

МІНІСТЕРСТВО ВНУТРІШНІХ СПРАВ УКРАЇНИ

**Харківський національний
університет внутрішніх справ**

Кафедра інформаційних технологій факультету № 4

ТЕКСТ ЛЕКЦІЇ

з навчальної дисципліни

**«Теорія розподілених інформаційних ресурсів, захист баз даних»
вибіркових компонент освітньої програми
другого (магістерського) рівня вищої освіти**

125 «Кібербезпека» (Безпека інформаційних та комунікаційних систем)

за темою – «Середовище бази даних»

Харків 2018

ЗАТВЕРДЖЕНО

Науково-методичною радою
Харківського національного
університету внутрішніх справ
Протокол від _____ № ____

СХВАЛЕНО

Вченою радою
факультету № 4
Протокол від _____ № ____

ПОГОДЖЕНО

Секцією Науково-методичної ради
ХНУВС з технічних дисциплін
Протокол від _____ № ____

Розглянуто на засіданні кафедри інформаційних технологій факультету № 4 (*протокол від _____ № ____*).

Розробники:

1. Доцент кафедри інформаційних технологій факультету № 4, кандидат технічних наук, доцент Тулупов В.В.

Рецензенти:

1. Професор кафедри кібербезпеки факультету № 4, кандидат технічних наук, доцент Носов В.В.
2. Завідувач кафедри систем інформації Національного технічного університету «Харківський політехнічний інститут», д.т.н., професор Серков О.А.

План лекції

1. Трьохрівнева архітектура ANSI-SPARC
2. Схеми, відображення й екземпляри
3. Незалежність від даних
4. Мови баз даних
5. Моделі даних і концептуальне моделювання

Рекомендована література:

Основна:

1. Дейт К.Дж. Введение в системы баз данных. 6-е издание. - Киев – Москва: Діалектика, 1998. - 784 с.
2. Хансен Г., Хансен Дж. Базы данных: разработка и управление. – Москва: Бином, 1999. - 700 с. Пер. с англ.
3. Коннолли Т., Бегг К., Страчан А. Базы данных: проектирование, реализация и сопровождение. Теория и практика. 2-е издание. Москва – Санкт – Петербург - Киев: Вильямс, 2000. - 1111 с.

Додаткова:

4. Д. Кренке. Теория и практика построения баз данных. 8-е изд. – СПб.: Питер, 2003. – 800 с.: ил. – (Серия «Классика Computer Science»).
5. Гарсиа-Молина, Гектор, Ульман, Джефри, Д., Уидом, Дженифер. Системы баз данных. Полный курс.: Пер. с англ. – М.: Издательский дом «Вильямс», 2003. – 1088 с.: ил.
6. Кириллов В.В. Основы проектирования реляционных баз данных. Учебное пособие. Санкт-Петербургский Государственный институт точной механики и оптики (технический университет). Кафедра вычислительной техники. - <http://www.cs.ifmo.ru>
7. Кузнецов С.Д. Основы современных баз данных. Информационно-аналитические материалы. - <http://www.citmgu.ru/>

Текст лекції

Основною метою системи управління базами даних (далі - просто СУБД) є запропонування користувачу абстрактне представлення даних, сховавши конкретні особливості збереження і керування ними. Отже, відправною крапкою при проектуванні бази даних повинні бути абстрактний і загальний опис інформаційних потреб організації, що повинні знайти своє відображення базі даних, яка створюється. Наприклад, у навчальному проекті, приклади з якого було наведено у попередній лекції, становить інтерес моделювання наступних понять:

- сутностей "реального світу", таких як Працівник, Об'єкт нерухомості, Власник, і Орендар;
- атрибутів, що описують властивості або якості кожної сутності (наприклад, сутність Власник має атрибути Ім'я, Адреса і Паспортні дані;
- зв'язків між цими сутностями (наприклад, Власник - володіє - Об'єкт нерухомості).

Більш того, оскільки база даних є загальним ресурсом, то кожному користувачеві може знадобитися своє, відмінне від інших представлення про характеристики інформації, що зберігається в ній. Для задоволення цих потреб архітектура більшості сучасних комерційних СУБД у тій чи іншій мірі будується на базі так званої архітектури **ANSI-SPARC**. У цій лекції обговоримо різні архітектурні і функціональні характеристики СУБД.

3.1.Трьохрівнева архітектура ANSI-SPARC

Перша спроба створення стандартної термінології і загальної архітектури СУБД була почата в 1971 році групою, називаної **DBTG**. Група DBTG визнала необхідність використання дворівневого підходу, побудованого на основі використання системного представлення, тобто **схеми** (schema), і користувальницьких відображень, тобто підсхем (subschemata). Подібні термінологія й архітектура були запропоновані в 1975 році Комітетом планування стандартів і норм SPARC (Standards Planning and Requirements Committee) Національного Інституту Стандартизації США (American National Standard Institute — ANSI), ANSI/X3/SPARC (ANSI, 1975). Комітет ANSI/SPARC визнав необхідність використання **трьохрівневого** підходу.

Ці рівні формують трьохрівневу архітектуру, що охоплює **зовнішній, концептуальний і внутрішній** рівні. Мета трьохрівневої архітектури полягає у відділенні користувальницького представлення бази даних від її фізичного представлення. Нижче перераховано кілька причин, з яких бажано виконувати такий поділ:

Кожен користувач повинний мати можливість, звертатися до тих самим даних, використовуючи своє власне представлення про їх і повинний мати можливість змінювати його, причому ця зміна не повинна впливати на інших користувачів.

Користувачі не повинні безпосередньо мати справу з такими подробицями фізичного збереження даних у базі, як індексування і хешування. Інакше кажучи, взаємодія користувача з базою не повинне залежати від особливостей збереження в ній даних.

Адміністратор бази даних (АБД) повинний мати можливість змінювати структуру збереження даних у базі, не роблячи впливу на користувальницькі представлення.

Внутрішня структура бази даних не повинна залежати від таких змін фізичних аспектів збереження інформації, як переключення на новий пристрій збереження.

АБД повинний мати можливість змінювати концептуальну чи глобальну структуру бази даних без будь-якого впливу на всіх користувачів.

Рівень, на якому сприймають дані користувачі, називається зовнішнім рівнем (external level), тоді як СУБД і операційна система сприймають дані на внутрішньому рівні (internal level). **Концептуальний рівень** (conceptual level) представлення даних призначений для відображення зовнішнього, рівня на внутрішній і забезпечення необхідної незалежності один від одного.

3.1.1.Зовнішній рівень

Зовнішній рівень - відображення бази даних з погляду користувачів. Цей рівень описує ту частину бази даних, яка відноситься до кожного користувача.

Зовнішній рівень складається з декількох різних зовнішніх відображень бази даних. Кожен користувач має справу з відображенням "реального світу", вираженим у найбільш зручній для нього формі. Зовнішнє представлення містить тільки ті сутності, атрибути і зв'язки "реального світу", які цікаві користувачу. Інші сутності, чи атрибути зв'язків, що йому нецікаві, також можуть бути представлені в базі даних, але користувач може навіть не підозрювати про їх існування.

Крім цього, різні представлення можуть по-різному відображати одні і ті ж дані. Наприклад, один користувач може переглядати дати у форматі (день, місяць, рік), а інший — у форматі (рік, місяць, день). Деякі відображення можуть включати похідні чи обчислювані дані, що не зберігаються в базі даних як такі, а створюються в міру потреби. Відображення можуть також включати комбіновані чи похідні дані з декількох об'єктів.

3.1.2.Концептуальний рівень

Концептуальний рівень - узагальнююче представлення бази даних. Цей рівень описує, які дані зберігаються в базі даних, а також зв'язки, що існують між ними.

Проміжним рівнем в трьохрівневій архітектурі є концептуальний рівень. Цей рівень містить логічну структуру всієї бази даних (з погляду АБД). Фактично, це повне представлення вимог до даних з боку організації, що не залежить від будь-яких розумінь щодо способу їх збереження. На концептуальному рівні представлені наступні компоненти:

- усі сутності, їхні атрибути і зв'язки;
- обмеження, які накладаються на дані;
- семантична інформація про дані;
- інформація про міри забезпечення безпеки і підтримки цілісності даних.

Концептуальний рівень підтримує кожне зовнішнє представлення, у тому змісті, що будь-які доступні користувачу дані повинні міститися (чи можуть бути обчислені) на цьому рівні. Однак цей рівень не містить ніяких зведень про методи збереження даних. Наприклад, опис сутності повинен містити зведення про типи даних атрибутів (цілий, дійсний чи символічний) і їх довжини (кількості значущих цифр чи максимальній кількості символів), але не повинен включати зведень про організацію збереження даних, наприклад про обсяг зайнятого простору в байтах.

3.1.3. Внутрішній рівень

Внутрішній рівень - фізичне представлення бази даних у комп'ютері. Цей рівень описує, як інформація зберігається в базі даних.

Внутрішній рівень описує фізичну реалізацію бази даних і призначений для досягнення оптимальної продуктивності і забезпечення ощадливого використання дискового простору. Він містить опис структур даних і організації окремих файлів, які використовуються для збереження даних на запам'ятовуючих пристроях. На цьому рівні здійснюється взаємодія СУБД із методами доступу операційної системи (допоміжними функціями збереження і витягу записів даних) з метою розміщення даних на запам'ятовуючих пристроях, створення індексів, витягу даних і т.д. На внутрішньому рівні зберігається наступна інформація:

- розподіл дискового простору для збереження даних і індексів;
- опис подробиць збереження записів (із вказівкою реальних розмірів елементів даних, що зберігаються);
- зведення про розміщення записів;
- зведення про стиск даних і обраних методах їхнього шифрування.

Нижче внутрішнього рівня знаходиться **фізичний рівень** (physical level), що контролюється операційною системою, але під керівництвом СУБД. Однак функції СУБД і операційної системи на фізичному рівні не цілком чітко розділені і можуть варіюватися від системи до системи. Фізичний рівень доступу до даних нижче СУБД складається тільки з відомих операційній системі елементів (наприклад, покажчиків, як реалізоване послідовний розподіл і т.д.).

3.2. Схеми, відображення й екземпляри

Загальний опис бази даних називається **схемою бази даних**. Існує три різних типи схем бази даних, що визначаються відповідно до рівнів абстракції трьохрівневої архітектури. На найвищому рівні мається кілька **зовнішніх схем** чи підсхем, що відповідають різним представленням даних.

На концептуальному рівні опис бази даних називають **концептуальною схемою**, а на найнижчому рівні абстракції — **внутрішньою схемою**.

Концептуальна схема описує всі елементи даних і зв'язки між ними, із вказівкою необхідних обмежень підтримки цілісності даних. Для кожної бази даних мається тільки одна концептуальна схема.

На нижньому рівні знаходиться **внутрішня схема**, що є повним описом внутрішньої моделі даних. Вона містить визначення збережених записів, методи представлення, опису полів даних, зведення про індекси й обрані схеми хешування. Для кожної бази даних існує тільки одна внутрішня схема.

СУБД відповідає за установлення відповідності між цими трьома типами схем, а також за перевірку їхньої несуперечності. Інакше кажучи, СУБД повинна переконатися в тім, що кожную зовнішню схему можна вивести на основі концептуальної схеми. Для встановлення відповідності між будь-якими зовнішньою і внутрішньою схемами СУБД повинна використовувати інформацію з концептуальної схеми. Концептуальна схема зв'язана з внутрішньою схемою за допомогою концептуально внутрішнього відображення. Воно дозволяє СУБД знайти фактичний запис чи набір записів на фізичному пристрої збереження, що утворюють логічний запис у концептуальній схемі, з урахуванням будь-яких обмежень, установлених для виконуваних над даним логічним записом операцій. Воно також дозволяє знайти будь-які розходження в іменах об'єктів, іменах атрибутів, порядку проходження атрибутів, їхніх типах даних і т.д. Нарешті, кожна зовнішня схема зв'язана з концептуальною схемою за допомогою зовнішнього концептуального відображення. З його допомогою СУБД може відображати імена користувальницького представлення на відповідну частину концептуальної схеми.

Зовнішнє відображення 1

Таб_Номер	Прізвище	Ім'я	Вік	Зарплата
-----------	----------	------	-----	----------

Зовнішнє відображення 2

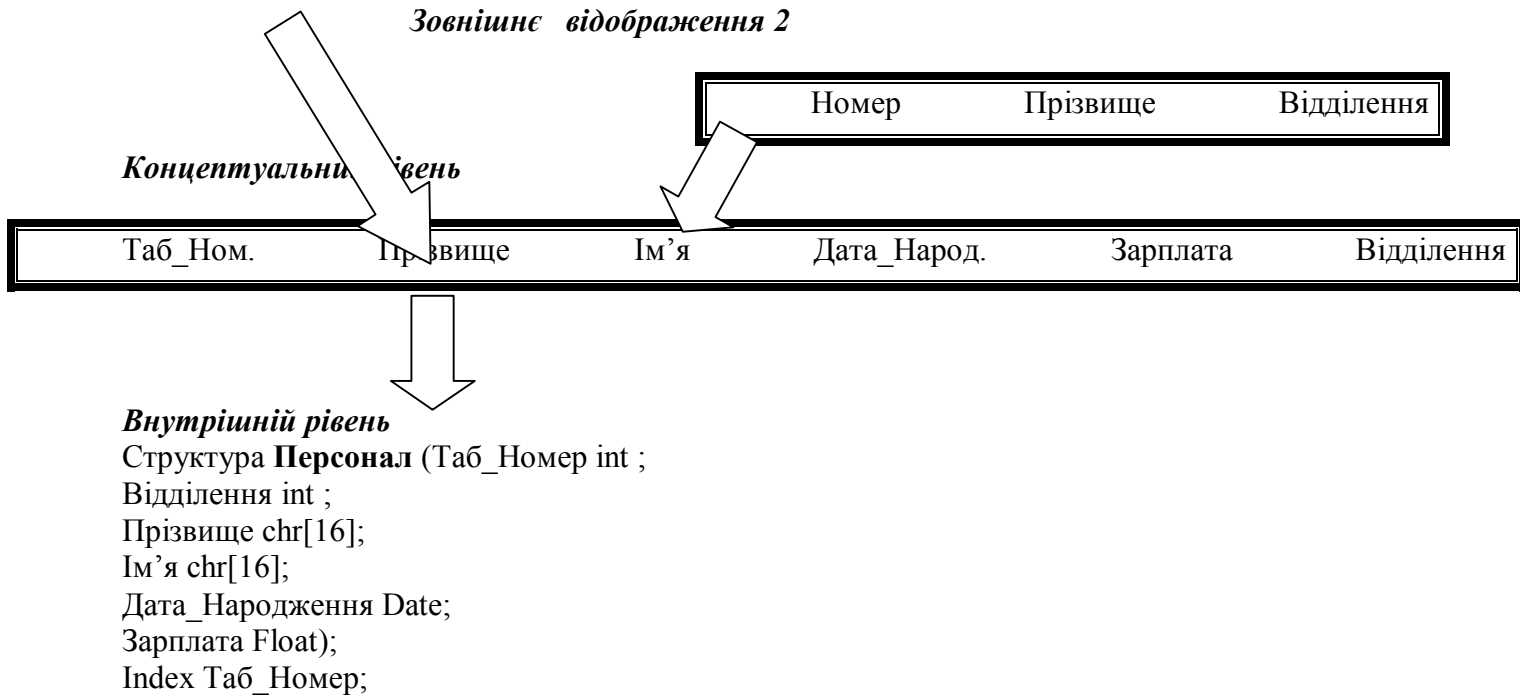


Рис.3.1. Три рівні відображень

Приклади різних рівнів наведені на рис.3.1. На ньому показані два різних зовнішніх представлення інформації про персонал: одне складається з особистого номера співробітника (Таб_Номер), його імені (Ім'я) і прізвища (Прізвище), віку (Вік), суми зарплати за рік (Зарплата) і номера відділення компанії, у якому цей співробітник працює (Відділення). Інше представлення включає особистий номер співробітника (Таб_Номер), прізвище (Прізвище) і номер відділення компанії, у якому він працює (Відділення). Ці зовнішні відображення зливаються воедино в одному концептуальному відображенні.

Важливо розрізняти опис бази даних і саму базу даних. Описом бази даних є **схема бази даних**. Схема створюється в процесі проектування бази даних, причому передбачається, що вона змінюється досить рідко. Однак інформація, що міститься в базі даних, може мінятися часто - наприклад, кожний раз при вставці зведень про нового співробітника новому об'єкту нерухомості, що здається в оренду. Сукупність інформації, що зберігається в базі даних у будь-який визначений момент часу, називається **станом бази даних**. Отже, однієї і тій же схемі бази даних може відповідати безліч її різних станів. Схема бази даних іноді називається **змістом** бази даних, а її стан — **деталізацією**.

3.3. Незалежність від даних

Основним призначенням трьохрівневої архітектури є забезпечення **незалежності від даних**, що означає, що зміни на **нижніх** рівнях ніяк не впливають на верхні рівні. Розрізняють два типи незалежності від даних: **логічну** і **фізичну**.

Логічна незалежність від даних означає повну захищеність зовнішніх схем від змін, внесених у концептуальну схему.

Такі зміни концептуальної схеми, як додавання чи видалення нових сутностей, чи атрибутів зв'язків, повинні здійснюватися без необхідності внесення змін у вже існуючі зовнішні схеми або переписування прикладних програм. Ясно, що користувачі, для яких ці зміни призначалися, повинні знати про їх, але дуже важливо, щоб інші користувачі навіть не підозрювали про це.

Фізична незалежність від даних означає захищеність концептуальної схеми від змін, внесених у внутрішню схему.

Такі зміни внутрішньої схеми, як використання різних файлових систем чи структур збереження, різних пристроїв збереження, модифікація індексів чи хешування, повинні здійснюватися без необхідності внесення змін у концептуальну чи зовнішню схеми. Користувачем можуть бути замічені зміни тільки в загальній продуктивності системи. Насправді найбільш

розповсюдженою причиною внесення змін у внутрішню схему є саме недостатня продуктивність виконання операцій.

3.4. Мови баз даних

Внутрішня мова СУБД для роботи з даними складається з двох частин: **мови визначення даних** (Data Definition Language - DDL) і **мови керування даними** (Data Manipulation Language - DML). Мова DDL використовується для визначення схеми бази даних, а мова DML - для читання і відновлення даних, збережених у базі.

Ці мови називаються **підмовами даних**, оскільки в **них** відсутні конструкції для виконання всіх обчислювальних операцій, які звичайно використовуються у мовах програмування високого рівня. У багатьох СУБД передбачена можливість впровадження операторів підмови даних у програми, написаних на таких мовах програмування високого рівня, як C++, Pascal, чи VB. У цьому випадку мову високого рівня прийнято називати **базовою мовою** (host language). Перед компіляцією файлу програми базовою мовою, що містять впроваджені оператори підмови даних, буде потрібно попередньо видалити ці оператори, замінивши їх викликами відповідних функцій СУБД.

3.4.1. Мова визначення даних - DDL

Мова DDL – мова опису, що дозволяє АБД або користувачеві описати і поіменувати сутності, необхідні для роботи деякої програми, а також зв'язки між різними сутностями.

Схема бази даних складається з набору визначень, виражених спеціальною мовою визначення даних – DDL. Мова DDL використовується як для визначення нової схеми, так і для модифікації вже існуючої. Цю мову не можна використовувати для керування даними.

Результатом компіляції DDL-операторів є набір таблиць, збережених в особливих файлах, називаних **системним каталогом**. У системному каталозі інтегровані **мета-дані** - тобто дані, що описують об'єкти бази даних, а також дозволяють спростити спосіб доступу до **них** і керування **ними**. Мета-дані включають визначення записів, елементів даних, а також інші об'єкти, що представляють інтерес для користувачів чи необхідні для роботи СУБД. Перед доступом до реальних даних СУБД звичайно звертається до системного каталогу. Для позначення системного каталогу також використовуються терміни **словник даних** і **каталог даних**, хоча перший з них (словник даних) звичайно відноситься до програмного забезпечення більш загального типу, чим просто каталог СУБД.

3.4.2. Мова керування даними - DML

Мова DML - мова, що містить набір операторів для підтримки основних операцій маніпулювання даними, що містилися в базі.

До операцій керування даними відносяться:

- вставка в базу даних нових зведень;
- модифікація зведень, збережених у базі даних;
- витяг зведень, що містяться в базі даних;
- видалення зведень з бази даних.

Таким чином, одна з основних функцій СУБД полягає в підтримці мови маніпулювання даними, за допомогою якої користувач може створювати вирази для виконання перерахованих вище операцій з даними. Поняття маніпулювання даними застосовується як до зовнішнього і концептуального рівнів, так і до внутрішнього рівня. Однак на внутрішньому рівні для цього необхідно визначити дуже складні процедури низького рівня, що дозволяють виконувати доступ до даних дуже ефективно. На більш високих рівнях, навпаки, акцент переноситься у бік більшої простоти використання й основних зусиль направляються на забезпечення ефективної взаємодії користувача із системою.

Мови DML відрізняються базовими конструкціями витягу даних. Варто розрізняти два типи мов DML: **процедурний** і **непроцедурний**. Основна відмінність між ними полягає в тому, що процедурні мови вказують те, **як** можна одержати результат оператора мови DML, тоді як непроцедурні мови описують те, **який** результат буде отриманий. Як правило, у процедурних мовах записи розглядаються окремо, тоді як не процедурні мови оперують з цілими наборами записів.

3.4.3. Процедурні мови DML

Процедурна мова DML - мова, яка дозволяє повідомити системі про те, які дані необхідні, і точно вказати, як їх можна витягти.

За допомогою процедурної мови DML користувач, а точніше - програміст, указує на те, які дані йому необхідні і як їх можна одержати. Це значить, що користувач повинний визначити всі операції доступу до даних, (здійснювані за допомогою виклику відповідних процедур), що повинні бути виконані для одержання необхідної інформації. Звичайно така процедурна мова DML дозволяє витягти запис, обробити його і, у залежності від отриманих результатів, витягти інший запис, що повинен бути підданий аналогічній обробці, і т.д. Подібний процес витягу даних продовжується доти, поки не будуть витягнуті всі запитувані дані. Мови DML **мережних і ієрархічних СУБД** як правило є процедурними.

3.4.4.Непроцедурні мови DML

Непроцедурна мова DML - мова, яка дозволяє вказати лише те, які дані потрібні, але не те, як їх варто витягати.

Непроцедурні мови DML дозволяють визначити весь набір необхідних даних за допомогою одного оператора витягу чи відновлення. За допомогою непроцедурних мов DML користувач указує, які дані йому потрібні, без визначення способу їх одержання. СУБД трансліює вираз мовою DML у процедуру (чи набір процедур), що забезпечує маніпулювання викликаним набором записів. Даний підхід звільняє користувача від необхідності знати деталі внутрішньої реалізації структур даних і особливості алгоритмів, які використовуються для витягу і можливого перетворення даних. У результаті робота користувача одержує визначений ступінь незалежності від даних. Реляційні СУБД у тій чи іншій формі як правило включають підтримку непроцедурних мов маніпулювання даними - найчастіше це буває **мова структурованих запитів SQL** (Structured Query Language) чи мова запитів за зразком **QBE (Query-by-Example)**. Непроцедурні мови звичайно простіше зрозуміти і використовувати, ніж процедурні мови DML, оскільки користувачем виконується менша частина роботи, а СУБД - велика.

Частина непроцедурної мови DML, що відповідає за витяг даних, називається **мовою запитів**. Мову запитів можна визначити як високорівневу вузькоспеціалізовану мову, яка призначена для задоволення різних вимог по вибірці інформації з бази даних. У цьому змісті термін "запит" зарезервований для позначення оператора витягу даних, вираженого за допомогою мови запитів. Терміни "мова запитів" і "мова керування даними" часто використовуються як синоніми, хоча, з технічної точки зору, це некоректно.

3.4.5.Мови 4GL

Абревіатура "4GL" являє собою скорочений англійський варіант написання терміна **мова четвертого покоління** (Fourth-Generation Language). Не існує чіткого визначення цього поняття, хоча, по суті, мова йде про деякий стенографічний варіант мови програмування. Якщо для організації деякої операції з даними мовою третього покоління (3GL) типу COBOL буде потрібно написати сотні рядків коду, то для реалізації цієї ж операції мовою четвертого покоління буде досить 10-20 рядків.

У той час, як мови третього покоління є процедурними, мови 4GL виступають як непроцедурні, оскільки користувач визначає, *що* повинно бути зроблено, але не говорить, як **саме** бажаний результат повинний бути досягнутий.

Як приклади мов четвертого покоління можна вказати згадувані вище мови SQL і QBE.

3.5.Моделі даних і концептуальне моделювання

Вище вже згадувалося, що схема створюється за допомогою деякої мови визначення даних. Насправді вона створюється на основі мови визначення даних конкретної цільовий СУБД. На жаль, це мова низького рівня; з її допомогою важко описати вимоги до даних у масштабі всієї організації так, щоб створена схема була доступна розумінню користувачів самих різних категорій. У чому ми дійсно бідуємо, так це в описі схеми на якомусь, більш високому рівні, що будемо називати **моделлю даних**.

Модель даних - інтегрований набір понять для опису даних, зв'язків між ними й обмежень, що накладаються на дані в деякій організації.

Модель є представленням "реального світу" об'єктів і подій, а також існуючих між ними зв'язків. Модель повинна відбивати основні концепції, представлені в такому вигляді, що дозволить

проектувальникам і користувачам бази даних обмінюватися конкретними і недвозначними думками про їх розуміння ролі тих чи інших даних у цій організації.

Мета побудови моделі даних полягає в представленні їх у зрозумілому вигляді. Якщо таке представлення можливе, то модель даних можна буде легко застосувати при проектуванні бази даних.

У літературі запропоновано й опубліковано досить багато моделей даних. Вони підрозділяються на три категорії:

- об'єктні (object-based) моделі даних;
- моделі даних на основі записів (record-based);
- фізичні моделі даних.

Перші дві використовуються для опису даних на концептуальному і зовнішньому рівнях, а остання - на внутрішньому рівні.

3.1.1. Об'єктні моделі даних

При побудові об'єктних моделей даних використовуються такі поняття як **сутності, атрибути і зв'язки**. **Сутність** - це окремий елемент (співробітник, чи місце, чи поняття подія) організації, що повинен бути представлений в базі даних. **Атрибут** - це властивість, що описує деякий аспект об'єкта і значення якого варто зафіксувати, а **зв'язок** є асоціативним відношенням між сутностями. Нижче перераховані деякі найбільш загальні типи об'єктних моделей даних.

- Модель типу "сутність-зв'язок", чи ER-модель (Entity-Relationship model).
- Семантична модель.
- Функціональна модель.
- Об'єктно-орієнтована модель.

В даний час **ER-модель** стала одним з основних методів концептуального проектування баз даних, і саме вона покладена в основу пропонованої в цьому курсі методології проектування баз даних. **Об'єктно-орієнтована** модель розширює визначення сутності з метою включення в нього не тільки атрибутів, що описують стан об'єкта, але і дій, що з ним зв'язані, тобто його поведінки. У такому випадку говорять, що об'єкт інкапсулює стан і поведінку.

3.2.1. Моделі даних на основі записів

У моделі на основі записів база даних складається з декількох записів фіксованого формату, що можуть мати різні типи. Кожен тип запису визначає фіксовану кількість полів, кожне з яких має фіксовану довжину. Існує три основних типи логічних моделей даних на основі записів:

- реляційна модель даних (relational data model),
- мережна модель даних (network data model)
- ієрархічна модель даних (hierarchical data model).

Ієрархічна і мережна моделі даних були створені майже на десять років раніш реляційної моделі даних, а тому їх зв'язок з концепціями традиційної обробки файлів більш очевидна.

3.2.1.1. Реляційна модель даних

Реляційна модель даних заснована на понятті математичних відношень. У реляційній моделі дані і зв'язки представлені у виді таблиць, кожна з яких має кілька стовпців з унікальними іменами. Наприклад між відношеннями **Персонал** і **Відділення** існує наступний зв'язок: співробітник *працює* у відділенні компанії. Однак між цими двома відношеннями немає явно заданого зв'язку; її існування можна помітити, тільки знаючи, що атрибут **Відділення** у відношенні **Персонал** еквівалентний атрибуту **Номер** у відношенні **Відділення**.

Зверніть увагу, що в реляційній моделі даних єдина вимога полягає в тому, щоб база даних з погляду користувача виглядала як набір таблиць. Однак таке сприйняття відноситься тільки до логічної структури бази даних, тобто до зовнішнього і концептуального рівнів архітектури ANSI/SPARC. Воно не відноситься до фізичної структури бази даних, що може бути реалізоване за допомогою різноманітних структур збереження.

3.2.1.2. Мережна модель даних

У мережній моделі дані представлені у вигляді колекцій записів, а зв'язки - у вигляді **наборів**. На відміну від реляційної моделі, зв'язку тут явно моделюються наборами, що реалізуються за допомогою покажчиків. Мережну модель можна представити як граф із записами у виді **вузлів** графа

і наборами у вигляді його **ребер**. Самою популярною мережною СУБД є система IDMS/R фірми Computer Associates.

3.2.1.3. Ієрархічна модель даних

Ієрархічна модель є обмеженим підтипом мережної моделі. У ній дані також представлені як колекції **записів**, а зв'язку — як **набори**. Однак в ієрархічній моделі вузол може мати тільки одного батька. Ієрархічна модель може бути представлена як деревоподібний граф із записами у виді вузлів (які також називаються сегментами) і множинами у виді ребер. Найпоширенішою ієрархічною СУБД є система IMS корпорації IBM, хоча вона має також деякі інші неієрархічні риси.

Засновані на записах моделі даних використовуються для визначення загальної структури бази даних і високорівневого опису її реалізації, але їх основний недолік полягає в тому, що вони не дають адекватних засобів для явної вказівки обмежень, що накладаються на дані. У той же час в об'єктних моделях даних відсутні засоби вказівки їх логічної структури, але за рахунок надання користувачу можливості вказати обмеження для даних, вони дозволяють у більшій мірі представити семантичну суть збереженої інформації.

Більшість сучасних комерційних систем засновано на **реляційній парадигмі**, тоді як найперші системи баз даних будувалися на основі мережної чи ієрархічної моделі. **При використанні останніх двох моделей від користувача потрібно знання фізичної організації бази даних**, до якої він повинен здійснювати доступ, у той час як при роботі з реляційною моделлю **незалежність від даних забезпечується в значно більшому ступені**. Отже, якщо в реляційних системах для обробки інформації в базі даних прийнятий декларативний підхід (тобто вони вказують, *які дані варто витягти*), то в мережних і ієрархічних системах - навігаційний підхід (тобто вони вказують, *як їх варто витягти*).

3.5.3. Фізичні моделі даних

Фізичні моделі даних описують те, як дані зберігаються в комп'ютері, представляючи інформацію про структуру записів, їх упорядкованості і існуючих шляхах доступу. Фізичних моделей даних не так багато, як логічних, а самими популярними серед них є *узагальнююча модель (unifying model)* і *модель пам'яті кадрів (frame memory)*.

Питання для контролю

1. З яких рівнів складається трьохрівнева архітектура **ANSI-SPARC**?
2. Дайте коротке пояснення у чому різниця між зовнішнім та концептуальним рівнем.
3. У чому різниця між концептуальним та внутрішнім рівнем трьохрівневої архітектури **ANSI-SPARC**?
4. У чому полягає основна мета використання трьохрівневої архітектури? Дайте стислу відповідь.
5. Чи повинен розроблювач бази даних детально визначати внутрішній рівень зберігання інформації?
6. Що таке схема бази даних? Які види схем існують? Дайте скорочений опис кожної з них.
7. Як Ви розумієте «незалежність від даних»? Які види незалежності існують? Опишіть їх риси.
8. З якою метою використовується мова визначення даних DDL?
9. Які операції можна виконати за допомогою використання мови керування даними DML?
10. Поясніть різницю між процедурними та не процедурними мовами DML.
11. Як Ви розумієте мову 4GL?
12. Які моделі даних існують та у чому їх характерні риси? Дайте стислий опис кожної з них.