальних даних між об’єктами.

Об’єктно орієнтована модель даних розглядає географічні об’єкти реального світу як об’єкти бази даних. Об’єктами подаються просторові об’єкти, наприклад, дорога, будинок, земельна ділянка. Об’єктами можуть бути шар доріг, система координат шару доріг тощо.

Модель бази геоданих визначає загальну модель для географічної (просторової) інформації. Це типова модель, яка може бути використана для визначення та роботи широким колом різних користувачів або додатків за конкретними моделями.

Перевагами бази геоданих є надання таких можливостей:

• централізовано зберігати географічні дані та керувати ними в одній реляційній СКБД;

• моделювати поведінку просторових об’єктів;

• застосовувати складні правила та відношення до даних;

• підтримувати цілісність просторових даних у несуперечливій, точній базі даних;

• робота в рамках багатокористувацького доступу та редагування середовища;

• масштабування створених рішень;

• інтеграція просторових даних з іншими базами даних;

• підтримка користувацьких функцій і поведінки.

8.4.3. Засоби надання інтелектуальних властивостей просторовим об’єктам

У моделі даних "покриття" користувач може описувати тільки геометрію об’єкта та його характеристику, де описана його поведінка за допомогою додаткових програм, що створені користувачем. Це обмежує можливості аналізу й обробки, наприклад, аналіз різних типів об’єктів у реальному часі.

Модель даних бази геоданих володіє істотною перевагою – можливістю будувати інтелектуальну модель просторової системи [50].

За допомогою моделі даних бази геоданих користувач може створювати більш змістовні об’єкти з новими якостями (інтелектуальні об’єкти) і тим самим моделювати об’єкти навколишнього світу.

Користувач працює не просто зі звичайними точками, лініями та полігонами, інформація про які зберігається в таблицях. У БГД користувач може оперувати такими поняттями, як об’єкти навколишнього світу, встановлювати та налаштовувати властивості й взаємовідношення об’єктів. Наприклад, замість точок можна працювати з трансформаторами, а замість ліній – з трубами. При цьому кожній трубі буде встановлено правило: через який перехідник вона з’єднується з іншою трубою.

Користувач може задавати поведінку окремих об’єктів, визначати взаємостосунки класів об’єктів, створювати правила та застосовувати топологічні моделі високого рівня без програмування.

Модель "База геоданих" має засоби створення більш змістовних просторових об’єктів, якими моделюється поведінка, підтримується цілісність даних і робота з просторовими відношеннями.

Такими засобами є передусім топологія, підтипи класів просторових і непросторових об’єктів, домени атрибутів, відношення, правила зв’язності тощо.

8.4.4. Топологія в базі геоданих

В базі геоданих топологія – це механізм, який визначає як точкові, лінійні, полігональні просторові об’єкти використовують загальну геометрію.

Наприклад, осі вулиць і поштових ділянок мають загальну геометрію, а суміжні полігони ґрунтів мають загальні кордони.

Топологія визначає та забезпечує правила цілісності даних (наприклад, не повинно бути щілин між полігонами). Вона підтримує запити топологічних відношень і переміщень (наприклад, переміщення просторових об'єктів суміжності або зв'язності), підтримує складні інструменти редагування, а також дозволяє створювати просторові об'єкти із неструктурованих геометрій (наприклад, створення полігонів з ліній).

У ArcGIS топологія включає компоненти:

1) топологічні моделі даних, які використовують відкритий формат для простих просторових об'єктів, правила топології та топологічно інтегровані координати для просторових об'єктів із загальною геометрією;

2) топологічні шари, які використовуються для відображення топологічних взаємовідношень, помилок і винятків;

3) комплекс інструментів геообробки для створення, аналізу, управління та перевірки топології;

4) досконалі програмні алгоритми для аналізу й знаходження топологічних елементів у класах просторових об'єктів (точкових, лінійних і полігональних);

5) потужні засоби редагування й автоматизації даних, які використовуються для створення, підтримки й перевірки топологічної цілісності та подання групових просторових об'єктів.

Властивості топології у базі геоданих.

У базі геоданих для кожної топології визначаються такі властивості:

– ім'я топології, яка буде створена;

– кластерний допуск, що визначає мінімально припустиму відстань між вершинами об'єктів;

– список класів просторових об'єктів, які братимуть участь у топології;

– ранги відносної точності координат для кожного класу просторових об'єктів;

– список правил топології.

Правила топології.

Топологія реалізується у вигляді набору правил цілісності, які визначають поведінку просторово взаємопозв'язаних географічних об'єктів і об'єктних класів. Ці правила дозволяють моделювати у базі геоданих такі просторові відношення, як зв'язність (чи пов'язані лінії вулично-дорожньої мережі?) і суміжність (чи існує щілина між двома полігонами?). Правила топології визначають припустимі просторові відношення між просторовими об'єктами.

Правила використовуються для контролю топологічних відношень між просторовими об'єктами усередині одного класу просторових об'єктів, між просторовими об'єктами у різних класах або між підтипами просторових об'єктів.

8.4.5. Підтипи

Підтипи (Subtypes) – це підмножина просторових об’єктів у класі просторових об’єктів або об’єкти в таблиці, які мають однакові атрибути.

Підтипи використовуються для створення розширених можливостей бази геоданих.

1) Підтипи дозволяють підвищити ефективність бази геоданих. Підтипи використовуються як метод класифікації даних, поділ класів просторових об’єктів або таблиць на логічні групи, що ґрунтуються на значеннях атрибутів. Наприклад, дороги у класі просторових об’єктів доріг можна розбити на три підтипи: дороги місцевого значення, дороги обласного значення і дороги загальнодержавного рівня. Подання різних об’єктів навколишнього світу як підмножини просторових об’єктів у даному класі просторових об’єктів замінює необхідність створення нового класу просторових об’єктів для кожного об’єкта.

2) Підтипи дозволяють керувати значеннями атрибутів, у тому числі:

– встановити значення за замовчуванням, яке буде автоматично застосовуватись при створенні нових просторових об’єктів. Наприклад, підтип вулиць місцевого значення може бути створений і визначений таким чином, що коли цей підтип вулиці буде доданий до класу просторових об’єктів вулиць, його атрибут максимальної швидкості буде автоматично встановлений на 60 км/год;

– застосувати домен діапазонів або домен кодованих значень до просторових об’єктів, щоб обмежити введення інформації згідно зі встановленим набором значень. Наприклад, у водопровідній мережі підтип магістральних водопроводів може мати домен кодованих значень для будівельних матеріалів, який обмежує їх виготовлення з заліза, чавуну або міді.

3) Підтипи дозволяють розширити правила поведінки, зокрема:

– створити правила з’єднання між іншими підтипами та класами просторових об’єктів для збереження цілісності мережі. Наприклад, у водопровідній мережі кінці труб різного діаметра можуть бути з’єднані спеціальним перехідним елементом;

– створити правила топології між іншими підтипами та класами просторових об’єктів, що знаходяться в топології. Наприклад, можна внести вимогу про те, що просторові об’єкти вулиць повинні бути пов’язані з іншими просторовими об’єктами вулиць з обох боків, за винятком підтипів тупикових вулиць; – розробити правила відношень між іншими підтипами, таблицями та класами просторових об’єктів. Наприклад, в електричній мережі можна створити правила відношень між підтипами, які полягають у тому, що сталеві стовпи підтримують клас трансформаторів B, а дерев’яні стовпи підтримують клас трансформаторів C; – створити користувацькі правила між просторовими об’єктами за допомогою написання коду.

8.4.6 Домени

Два механізми *Домени* та *Підтипи* використовуються для організації даних таким чином, щоб операції управління, відображення та редагування даними стали більш ефективними при підтримці цілісності атрибутів.

Домен – це декларація (фіксований список) припустимих значень атрибута. Кожен раз, коли домен асоціюється з полем атрибутів, тільки значення з домену будуть правильними для цього поля. Інакше кажучи, поле не прийме ніякого значення, якщо воно буде не з домену.

У базі геоданих домен атрибутів є механізмом посилення цілісності даних. Домени атрибутів визначають, які значення є припустимими у полі класу просторових об'єктів або таблиці непросторових атрибутів.

Якщо просторові або непросторові об'єкти згруповані у *Підтипи*, різні домени атрибутів можуть асоціюватися з кожним підтипом.

База геоданих використовує два типи доменів атрибутів – домени діапазонів і домени кодів.

Домени діапазонів створюються на інтервальних даних.

Домен діапазонів задає допустимий діапазон значень для числового атрибута.

При створенні ряду доменів необхідно ввести мінімальне та максимальне значення. Наприклад, для підтипу ліній газопостачання низького тиску задається діапазон діаметрів труб від 40 мм до 270 мм.

Домени кодів створюються на базі номінальних даних. Домен кодованих значень може застосовуватись до будь-якого типу атрибутів – текстових, цифрових, дат тощо. Домен кодованих значень визначає правильний набір значень для атрибута. Наприклад, затверджений міською радою стандарт на назву вулиць, який є доменом кодів, включає у тому числі значення атрибута "Рокосовського вул."; поле назв вулиць не прийме значення атрибута "Рокосовська вул.".

8.4.7. Відношення та класи відношень

Різні види географічних і негеографічних сутностей можуть бути пов'язані між собою відношеннями. Відношення можуть існувати між:

– географічними сутностями, наприклад, будинок може бути пов'язаний із земельною ділянкою;

– географічними сутностями та негеографічними сутностями, наприклад, земельна ділянка може бути пов'язана з власником;

– негеографічними сутностями, наприклад, власник земельної ділянки може бути пов'язаний з податковим кодом.

При поданні зв'язків між географічними об'єктами необхідно моделювати просторові відношення між просторовими об'єктами.

Зв'язування (Relating) – операція, яка встановлює часові зв'язки між записами в двох таблицях, в обох використовуючи загальний ключ.

Зв'язування реалізують типи відношень "один-до-одного" (one-to-one), "один-до-багатьох" (one-to-many) і "багато-до-багатьох" (many-to-many).

З’єднання (Joining) – операція зв’язування та фізичного з’єднання двох таблиць атрибутів, використовуючи їх спільні елементи, які існують у двох таблицях. З’єднання використовується для того, щоб доповнити поля однієї таблиці до полів іншої таблиці за допомогою спільних атрибутів або полів. З’єднання реалізують типи відношень "один-до-одного" і "багато-доодного".

Відношення "один-до-одного" та "один-до-багатьох" не вимагають нової таблиці у базі геоданих. Зв’язування і з’єднання використовуються для створення даних, їх вивчення та аналізу. У базі геоданих відношення між об’єктами зберігаються в класах відношень.

Клас відношень (Relationship class) – це елемент бази геоданих, який зберігає інформацію про відношення. Класи відношень керують асоціаціями між об’єктами в одному класі (класі просторових об’єктів або таблиці) й об’єктами в іншому класі.

Класи відношень забезпечують множину ефективних можливостей, які відсутні в операціях зв’язування та з’єднання.

1) Класи відношень допомагають забезпечувати цілісність посилальних даних. Клас відношень може бути налаштовано таким чином, що при зміні об’єкта пов’язані об’єкти оновлюються автоматично. Це може включати фізичне переміщення пов’язаних просторових об’єктів, видалення пов’язаних об’єктів або оновлення атрибута. Наприклад, можна створити такі відношення, при яких при переміщенні опори електромережі разом з нею переміщуються прикріплені трансформатори та інше обладнання. Встановлюючи правила, клас відношень може обмежити тип

відношень. Наприклад, одна опора може підтримувати не більше трьох

трансформаторів. Сталева опора може підтримувати трансформатори

класу B, але не трансформатори класу C.

2) Класи відношень полегшують редагування, допомагаючи знизити витрати на технічне обслуговування. Забезпечуючи автоматичне оновлення для зв’язаних об’єктів, клас відношень може позбавити від потреби виконання додаткових операцій редагування. Класи відношень полегшують користувачам доступ до об’єктів у процесі редагування. Можна обрати один об’єкт, а потім знайти всі пов’язані об’єкти, використовуючи діалогове вікно атрибутів або таблиці. Вийшовши на відповідний об’єкт, є можливість редагувати його атрибути.

3) Класи відношень дозволяють виконувати запити до пов’язаних просторових об’єктів та записів, виконувати аналіз і формувати звіти з атрибутами з класу відношень. Клас відношень може мати атрибути. Клас відношень, який має атрибути, зберігається у таблиці бази геоданих. Класи відношень мають такі характеристики: ім’я, атрибути відношення, клас-джерело та клас-адресат, первинні та зовнішні ключі, тип відношення, потужність, напрямок повідомлення інформації.

Усі відношення в класі відношень зв’язують об’єкти з одного класу джерела (origin class) з об’єктами з одного класу-адресата (destination class).

Таблиця класу відношень має пару зовнішніх ключів (foreign key – FK), один з яких відноситься до класу-джерела, а інший – до класу-адресата.

Класи об’єктів мають внутрішні ключі (primary key – PK), один з яких відноситься до класу-джерела, а інший – до класу-адресата.

Клас відношень має мітку прямого напрямку (forward path label) і мітку зворотного (backward path label). Приклади міток напрямку: "керує", "керується за допомогою". Потужність відношення (cardinalities) визначає кількість об’єктів у класі-джерелі, які можуть відноситися до числа об’єктів у класіадресата. Відношення у базі геоданих можуть мати одне з трьох значень потужності: "один-до-одного", "один-до-багатьох" або "багато-до-багатьох".

База геоданих підтримує два типи відношень – прості та складені.

Просте відношення (simple relationship) – це рівноправне відношення, при якому пов’язані об’єкти можуть існувати незалежно один від одного.

Складене відношення (composite relationship) – це відношення "один-до-багатьох", при якому об’єкти з класу-адресата не можуть існувати незалежно від об’єктів з класу-джерела. Коли видаляється джерело, відповідні об’єкти з класу-адресата також видаляються.

8.4.8. Елементи об’єктно орієнтованої моделі "База геоданих"

База геоданих містить три основні типи наборів даних незалежно від системи їх використання:

• таблиці (Tables);

• класи просторових об’єктів (Feature classes);

• набори растрових даних (Raster datasets).

Ці набори даних можна розглядати як універсальну відправну точку для розробки бази геоданих.

База геоданих має низку додаткових елементів і типів наборів даних, які можуть бути використані для моделювання складної поведінки, підвищення цілісності даних, розширення можливостей бази геоданих в управлінні даними.

Ці типи наборів даних у базі геоданих, а також інші елементи бази геоданих зберігаються у таблицях.

8.4.9. Таблиці

У базі геоданих управління атрибутами здійснюється в таблицях, які ґрунтуються на серії простих істотних концептів реляційних даних:

– таблиці містять ряди;

– усі ряди таблиці мають однакові колонки;

– кожна колонка має тип даних;

– серія реляційних функцій і операторів доступна для операцій на таблицях та їх елементах даних.

Таблиці та відношення відіграють ключову роль, бо вони знаходяться у додатках традиційних баз даних. Користувачі за допомогою таблиць виконують дуже багато традиційних табличних і реляційних операцій.

Таблиці (Tables) забезпечують описовою інформацією географічні об’єкти, растри і традиційні таблиці атрибутів у базі геоданих. Рядки у таблиці можуть використовуватися для збереження всіх властивостей географічних об’єктів. Вони включають збереження й управління геометрією просторових об’єктів у колонці (стовпчику) Shape.

Для моделювання відношень і поведінки таблиці можуть використовуватися спільно з елементами, які надають додаткові ефективні можливості базі геоданих. База геоданих включає наступні елементи, що утворюють розширення таблиць (Extending tables).

Домени атрибутів (Attribute Domains) – представляють списки припустимих значень або діапазон допустимих значень атрибутів стовпчиків. Домени використовуються для забезпечення цілісності значень атрибутів і класифікації даних.

Класи відношень (Relationship Classes) – створюють відношення між двома таблицями, використовуючи загальний ключ. Класи відношень визначають рядки у другій таблиці, що відповідають обраним рядкам у першій таблиці.

Підтипи (Subtypes) – керують наборами атрибутів підкласів у одній таблиці. Підтипи часто використовуються в таблицях класів просторових об’єктів для керування різною поведінкою в підмножинах того ж типу просторових об’єктів.

Управління версіями (Versioning) – керують довгими транзакціями

оновлення даних, історичними архівами та багатокористувацьким редагуванням.

8.4.10. Класи просторових об’єктів

Класи просторових об’єктів (Feature Classes) – це гомогенні зібрання просторових об’єктів з однаковим просторовим поданням, зі спільною системою координат і набором атрибутів, що зберігаються у таблиці бази геоданих.

У базі геоданих основні класи просторових об’єктів мають 7 типів:

1. Точки (Points) – використовуються для подання географічних об’єктів, розмірами яких можна знехтувати для даного завдання. Точками характеризують місце розташування географічних об’єктів.

2. Лінії (Lines) – використовуються для подання географічних об’єктів, які мають довжину, але не мають площі. Лініями подають форму та місце розташування географічних об’єктів.

3. Полігони (Polygons) – набір різносторонніх площинних просторових об’єктів, які представляють форму та місце розташування однорідних типів просторових об’єктів.

4. Анотація (Annotation) – текст карти, що містить властивості подання тексту.

5. Розміри (Dimensions) – спеціальний тип анотації, який показує певну довжину або відстань. Наприклад, довжина сторони будинку або відстань між точками.

6. Мультиточки (Multipoints) – використовуються для подання просторових об’єктів, які утворені більше ніж однією точкою із загальними атрибутами.

7. Мультифрагменти (Multipatches) – використовуються для подання зовнішньої поверхні або оболонки просторових об’єктів, які займають окрему частину або об’єм у тривимірному просторі. Мультифрагменти включають планові 3D-кільця і трикутники, які використовуються у комбінації для моделювання тривимірної оболонки. Мультифрагменти можуть використовуватися для подання будь-яких об’єктів – від простих (сфери або куби) до складених об’єктів (ізоповерхні або будинки).

8.4.11. Розширення класів просторових об’єктів

Для моделювання просторових відношень і поведінки класи просторових об’єктів можуть використовуватися спільно з елементами, які підвищують ефективність бази геоданих. База геоданих містить наступні елементи, які утворюють розширення класів просторових об’єктів (Extending feature classes).

1. Набір класів просторових об’єктів (Feature Dataset) – зібрання класів просторових об’єктів, які мають загальну систему координат.

2. Підтипи (Subtypes) – використовуються з таблицями класу просторових об’єктів для управління різною поведінкою у підмножинах одного типу просторових об’єктів. Підтипи керують набором підкласів просторових об’єктів в одному класі просторових об’єктів.

3. Домени атрибутів (Attribute Domains) – визначають список припустимих значень або діапазони допустимих значень атрибутів стовпчик.

4. Домени використовуються для забезпечення цілісності значень атрибутів і для класифікації даних.

5. Класи відношень (Relationship Classes) – створюють відношення між класами просторових об’єктів та іншими таблицями з використанням загального ключа.

6. Топологія (Topology) – моделює загальну геометрію просторових об’єктів.

7. Мережевий набір даних (Network Dataset) – моделює транспортну зв’язність і потоки.

8. Геометричні мережі (Geometric Network) – моделюють комунальні мережі й трасування.

9. Набір даних місцевості (Terrain Dataset) – моделює мережу нерегулярних трикутників (TIN) і керує великими лідарними та гідролокаційними колекціями точок.

10. Адресний локатор (Address Locator) – адресне геокодування.

11. Набір даних кадастрового матеріалу (Cadastral Fabric Dataset) – інтегрує та підтримує геодезичну інформацію для підрозділів і планів земельних ділянок як частина моделі даних безперервного кадастрового матеріалу у базі геоданих. Набір даних кадастрового матеріалу вносить додаткові поліпшення точності для матеріалу земельної ділянки, як тільки буде уведено новий підрозділ планів і опис земельної ділянки.

12. Лінійне моделювання (Linear Referencing) – визначає місце розташування подій уздовж лінійних просторових об'єктів з вимірами.

13. Картографічні подання (Cartographic Representations) – керує множиною картографічних подань і правилами покращеного картографічного креслення.

14. Управління версіями (Versioning) – керує деякими ключовими потоками робіт у ГІС для управління даними.

8.4.12. Розширення растрів

Растрові дані – це географічні об'єкти, одержані шляхом ділення території на дискретні квадратні або прямокутні комірки. Кожна комірка має значення, яке використовується для подання характеристик цього місця.

Растрові дані зазвичай використовуються для управління та подання зображень, цифрових моделей рельєфу, а також низки інших явищ.

Нерідко растри використовуються як спосіб подання точкових, лінійних і полігональних просторових об'єктів.

Растри цікаві з двох причин: по-перше, вони можуть бути використані для подання усієї географічної інформації (просторових об'єктів, зображень і поверхонь); по-друге, вони мають багатий набір аналітичних операторів геопроцесингу.

Растри набули найбільшого поширення у геоінформаційних додатках. База геоданих може керувати растрами як індивідуальними наборами даних, як логічною колекцією наборів даних, а також атрибутами зображень у таблицях.

Геопроцесинг – обробка даних про землю.

Растри можуть використовуватися спільно з елементами, які підвищують ефективність бази геоданих. База геоданих містить наступні елементи, які утворюють розширення растрів (Extending rasters):

1. Набір растрових даних (Raster Datasets) – керує дуже великими безперервними наборами даних зображень і мозаїками зображень.

2. Каталоги растрів (Raster Catalogs) – використовуються для низки цілей: керування шаром аркушів зображень, де кожний аркуш є зображенням; керування серією зображень у СКБД; керування серією часових растрів.

3. Колонки атрибутів растрів у таблицях (Raster Attribute Columns in Tables) – зберігають зображення або скановані документи як атрибути у таблиці.

4. Визначення сервісу зображення (Image Service Defi nition) – сервіси публікації зображень для колекції растрових даних у полі на диску, у каталозі зображень і базі геоданих.

8.5. Типи баз геоданих

Існує два типи баз геоданих – персональні та багатокористувацькі БГД. У персональних базах геоданих дані зберігаються локально на окремому

комп'ютері. Локальна версія бази геоданих може зберігатися і в комп'ютері у вигляді файлу Microsoft Access (*.mdb). Використання персональних баз геоданих доцільно при малих об'ємах даних і коли не потрібно використовувати багатокористувацькі сеанси доступу й редагування.

У багатокористувацьких базах геоданих дані зберігаються на сервері, використовуючи як сховище даних відомі комерційні РСКБД, такі як IBM DB2, Informix, Oracle, SQL Server, PostgreSQL. Багатокористувацький варіант бази геоданих доцільно використовувати для великих організацій, коли об'єми даних перевищують позначку з шістьма нулями.

При менших обсягах користувач може обрати як початкову основу локальні версії баз геоданих. У міру зростання обсягу накопичених даних користувач може поступово здійснювати перехід на багатокористувацьку корпоративну РСКБД. Цей перехід не потребує кардинальної реорганізації виробничого геоінформаційного процесу, що є досить істотним фактором. При цьому локальна база геоданих збереже усю структуру даних й успадковує правила та властивості усіх об'єктів, що задані у базі геоданих на сервері.

Багатокористувацькі бази геоданих дозволяють редагувати одні й ті ж географічні дані багатьом користувачам одночасно на декількох робочих місцях. Наприклад, в організації, що складається з багатьох відділів, де кожний з відділів повинен брати участь у робочому процесі, причому бажано, а іноді й необхідно, щоб усі ці відділи працювали спільно й одночасно – у реальному часі. Для цього потрібно правильно організувати робочий процес і завжди мати під рукою зручний інструмент для управління цим процесом. При такій постановці робочого процесу його учасники повинні оперувати свого роду версіями (копіями) загальної бази даних, з якими

вони взаємодіють.

Версія представляє моментальну копію усієї бази геоданих. Вона містить усі набори даних у базі геоданих. Версія ізолює роботу користувачів за допомогою множини сесій редагування, дозволяючи користувачам редагувати без блокування просторові об'єкти у виконуваній версії.

Ефективність описаних вище можливостей, які забезпечує багатокористувацька база геоданих, очевидна.

8.9. Вимоги до баз геопросторових даних

Функції цієї групи є переважно загальнотехнічними і застосовуються при створенні будь-яких баз даних й управлінні ними. Специфіка функцій цієї групи виявляється в організації даних позиціонування з урахуванням координатних систем, просторових моделей і масштабів картографування територій. Найбільш важливими є такі функції:

1) задання внутрішньої для ГІС моделі даних, яка забезпечує опис об'єктів довільного типу;

2) забезпечення багаторівневого (за масштабами) представлення території з узгодженням координатних систем;

3) введення даних про якість інформації, яка включає походження, точність даних, детальність і повноту (у тому числі пооб'єктно);

4) введення й організацію растрових даних (фільтрація, зшивка) за аркушами або за ділянками території;

5) введення й організація векторних даних (зведення, звірення, зшивка – інтерактивне або автоматичне з'єднання геометрично суміжних об'єктів, які перекривають або розділяють один одного, кліпування, додавання і / або видалення точок) за аркушами або за ділянками території;

6) введення та зміну атрибутивних даних (зміна ідентифікаторів, об'єднання кодів);

7) забезпечення організації масивів даних за типом локалізації, темою, класами об'єктів;

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. **Геоінформаційні** системи і бази даних : монографія / В. І. Зацерковний, В. Г. Бурачек, О. О. Железняк, А. О. Терещенко. – Ніжин : НДУ ім. М. Гоголя, 2014. – 492 с
2. Іщук О. О. Просторовий аналіз в ГІС : навч. посіб. / О. О. Іщук, М. М. Коржнев, О. Є. Кошляков ; за ред. акад. Д. М. Гродзинського. – К. : ВПЦ "Київський університет", 2003. – 195 с.
3. Кліменко І. В. Технології електронного урядування : навч. посіб. / І. В. Кліменко, К. О. Линьов. – К. : ДУС, 2006. – 225 с.
4. Ладичук Д. О. Бази геоінформаційних даних / Д. О. Ладичук, В. І. Пічура. – Херсон : ХДУ, 2007. – 103 с
5. Костріков С. В. Дослідження самоорганізації флювального рельєфу на засадах синергетичної парадигми сучасного природознавства: монографія / С. В. Костріков, І. Г. Черваньов. – Х. : ХНУ імені В. Н. Каразіна, 2010. – 144 с
6. Морозов В. В. ГІС в управлінні водними і земельними ресурсами : навч. посіб. / В. В. Морозов. – Херсон : Вид-во ХДУ, 2006. – 88 с
7. Морозов В. В. Геоінформаційні технології в агросфері /В. В. Морозов, К. С. Лисогоров, Н. М. Шпоринська. – Херсон : ХДУ,2007. – 223 с.
8. Морозов В. В. Моделювання та прогнозування для проектів геоінформаційних систем / В. В. Морозов, С. Я. Плоткін, М. Г. Поляковта ін. – Херсон : ХДУ, 2007. – 328 с
9. Світличний О. О. Основи ГІС / О. О. Світличний, С. В. Плотницький. – Суми : Університетська книга, 2006. – 296 с.
10. Суховірський Б. І. Географічні інформаційні системи : навч. посіб. / Б. І. Суховірський. – Чернігів : ЧДІЕУ, 2000. – 197 с.
11. Суховірський Б. І. Геоінформаційні системи і технології в регіональному розвитку / Б. І. Суховірський. – К. : Знання України, 2002. – 210 с.
12. Ушкаренко В. О. Системи управління базами даних ГІС для моніторингу ґрунтів / В. О. Ушкаренко, В. В. Морозов, О. В. Морозов та ін. – Херсон : ХДУ, 2007. – 112 с.
13. Шипулін В. Д. Планування і управління ГІС-проектами / В. Д. Шипулін, Є. І. Кучеренко. – Харків : ХНАМГ, ХНУРЕ, 2009. – 158 с
14. Географічна інформація – Еталонна модель: Нац. Стандарт України (ДСТУ ISO 19101:2002(Е). – К. : Держспоживстандарт України,2005. – 65 с
15. Joseph K. Berry. Beyond Mapping III. Understanding Spatial Patterns and Relationships [Електронний ресурс] / K. Joseph. – BASIS Press, 2007, 227 p. –

Режим доступу: http://www.innovativegis.com/basis/MapAnalysis/]. – Назва з екрана.

16. Шипулін В. Д. Основні принципи геоінформаційних систем : навч. посіб. / В. Д. Шипулін ; Харк. нац. акад. міськ. госп-ва. – Х. : ХНАМГ, 2010. – 313 с
17. Галузевий стандарт України. Правила кодування та цифрового опису векторних даних. СОУ 742-337395400012:2010. Проект. Перша редакція : у 2 т. – К. : Держспоживстандарт України, 2010. Т. 2. – 2010. – 228 с.