МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
“ДНІПРОВСЬКА ПОЛІТЕХНІКА”

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра інформаційних технологій та
комп’ютерної інженерії

Каштан В.Ю., Іванов Д.В.

Конспект лекцій
з дисципліни “Бази даних в інформаційних системах ”.
Чистина 1

Для студентів галузі знань 12 “Інформаційні технології”
спеціальності 126 “Інформаційні системи та технології”

Дніпро
НТУ “ДП”
2021

УДК 004.4'23+004.42+004.432
Г20

Каштан В.Ю., Іванов Д.В. Конспект лекцій з дисципліни “Бази даних в інформаційних системах”. Для студентів галузі знань 12 “Інформаційні технології” спеціальності 126 “Інформаційні системи та технології”. – Д.: НТУ «ДП», 2021. – 58 с.

Розробники:

– Каштан В.Ю. – кандидат технічних наук, доцент кафедри інформаційних технологій та комп’ютерної інженерії (лекції 1-4);

– Іванов Д.В. – асистент кафедри інформаційних технологій та комп’ютерної інженерії (лекції 5-7).

В конспекті лекцій з дисципліни “Бази даних в інформаційних системах” для студентів спеціальності 126 “Інформаційні системи та технології” приведена перша частина, яка викладається за навчальним планом у п’ятій чверті другого курсу.

Основна увага в лекціях приділяється теоретичним і практичним знанням в області баз даних з використання мови запитів SQL в складі інформаційних системах

Протягом лекційного курсу розглядаються приклади створення баз даних.

Погоджено рішенням науково-методичної комісії спеціальності 126 *Інформаційні системи та технології* (*протокол № 7 від* 27.08.2020).

ЗМІСТ

ВСТУП

Метою дисципліни “Бази даних в інформаційних системах ” для студентів спеціальності 126 “Інформаційні системи та технології” є формування компетентностей щодо формування системи теоретичних і практичних знань в області баз даних, вивчення концепції моделювання даних в інформаційних системах, організації реляційних, розподілених та об’єктно-орієнтованих баз даних та знань, етапів їх проектування, організації ефективної структури зберігання даних, організації запитів до збережених даних, методів забезпечення цілісності даних, а також здобуття практичних навичок використання мови запитів SQL в складі інформаційних системах [1].

В першій частині конспекту лекцій приділяється значна увага проектуванню та створенню бази даних, її нормалізації та застосуванню мови SQL.

Всі приклади кодів програм, представлених в конспекті лекцій, , були скомпільовані за допомогою MySQL. Для створення, модифікації та керування даними у реляційних базах даних вивчається універсальна мова структурованих запитів SQL. Для аналізу інформації, яка зберігається у базі даних, вивчаються такі засоби SQL, як уявлення, збережені процедури та тригери. Також розглядаються основні моделі баз знань

Основними дисциплінарними результатами навчання, після завершення першої частини лекційного курсу “ Бази даних в інформаційних системах ”, є:

– надбання базових теоретичних знань щодо проектування баз даних;

– вміння проводити аналіз предметної області, для якої розробляється база даних;

– розроблення інформаційної системи та бази даних засобами MS SQL Server та в и в архітектурі клієнт-сервер;

– отримання навичок проектування реляційну модель бази даних, а також вміння виконання транзакцій в БД.

Лекція 1. Вступ до курсу: «Бази даних в інформаційних системах».

1.1 Загальні роложення

Необхідність пошуку потрібної інформації у людини виникає повсякчас, незалежно від сфери її професійних інтересів: з якої платформи відправляється потяг на Хмельницький, як приготувати вареники з вишнями, яку будову має молекула води, скільки днів тривала Друга світова війна, чи справджується прикмета про чорну кішку, яка частота змінного електричного струму в побутовій електричній мережі, яке закінчення мають іменники третьої відміни в родовому відмінку однини та ін. Відповіді на частину з цих запитань людина може отримати зі своєї пам'яті, для отримання інших необхідно звернутися до інформаційної системи залізничного вокзалу, переглянути кулінарну книжку, довідник з хімії, фізики чи електротехніки, посібник з правопису тощо. Для полегшення пошуку потрібної інформації людство придумало багато засобів – універсальні енциклопедії та енциклопедії з предметних галузей, довідники й словники, довідкові бюро та інформаційні табло та ін. [2]

Обсяги повідомлень, які накопичило людство, невпинно зростають. Так, при розкопках стародавнього міста шумерів Ур було знайдено понад 20 тисяч глиняних табличок з відомостями про звичаї давнього народу, його легенди та історичні події, що відбувалися понад 5 тисяч років тому. Знаменита Александрійська бібліотека, яка була заснована в Єгипт у III ст. до нової ери, за різними джерелами містила від 100 до 700 тисяч рукописів. Сьогоднішні бібліотеки вражають обсягами різноманітних даних. Найбільшою у світі вважається Британська бібліотека в Лондоні, яка нараховує понад 150 млн одиниць зберігання, а найбільша бібліотека нашої країни – Національна бібліотека України імені В.І. Вернадського в Києві нараховує понад 15 млн одиниць зберігання.

Учені запевняють, що зберігання великих обсягів даних виправдано тільки за умови, якщо пошук потрібних даних здійснюється швидко і подаються вони в доступній для розуміння формі. Ці умови забезпечують сучасні технології зберігання даних. Основою цих технологій є комп'ютеризовані бази даних (БД).

База даних – це впорядкований за певними правилами набір взаємопов'язаних даних.

Перша в Україні комп'ютерна база даних була розроблена в ході робіт з проектування і експлуатації електронної обчислювальної машини «Київ» (1959 р.). ЕОМ була розроблена для обчислювального центру Академії наук УРСР Л.Н. Дашевським, К.Л. Ющенко, К.О. Шкарабарою, С.Б. Погребинським під науковим керівництвом Б.В. Гніденка та В.М. Глушкова.

1.2 Інформація та інформаційна система

Змістовна сторона поняття "інформація" дуже багатогранна й не має чітких

семантичних меж. Однак завжди можна сказати, що можна з нею робити. Саме відповідь на це запитання найчастіше й цікавить як системних аналітиків і розроблювачів інформаційної системи (ІС), так і користувачів інформації (її основних споживачів).

З погляду як користувачів, так і розроблювачів ІС в інформації є одна важлива властивість - вона є одиницею даних, яка підлягає обробці. Звичайно інформація надходить споживачеві саме у вигляді даних: таблиць, графіків, малюнків, фільмів, усних повідомлень, які фіксують у собі інформацію певної структури й типу. Таким чином, дані виступають як засіб подання інформації у певній, фіксованій формі, придатній для обробки, зберігання й передачі. Хоча дуже часто терміни "інформація" й "дані" виступають як синоніми, варто пам'ятати про цю їхню істотну відмінність. Саме в даних інформація знаходить інтерпретацію у конкретній ІС.

При згадуванні про "форму" подання інформації варто сказати ще про одну, "людську" властивість інформації - її сприйняття різними категоріями людей. Дані можуть бути згруповані спільно у документ. Документ може мати або не мати певної внутрішньої структури. Дані можуть бути відображені на екрані дисплея комп'ютера. Документи можуть мати аудіо- або відеоформу. Розробляючи ІС, ніколи не слід забувати, для кого вони (системи) створюються і хто буде їх використовувати. Форма подання інформації в ІС визначає також і категорії користувачів. ІС створюються для конкретних груп користувачів, тобто вони, як правило, проблемно- орієнтовані.

Інформація є даними, яким надається деякий зміст (інтерпретація) у конкретній ситуації у рамках деякої системи понять. Інформація подається за допомогою кодування даних і здобувається шляхом їхнього декодування та інтерпретації.

У цьому визначенні фіксується три основних перетворення інформації й даних у процесі їхньої обробки в ІС: інформація – дані, дані – дані, дані – інформація.

На рисунку 1.1 наведені дві сторони визначення поняття інформації: функціональна й представницька. Перша загалом визначає коло дій над інформацією, а друга – результат виконання цих дій.

Основною метою створення ІС є задоволення інформаційних потреб користувачів шляхом надання необхідної їм інформації на основі збережених даних. Потреба в інформації як такій не вичерпує поняття інформаційних потреб. Звичайно в поняття інформаційних потреб включають певні вимоги до якості інформаційного обслуговування й поводження системи в цілому (продуктивність, актуальність і надійність даних, орієнтація на користувача та ін.).

Під інформаційною системою розуміється організаційна сукупність технічних засобів, технологічних процесів і кадрів, що реалізують функції збору, обробки, зберігання, пошуку, видачі й передачі інформації.

Необхідність підвищення продуктивності праці у сфері інформаційної діяльності приводить до того, що в якості зовнішніх засобів зберігання й швидкого

доступу до інформації найчастіше використовуються засоби обчислювальної техніки (цифровий і аналоговий) на основі комп'ютерів. Сучасні ІС - складні комплекси апаратних і програмних засобів, технології й персоналу, які ще називають автоматизованими інформаційними системами [1,5].

Рисунок 1.1 - Зміст поняття "інформація"

Структурно ІС містять у собі апаратне (hardware), програмне (software), комунікаційне (netware), проміжного шару (middleware), лінгвістичне й організаційно-технологічне забезпечення.

Апаратне забезпечення ІС містить у собі широкий набір засобів обчислювальної техніки, передачі даних, а також цілий ряд спеціальних технічних пристроїв (пристрою графічного відображення інформації, аудио- і відеопристрою, засобу мовного уведення й т.д.). Апаратне забезпечення є основою будь-якої ІС.

Комунікаційне (мережне) забезпечення містить у собі комплекс апаратних мережних комунікацій і програмних засобів підтримки комунікацій в ІС. Воно має істотне значення при створенні розподілених ІС й ІС на основі Інтернету.

Програмне забезпечення ІС забезпечує реалізацію функцій введення даних, їх розміщення на носіях, модифікації даних, доступ до даних, підтримку функціонування устаткування. Програмне забезпечення можна розділити на системне (яке завершує процес вибору апаратно-програмного рішення або платформи) і користувацьке (яке застосовується для вирішення завдань задоволення потреб користувача у комп'ютерному середовищі).

Лінгвістичне забезпечення ІС призначене для вирішення завдань формалізації змісту повнотекстової і спеціальної інформації для створення пошукового образу даних (профілю). У класичному змісті звичайно воно включає процедури індексування текстів, їхню класифікацію і тематичну рубрикацію.

Найчастіше ІС, що містять складно-структуровану інформацію, містять у собі тезауруси термінів і понять. Сюди можна віднести й створення процесорів спеціалізованих формальних мов кінцевих користувачів, наприклад, мов для маніпулювання бухгалтерською інформацією і т.д. Найчастіше працям з розроблення лінгвістичного забезпечення не надається належного значення. Подібні недогляди найчастіше призводять до несприйняття користувачами самої інформації. Це стосується в першу чергу вузько спеціалізованих ІС.

У міру зростання складності і масштабів ІС важливу роль починає відігравати організаційно-технологічне забезпечення, що з'єднує різнорідні компоненти (апаратури, програми й персонал) у єдину систему й забезпечує процедури її керування й функціонування. Недооцінка цієї складової ІС найчастіше призводить до зриву строків впровадження системи й виведення її на виробничі потужності.

1.3 Обробка інформації в ІС

Одним з головних питань розроблення програмного забезпечення ІС є питання про співвіднесення програм і даних, тому що вирішення цього питання, в остаточному підсумку, визначає вибір алгоритмів обробки інформації, апаратних засобів і технологічної платформи. Фундаментальним принципом у вирішенні питання про співвіднесення програм і даних є концепція незалежності прикладних програм від даних, і неважливо, яка обробка даних передбачається: централізована або розподілена. Суть цієї концепції полягає не стільки у відділенні програм від даних, скільки у розгляді їх як самостійних взаємодіючих об'єктів.

Однією з останніх модифікацій цього принципу є концепція незалежності прикладних програм від даних разом із процедурами їхньої обробки (об’єктно-орієнтований підхід у програмуванні), що дозволяє вирішити ряд питань обробки даних, пов'язаних з інтерпретацією семантичного змісту даних.

Формування концепції БД і створення на її основі методу баз даних для вирішення завдань обробки інформації відбулося у 1962 році. До середини 60-х років минулого століття основною концепцією побудови програмного забезпечення були концепція файлової системи і так званий позадачний метод. Наприкінці 80-х років минулого століття була запропонована концепція об’єктно-орієнтованих баз даних й об’єктно-орієнтований підхід розроблення програм на основі обробки подій [3].

Отже, при розробленні автоматизованих ІС необхідно враховувати:

- конкретні групи користувачів, тобто вони є проблемно-орієнтовані;
- задоволення інформаційних потреб користувачів шляхом надання необхідної їм інформації на основі збережених даних;
- складні комплекси апаратних і програмних засобів, технології й персоналу, які ще називають автоматизованими інформаційними системами;
- концепції незалежності прикладних програм від даних разом із процедурами їхньої обробки (об’єктно-орієнтований підхід у програмуванні) дозволяє вирішити ряд питань обробки даних, пов'язаних з інтерпретацією

семантичного змісту даних;

− упорядковані взаємозалежні данні, які зберігаються з мінімальною надлишковістю та становлять базу даних. Системою керування базами даних називається сукупність програмних засобів, необхідних для використання БД і подання розробникам і користувачам безлічі різних подань даних.

Лекція 2. Архітектура БД. Моделі даних.

2.1 Архітектура БД

Якщо говорити про використання обчислювальної техніки, то глобально можна виділити два основних напрямки.

Перший напрямок - чисельні розрахунки. Історично воно з'явилося раніше і сприяло розвитку методів чисельного рішення складних математичних задач, розвитку мов програмування, орієнтованих на рішення обчислювальних задач.

Другий напрямок - це зберігання і обробка даних. Метою будь-якої інформаційної системи є зберігання і обробка даних про будь-які об'єкти реального світу.

Розглянемо такі важливі для нас поняття як «дані» і «інформація». Незважаючи на величезну кількість визначень для цих понять зупинимося на наступних визначеннях.

Інформація являє собою відомості про оточуючих людини предмети, явища і процеси і є об'єктом таких операцій як сприйняття, передача, перетворення, зберігання і використання.

Коли використовується термін «дані», то мова йде про інформацію, представленої в формалізованому вигляді, придатної для автоматичної обробки при можливу участь людини.

У широкому сенсі слова термін «база даних» (БД) - це сукупність відомостей про конкретні об'єкти.

При створенні БД в основному мають на меті впорядкувати дані за різними ознаками, щоб мати можливість брати з даних потрібну інформацію.

Створення БД, її підтримка, управління, а також доступ користувачів до самих даних здійснюється за допомогою спеціальних програмних продуктів, які називаються системами управління базами даних (СКБД).

Основна особливість СУБД - це наявність процедур для введення і збереження не тільки самих даних, але і описів їх структури.

Файли, забезпечені описом збережених у них даних і знаходяться під управлінням СУБД, стали називати БД [4].

Функції СУБД:

- управління буферами оперативної пам'яті;
- управління транзакціями;
- захист від відмов і відновлення (журнализация);
- забезпечення різних рівнів доступу до даних.

Термін «архітектура» може мати різні тлумачення. Наприклад, коли вказують-ся функціональні модулі системи й способи їхньої взаємодії, йдеться про функці-ональну архітектуру. Спосіб реалізації функцій системи, її компоненти та вза-ємозв'язки між ними фіксує архітектура реалізації системи. У цьому контексті можна також згадати архітектуру технічних засобів систем. Говорячи ж про термін «архітектура БД», ми матимемо на увазі архітектуру інформаційного за-безпечення.

Архітектура БД уперше була специфікована дослідницькою групою ANSI/X3/ SPARC Study Group on Data Base Management Systems (ANSI - Американсь-кий національний інститут стандартів, X3 – його комітет обчислювальної техніки й обробки інформації, SPARC – підкомітет ANSI/X3 з планування стан-дартів ANSI). Метою дослідницької групи було визначення іалузей і технологій баз даних, в яких було б доречно проводити стандартизацію, а також вироблення рекомендацій для роботи в кожній із таких галузей. Група дійшла висновку, що, можливо, єдиним аспектом систем баз даних, який можна стандартизувати, є інтерфейси. Нею було докладено чимало зусиль для визначення загальної архітектури системи баз даних. Запропонована цією групою архітектура стала кла-сичною та є актуальною й донині.

За принципами обробки даних БД класифікуються на централізовані і розподілені.

Централізована БД має на увазі, що робота з БД можлива тільки локально. Якщо комп'ютер працює в мережі, то доступ до інформації може здійснюватися віддалено з інших комп'ютерів мережі. Централізовані БД найбільш поширені в даний час. При цьому можливі кілька варіантів обробки даних.

Файл-серверна архітектура передбачає наявність в мережі сервера, на якому зберігаються файли централізованої БД. Відповідно до запитів користувачів файли з файл-сервера передаються на робочі станції користувачів, де і здійснюється основна частина обробки даних. Центральний сервер виконує в основному тільки роль сховища файлів, не беручи участь в обробці самих даних. Після завершення роботи користувачі копіюють файли з обробленими даними назад на сервер, звідки їх можуть взяти і обробити інші користувачі. Недоліки такої організації даних очевидні. При одночасному зверненні безлічі користувачів до одних і тих же даних продуктивність роботи різко падає, тому що необхідно дочекатися поки користувач, що працює з даними завершить роботу. В іншому випадку можливе затирання виправлень зроблених одним користувачем, змінами інших користувачів.

В основі концепції клієнт-сервер лежить ідея про те, що крім зберігання файлів БД, центральний сервер повинен виконувати основну частину обробки даних. Користувачі звертаються до сервера за допомогою спеціальної мови структурованих запитів (SQL, Structed Query Language), на котрому описується список завдань, які виконуються сервером. Запити приймаються сервером і породжують процеси обробки даних. У відповідь користувач отримує вже відпрацьований набір даних. Технологія клієнт-сервер дозволяє уникнути передачі по мережі величезних обсягів інформації, переклавши всю обробку на центральний сервер. Такий підхід також дозволяє уникнути конфліктів при редагуванні одних і тих же даних безліччю користувачів [2].

Трирівнева архітектура («Тонкий клієнт» - сервер додатків - сервер бази даних) функціонує в інтранет та Інтернет-мережах.

Клієнтська частина ("тонкий клієнт"), що взаємодіє з користувачем, являє собою HTML-сторінку в Web-браузері або Windows-додаток, що взаємодіє з Web-сервісами. Вся програмна логіка винесена на сервер додатків, який забезпечує

формування запитів до бази даних, що передаються на виконання сервера баз даних. Сервер додатків може бути Web-сервером або спеціалізованою програмою (див. Рис. 2.1).

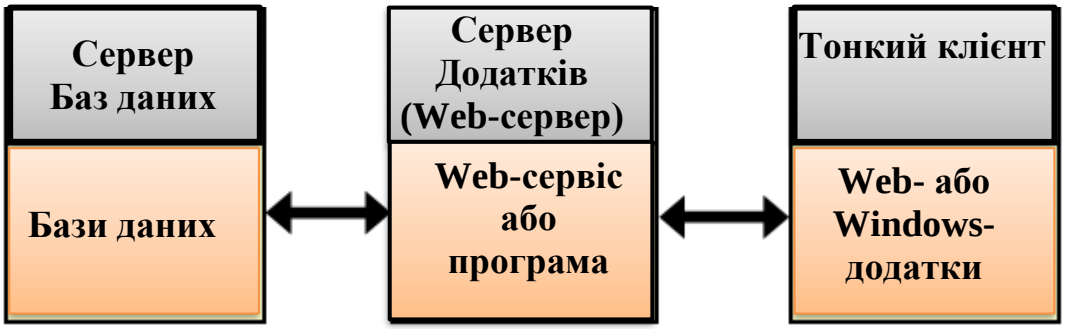

Рисунок 2.1 - Схема роботи з БД в трирівневої архітектурі

Розподілена БД розташовується на кількох комп'ютерах. Інформація на цих комп'ютерах може перетинатися і навіть дублюватися. Для управління такими БД призначена система управління розподіленими БД. Система приховує від користувачів доступу до даних, розташованим на інших комп'ютерах. Для користувача все виглядає так, як ніби вся інформація знаходиться на одному сервері.

2.2 Моделі даних

Виділяють наступні моделі даних:

- інфологічні;
- даталогічні;
- фізичні.

2.1. Інфологічна модель даних

Процес проектування БД починається зі створення інфологічної моделі.

Інфологічна модель даних - узагальнене неформальний опис створюваної бази даних, виконане з використанням природної мови, математичних формул, таблиць, графіків і інших засобів, зрозумілих всім людям, які працюють над проектуванням бази даних .

Інфологічна модель даних - узагальнене, Неприв'язані до будь-яких СУБД опис предметної області.

Інфологічна модель відображає реальний світ у деякі зрозумілі людині концепції, повністю незалежні від параметрів середовища зберігання даних. Тому інфологіческая модель не повинна змінюватися до тих пір, поки якісь зміни в реальному світі не зажадають зміни в ній деякого визначення, щоб ця модель продовжувала відображати предметну область.

Існує безліч підходів до побудови таких моделей: графові моделі, семантичні мережі, модель «сутність-зв'язок» та ін. [5, 6].

2.1.1. Семантичні мережі [2]

Семантична мережа (СМ) - це граф, дуги якого є відносини між вершинами (значеннями). Семантичні мережі з'явилися при вирішенні завдань розбору і розуміння сенсу природної мови. Приклад семантичної мережі для пропозиції типу "Постачальник здійснив поставку виробів на замовлення клієнта до 1 червня 2004 року в кількості 1000 штук" наведено на рис. 2.2.

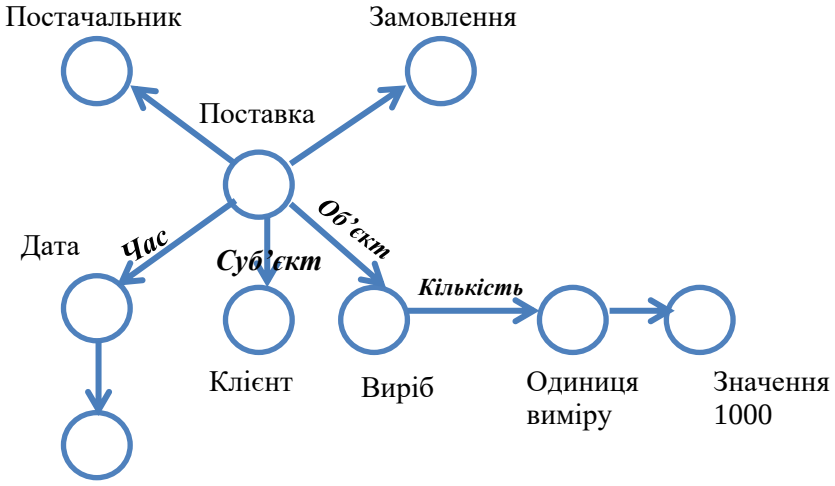

Рисунок 2.2 – Приклад семантичної мережі

На цьому прикладі видно, що між об'єктами *Постачальник* і *Поставка* визначено ставлення "агент", між об'єктами Виріб і Поставка визначено ставлення "об'єкт" і т.д.

Число відносин, використовуваних в конкретних семантичних мережах, може бути саме різне. Неповний список можливих відносин, що використовуються в семантичних мережах для розбору пропозицій, виглядає наступним чином.

Агент - це те, що (той, хто) викликає дію.

Об'єкт - це те, на що (на кого) спрямована дія. У реченні об'єкт часто виконує роль прямого доповнення, наприклад, "Роббі взяв жовту піраміду".

Інструмент - то засіб, який використовується агентом для виконання дії, наприклад, "Роббі відкрив двері за допомогою ключа".

Напівагент служить як підлеглий партнер головному агенту, наприклад, "Роббі зібрав кубики з допомогою Суззі".

Пункт відправлення і пункт призначення - це відправна і кінцева позиції при переміщенні агента або об'єкта.

Траєкторія - переміщення від пункту відправлення до пункту призначення: "Вони пройшли через двері по сходах на сходи".

Засіб доставки - то в чому або на чому відбувається переміщення: "Він завжди їде додому на метро".

Місцезнаходження - то місце, де відбулося (відбувається, буде відбуватися) дію, наприклад, "Він працював за столом".

Споживач - то особа, для якого виконується дія: "Роббі зібрав кубики для Суззі".

Сировина - це, як правило, матеріал, з якого щось зроблено або складається. Зазвичай сировину вводиться приводом з, наприклад, "Роббі зібрав Суззі з інтегральних схем".

Час - вказує на момент вчинення дії: "Він закінчив свою роботу пізно ввечері".

Найбільш типовий спосіб виведення в семантичних мережах (СС) - це спосіб зіставлення частин мережевої структури. Це видно на наступному простому прикладі, представленому на рис. 2.3.

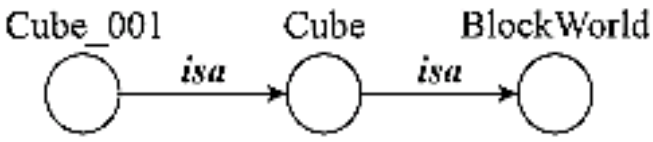

Рисунок 2.3 – Процедура співставлення СМ

Куб Cube належить світу BlockWorld.

Куб Cube_001 є різновид куба Cube.

Легко зробити висновок: Куб Cube_001 є частина світу BlockWorld.

2.2 Даталогічна модель

Інфологічна модель повинна бути відображена в даталогіческую модель, «зрозумілу» СУБД.

Даталогічна модель - опис мовою конкретної СУБД.

2.2.1. Ієрархічна модель

Спочатку стали використовувати ієрархічні даталогіческіе моделі. Ця модель являє собою сукупність пов'язаних елементів, що утворюють ієрархічну структуру. До основних понять ієрархії відносяться рівень, вузол і зв'язок. Вузлом називається сукупність атрибутів даних, що описують деякий об'єкт. Кожен вузол пов'язаний з одним вузлом вищого рівня і з будь-якою кількістю вузлів нижнього рівня. Винятком є вузол найвищого рівня, який не пов'язаний ні з одним вузлом вищого рівня.

Кількість дерев в БД визначається кількістю коренів дерев. До кожного запису БД існує єдиний шлях від кореневої запису.

Прикладом ієрархічної моделі даних може служити адреса. На першому рівні (корені дерева) лежить наша планета - Земля. На другому - країна. На третьому - регіон (республіка, край, район), потім - населений пункт, вулиця, будинок, квартира (рис.2.4).

Ще один приклад - це система доменних імен в Інтернеті.

Типовим представником СУБД (найбільш відомим і поширеним), заснованої на ієрархічній моделі, є Information Management System (IMS) фірми IBM. Перша версія з'явилася в 1968 р.

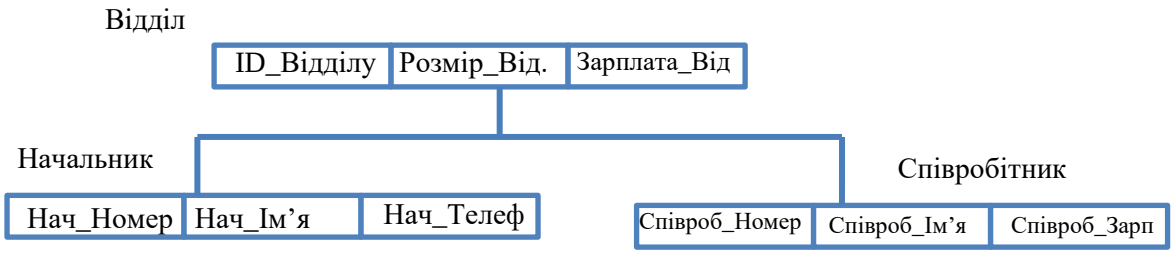

Рисунок 2.4 – Приклад ієрархічної структури

В даному прикладі відділ є предком для Начальник і Співробітники, а Начальник і Співробітники - нащадки Відділ. Між типами записи підтримуються зв'язки.

2.2.2. Мережева модель

В основі мережевої моделі даних лежать ті ж поняття, що і в основі ієрархічної моделі - вузол, рівень і зв'язок. Однак істотною відмінністю є те, що в ієрархічних структурах запис-нащадок повинна мати в точності одного предка; в мережевій структурі даних нащадок може мати будь-яке число предків.

Мережевий підхід до організації даних є розширенням ієрархічного.

У мережній моделі даних будь-який об'єкт може бути одночасно і головним, і підлеглим, і може брати участь в утворенні будь-якого числа взаємозв'язків з іншими об'єктами. Мережева БД складається з набору записів і набору зв'язків між цими записами, а якщо говорити більш точно - з набору примірників кожного типу із заданого в схемі БД набору типів запису і набору примірників кожного типу із заданого набору типів зв'язку (див. рис. 2.5).

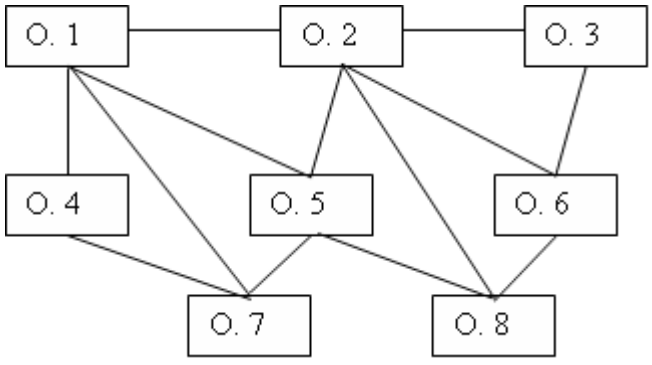

Рисунок 2.5 – Схема мережевої архітектури

Мережеві моделі також створювалися для мало ресурсних ЕОМ. Це досить складні структури, що складаються з "наборів" - пойменованих дворівневих дерев. "Набори" з'єднуються за допомогою "записів-зв'язок", утворюючи ланцюжки і т.д. При розробці мережевих моделей було вигадано безліч "маленьких хитрощів", що дозволяють збільшити продуктивність СУБД, але істотно ускладнити останні. Прикладний програміст повинен знати масу термінів, вивчити декілька внутрішніх

мов СУБД, детально представляти логічну структуру бази даних для здійснення навігації серед різних примірників, наборів, записів і т.п.

Типовим представником є Integrated Database Management System (IDMS) компанії Cullinet Software, Inc., призначена для використання на машинах основного класу фірми IBM під управлінням більшості операційних систем. Архітектура системи заснована на пропозиціях Data Base Task Group (DBTG), Комітету з мов програмування Conference on Data Systems Languages (CODASYL), організації, відповідальної за визначення мови програмування Кобол. Звіт DBTG був опублікований в 1971 р, а в 70-х роках з'явилося кілька систем, серед яких IDMS.

Складність практичного використання ієрархічних та мережевих БД змушувала шукати інші способи представлення даних. В кінці 60-х років з'явилися СУБД на основі інвертованих файлів, що відрізняються простотою організації і наявністю досить зручних мов маніпулювання даними.

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 у всіх сучасних реляційних БД, але в цих системах користувачі не мають безпосереднього доступу до інвертованим списками (індексам).

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

1. Рядки таблиць упорядковані системою в деякої фізичної послідовності.

2. Фізична упорядкованість рядків всіх таблиць може визначатися і для всієї БД (так робиться, наприклад, в Datacom / DB).

3. Для кожної таблиці можна визначити довільне число ключів пошуку, для яких будуються індекси. Ці індекси автоматично підтримуються системою, але явно видно користувачам.

Підтримуються два класи операторів:

1. Оператори, встановлюють адресу записи, серед яких:

1.1. прямі пошукові оператори (наприклад, знайти перший запис таблиці по деякому шляху доступу);

1.2. оператори, що знаходять запис в термінах відносної позиції від попередньої записи по деякому шляху доступу.

2. Оператори над адресованими записами

Типовий набір операторів:

– LOCATE FIRST - знайти перший запис таблиці T у фізичному порядку; повертає адресу записи;

– LOCATE FIRST WITH SEARCH KEY EQUAL - знайти перший запис таблиці T з заданим значенням ключа пошуку K; повертає адресу записи;

– LOCATE NEXT - знайти перший запис, наступну за записом з заданим адресою в заданому шляху доступу; повертає адресу записи;

– LOCATE NEXT WITH SEARCH KEY EQUAL - знайти следует запис таблиці T в порядку шляхи пошуку з заданим значенням K; має бути відповідність між використовуваним способом сканування і ключем K; повертає адресу записи;

– LOCATE FIRST WITH SEARCH KEY GREATER - знайти перший запис таблиці T в порядку ключа пошуку K со значенням ключового поля, великим заданого значення K; повертає адресу записи;

– RETRIVE - вибрати запис з вказаною адресою;

– UPDATE - оновити запис з вказаною адресою;

– DELETE - видалити запис з вказаною адресою;

– STORE - включити запис в зазначену таблицю; операція генерує адреса записи.

Загальні правила визначення цілісності БД відсутні. У деяких системах підтримуються обмеження унікальності значень деяких полів, але в основному все покладається на прикладну програму.

Однак такі СУБД мають ряд обмежень на кількість файлів для зберігання даних, кількість зв'язків між ними, довжину запису і кількість її полів [4, 7].

Лекція 3. Реляційна модель даних та її характеристики.

3.1 Реляційна структура даних

В кінці 60-х років з'явилися роботи, в яких обговорювалися можливості застосування різних табличних даталогіческіх моделей даних. Найбільш значною з них була стаття співробітника фірми IBM д-ра Едварда Кодда (Codd E.F., A Relational Model of Data for Large Shared Data Banks. CACM 13: 6, June 1970), де вперше був застосований термін «реляційна модель даних».

Будучи математиком за освітою, Е. Кодд запропонував використовувати для обробки даних апарат теорії множин (об'єднання, перетин, різниця, декартовій твір). Він показав, що будь-яке представлення даних зводиться до сукупності двовимірних таблиць особливого виду, відомого в математиці як відношення - relation.

Найменша одиниця даних реляційної моделі - це окреме атомарному (нерозкладних) для даної моделі значення даних.

Так, в одній предметної області прізвище, ім'я та по батькові можуть розглядатися як єдине значення, а в іншій - як три різних значення.

Доменом називається безліч атомарних значень одного і того ж типу.

Так, в табл. 3.1 домен пунктів відправлення (призначення) – безліч назв населених пунктів, а домен номерів рейсу - безліч цілих позитивних чисел.

Таблиця 3.1 – Таблиця даних «Рейс»

Номер рейсу	Дні тижня	Пункт відправлення	Час вильоту	Пункт призначення	Час прибуття	Тип літака	Вартість квитка
138	2_4_7	Баку	21.12	Москва	0.52	ИЛ-86	115.00
57	3_6	Єреван	7.20	Київ	9.25	ТУ-154	92.00
1234	2_6	Казань	22.40	Баку	23.50	ТУ-134	73.50
242	1 по 7	Київ	14.10	Москва	16.15	ТУ-154	57.00
86	2_3_5	Мінськ	10.50	Сочі	13.06	ИЛ-86	78.50
137	1_3_6	Москва	15.17	Баку	18.44	ИЛ-86	115.00
241	1 по 7	Москва	9.05	Київ	11.05	ТУ-154	57.00
577	1_3_5	Рига	21.53	Талін	22.57	АН-24	21.50
78	3_6	Сочі	18.25	Баку	20.12	ТУ-134	44.00
578	2_4_6	Талін	6.30	Рига	7.37	АН-24	21.50

Сенс доменів полягає в наступному. Якщо значення двох атрибутів беруться з одного і того ж домена, то, ймовірно, мають сенс порівняння, використовують ці два атрибути (наприклад, для організації транзитного рейсу можна дати запит «Видати рейси, в яких час вильоту з Москви в Сочі більше часу прибуття з Архангельська до Москви »). Якщо ж значення двох атрибутів беруться з різних доменів, то їх порівняння, ймовірно, позбавлене сенсу: чи варто порівнювати номер рейсу з вартістю квитка?

Ставлення на доменах D1, D2, ..., Dn складається з заголовка і тіла. На рис.

3.1 наведено приклад відносини для розкладу руху літаків. Ai - атрибути, Vi - значення атрибутів

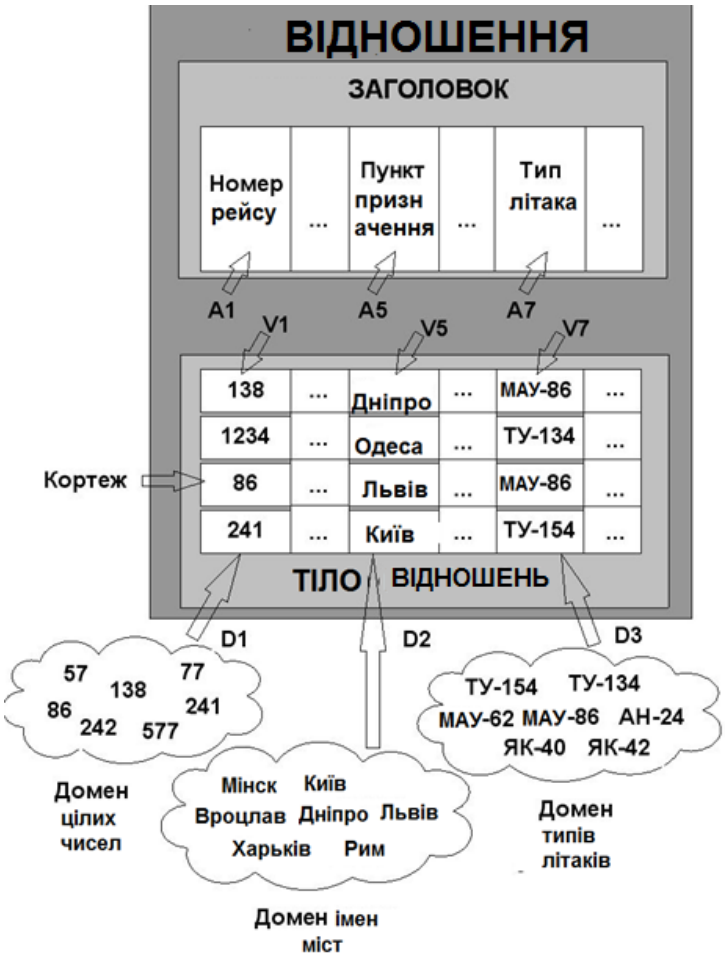

Рисунок 3.1 – Ставлення (зв'язок) з математичної точки зору

Заголовок відносини складається з фіксованих і атрибутів A1, A2, ..., An, що існує взаємно однозначна відповідність між цими атрибутами Ai і визначальними їх доменами Di (i = 1,2, ..., n).

Тіло відносини складається з мінливого в часі безлічі кортежів, де кожен кортеж складається в свою чергу з безлічі пар атрибут-значення (Ai: Vi), (i = 1,2, ..., n), по одній такій парі для кожного атрибута Ai в заголовку.

Для будь-якої заданої пари атрибут-значення (Ai: Vi) Vi є значенням з єдиного домену Di, який пов'язаний з атрибутом Ai.

Ступінь відносини - це число його атрибутів.

Ставлення ступеня один називають унарним, ступеня два - бінарним, ступені три - тернарного, ..., а ступеня n - n-арним. Ступінь відносини «Рейс» (рис. 3.2) дорівнює 8.

Кардинальне число або потужність відносини - це число його кортежів.

Потужність відносини «Рейс» дорівнює 10. Кардинальне число відносини змінюється в часі на відміну від його ступеня.

Оскільки ставлення - це безліч, а безлічі за визначенням не містять

співпадаючих елементів, то ніякі два кортежу відносини не можуть бути дублікатами один одного в будь-який довільно заданий момент часу.

Нехай R - відношення з атрибутами A1, A2, ..., An. Кажуть, що безліч атрибутів K = (Ai, Aj, ..., Ak) відносини R є можливим ключем R тоді і тільки тоді, коли задовольняються два незалежних від часу умови:

1. Унікальність: в довільний заданий момент часу ніякі два різних кортежу R не мають одного і того ж значення для Ai, Aj, ..., Ak.

2. Мінімальність: жоден з атрибутів Ai, Aj, ..., Ak не може бути виключений з K без порушення унікальності.

Кожне відношення має хоча б одним можливим ключем, оскільки щонайменше комбінація всіх його атрибутів задовольняє умові унікальності. Один з можливих ключів (вибраний довільним чином) приймається за його первинний ключ. Інші можливі ключі, якщо вони є, називаються альтернативними ключами.

Вищезазначені та деякі інші математичні поняття з'явилися теоретичною базою для створення реляційних СУБД, розробки відповідних мовних засобів і програмних систем, що забезпечують їх високу продуктивність, і створення основ теорії проектування баз даних.

Також на практиці широко використовуються неформальні еквіваленти цих понять: Ставлення - Таблиця, Кортеж - Рядок таблиці або Запис, Атрибут - Стовпець Таблиці або Поле.

При цьому приймається, що «запис» означає «екземпляр запису», а «поле» означає «ім'я і тип поля».

3.2 Побудова бази даних

Розглянемо приклад побудови інфологічної моделі бази даних «Харчування», де повинна зберігатися інформація про страви, їх щоденному споживанні, продуктах, з яких готуються ці страви, і постачальників цих продуктів. Інформація буде використовуватися кухарем і керівником невеликого підприємства громадського харчування, а також його відвідувачами.

За допомогою зазначених користувачів виділені наступні об'єкти і характеристики проектованої бази (рис.3.2):

1. Страви, для опису яких потрібні дані, що входять до їх кулінарні рецепти: номер страви (наприклад, з книги кулінарних рецептів), назва страви, вид страви (закуска, суп, гаряче і т.п.), рецепт (технологія приготування страви), вихід (вага порції), а також назва, калорійність і вага кожного продукту, що входить в страви;

2. Для кожного постачальника продуктів: найменування, місто і країна, назва поставленого товару, дата поставки і ціна на момент поставки.

3. Щоденне споживання страв (витрата): страва, кількість порцій, дата.

Аналіз дозволяє виділити:

1. незалежні сутності - Продукти, Страви, Постачальники, Міста;

2. залежні сутності - Склад страв, Поставки, Рецепти, Витрата страв.

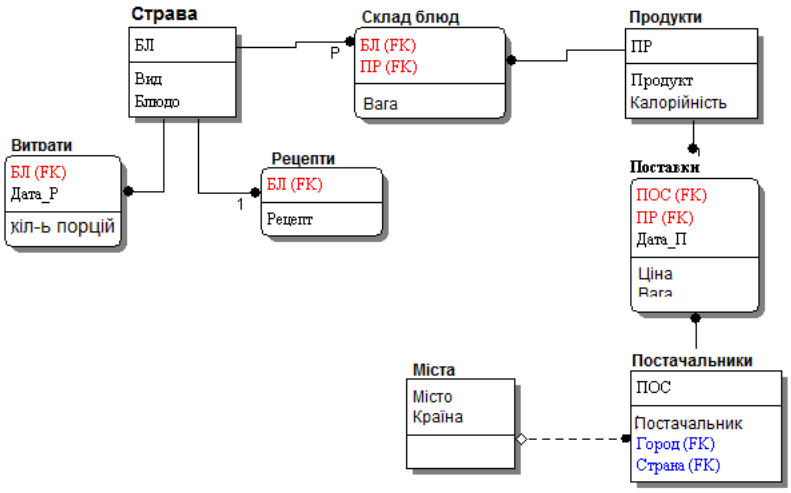

Рисунок 3.2 – Модель бази даних «Харчування»

У цих моделях Страва, Продукт і Постачальник - найменування, а БЛ, ПР і ПОС - цифрові коди страв, продуктів і постачальників продуктів.

3.3 Реляційна база даних

Реляційна база даних - це сукупність відносин, що містять всю інформацію, яка повинна зберігатися в БД.

Однак користувачі можуть сприймати таку базу даних як сукупність таблиць. Так на рис. 3.2 показані таблиці бази даних, побудовані по інфологічної моделі бази даних «Харчування» (рис. 3.2).

1. Кожна таблиця складається з однотипних рядків і має унікальне ім'я.

2. Рядки мають фіксоване число полів (стовпців) і значень (множинні поля і повторювані групи неприпустимі). Інакше кажучи, в кожній позиції таблиці на перетині рядка і стовпця завжди є в точності одне значення або нічого.

3. Рядки таблиці обов'язково відрізняються один від одного хоча б єдиним значенням, що дозволяє однозначно ідентифікувати будь-який рядок такої таблиці.

4. стовпців таблиці однозначно присвоюються імена, і в кожному з них розміщуються однорідні значення даних (дати, прізвища, цілі числа або грошові суми).

5. Повний інформаційний зміст бази даних представляється у вигляді явних значень даних і такий метод подання є єдиним. Зокрема, не існує будь-яких спеціальних «зв'язків» або покажчиків, що з'єднують одну таблицю з іншого. Так, зв'язку між рядком з БЛ = 2 таблиці «Страви» на рис. 3.5 і рядком з ПР = 7 таблиці «Продукти» (для приготування «Харчо» потрібен «Рис»), видається не за допомогою покажчиків, а завдяки існуванню в таблиці «Склад» рядки, в якій номер страви дорівнює 2, а номер продукту дорівнює 7.

6. При виконанні операцій з таблицею її рядки і стовпці можна обробляти в будь-якому порядку безвідносно до їх інформаційного змісту. Цьому сприяє наявність імен таблиць і їх стовпців, а також можливість виділення будь-якої їх

рядки або будь-якого набору рядків із зазначеними ознаками (наприклад, рейсів з пунктом призначення "Париж" і часом прибуття до 12 годин).

Страви

БЛ	Страва	Вид
1	Лобіо	Закуска
2	Харчо	Суп
3	Шашлик	Гаряче
4	Кава	Десерт

Расход

БЛ	Порций	Дата_Р
1	158	1/9/94
2	144	1/9/94
3	207	1/9/94
4	235	1/9/94
...	...	...

Продукти

ПР	Продукт	Калор.
1	Фасоль	3070
2	Цибуля	450
3	Масло	7420
4	Зелень	180
5	М’ясо	1660
6	Помідори	240
7	Рис	3340
8	Кава	2750

Рецепти

БЛ	Рецепт
1	Ломаную очищ
...	...

Состав

БЛ	ПР	Вага(г)
1	1	200
1	2	40
1	3	30
1	4	10
2	5	80
2	2	30
2	6	40
2	7	50
2	3	15
2	4	15
3	5	180
3	6	100
3	2	40
3	4	20
4	8	8

Постачальники

ПОС	Поставщик	Город
1	"Полісся"	Київ
2	"Наталка"	Київ
3	"Хуанхэ"	Пекін
4	"Лайма"	Рига
5	"Юрмала"	Рига
6	"Даугава"	Рига

Города

Город	Страна
Київ	Україна
Пекін	Китай

Поставки

ПОС	ПР	Вага (кг)	Ціна	Дата_П
1	6	120	0.45	27/8/94
1	3	50	1.82	27/8/94
1	2	50	0.61	27/8/94
2	2	100	0.52	27/8/94
2	5	100	2.18	27/8/94
2	4	10	0.88	27/8/94
3	1	250	0.37	24/8/94
3	7	75	0.44	24/8/94
3	8	40	2.87	24/8/94
4	3	70	1.56	30/8/94
5	5	200	2.05	30/8/94

Рисунок 3.2 – База даних «Харчування»

3.4 Управління реляційними базами даними

Прагнення до мінімізації числа таблиць для зберігання даних може привести до виникнення різних проблем при їх оновленні та будуть надані

рекомендації щодо розбиття деяких великих таблиць на кілька маленьких. Але як сформувати необхідний відповідь, якщо потрібні для нього дані зберігаються в різних таблицях?

Запропонувавши реляційну модель даних, Е. Кодд створив і інструмент для зручної роботи з відносинами - реляційну алгебру. Кожна операція цієї алгебри використовує одну або кілька таблиць (відносин) в якості її операндів і продукує в результаті нову таблицю, тобто дозволяє «розрізати» або «склеювати» таблиці (рис.3.3).

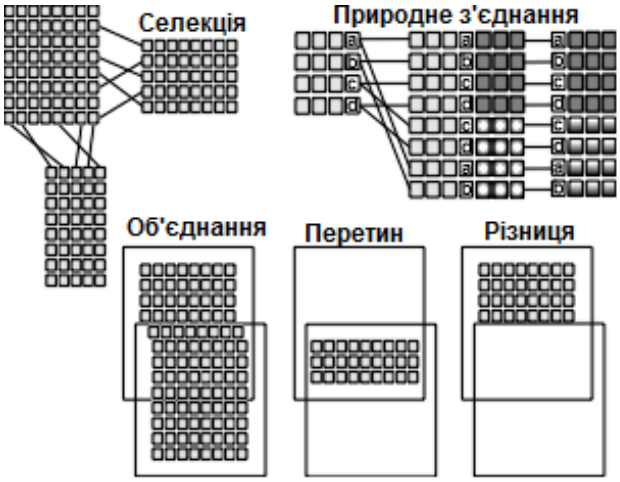

Рисунок 3.3 – Деякі операції реляційної алгебри

Створено мови управління (маніпулювання) даними, що дозволяють реалізувати всі операції реляційної алгебри і практично будь-які їх поєднання. Серед них найбільш поширений SQL (Structured Query Language - Структурізованний мову запитів).

За допомогою єдиного запиту на мові SQL можна з'єднати кілька таблиць в тимчасову таблицю і вирізати з неї необхідні рядки і стовпці (селекція і проекція).

Основна мета проектування БД - це скорочення надмірності збережених даних, а отже, економія обсягу використовуваної пам'яті, зменшення витрат на багаторазові операції оновлення надлишкових копій і усунення можливості виникнення протиріч через зберігання в різних місцях відомостей про одне й те ж об'єкті. Так званий, «чистий» проект БД («Кожен факт в одному місці») можна створити, використовуючи методологію нормалізації відносин. І хоча нормалізація повинна використовуватися на завершальній перевірочної стадії проектування БД, розглянемо причини, які змусили Кодда створити основи теорії нормалізації.

Припустимо, що проектування бази даних «Харчування» починається з виявлення атрибутів і підбору даних, зразок яких показаний в табл.3.2.

Цей варіант таблиці «Харчування» не є відношенням, так як більшість її рядків не атомарний. Атомарними є лише значення полів Блюдо, Вид, Рецепт (хоча він і великий), Працюй і Дата_Р інші ж поля таблиці 3.2 - множинні. Для додання таких даних форми відносини необхідно реконструювати таблицю. Найпростіше це

зробити за допомогою простого процесу вставки. Однак таке перетворення призводить до виникнення великого обсягу надлишкових даних.

Таблиця 3.2 – Дані для створення БД «Харчування»

Страва	Вид	Рецепт	Порції	Дата Р	Продукт	Калорійність	Вага (г)	Постачальник	Місто	Країна	Вага (кг)	Ціна ($)	Дата П
Лобио	Закуска	Лом.	158	1/9/94	Фасоль	3070	200	"Хуанхэ"	Пекін	Китай	250	0.37	24/8/94
					Цибуля	450	40	"Наталка"	Київ	Україна	100	0.52	27/8/94
					Масло	7420	30	"Лайма"	Рига	Латвія	70	1.55	30/8/94
					Зелень	180	10	"Даугава"	Рига	Латвія	15	0.99	30/8/94
Шашлик	Горяче	...	207	1/9/94	М’ясо	1660	180	"Юрмала"	Рига	Латвия	200	2.05	30/8/94
					Цибуля	450	40	"Полесье"	Київ	Україна	50	0.61	27/8/94
					Помідори	240	100	"Полесье"	Київ	Україна	120	0.45	27/8/94
					Зелень	180	20	"Даугава"	Рига	Латвія	15	0.99	30/8/94
Кава	Десерт	...	235	1/9/94	Кава	2750	8	"Хуанхэ"	Пекін	Китай	40	2.87	24/8/94

3.5 Недоліки проекту БД

Початківець проектувальник буде використовувати відношення «Харчування» з вставками в якості завершеною БД. Дійсно, навіщо розбивати відношення «Харчування» на кілька дрібніших відносин, якщо воно містить в собі всі дані? А розбивати треба тому, що при використанні універсального відносини виникає кілька проблем:

1. Надмірність. Дані практично всіх стовпців багаторазово повторюються. Повторюються і деякі набори даних (Блюдо-Від-Рецепт, Продукт-Калорійність, Постачальник-Місто-Країна). Небажано повторення рецептів, деякі з яких набагато більше рецепта «Лобіо». І вже зовсім погано, що всі дані про страву (включаючи рецепт) повторюються кожен раз, коли це блюдо включається в меню.

2. Потенційна суперечливість (аномалії оновлення). Внаслідок надмірності можна оновити адресу постачальника в одному рядку, залишаючи його незмінним в інших. Якщо постачальник кави повідомив про свій переїзд в Харбін і була оновлена рядок з продуктом кави, то у постачальника «Хуанхе» з'являється дві адреси, один з яких не актуальний. Отже, при оновленнях необхідно переглядати всю таблицю для знаходження і зміни всіх відповідних рядків.

3. Аномалії включення. В БД не може бути записаний новий постачальник («Нярінга», Вільнюс, Литва), якщо поставлений їм продукт (Огірки) не використовується ні в одній страві. Можна, звичайно, помістити невизначені значення в стовпці Страва, Вид, Працюй і Вага (г) для цього постачальника. Але якщо з'явиться блюдо, в якому використовується цей продукт, не забудемо ми видалити рядок з невизначеними значеннями?

З аналогічних причин не можна ввести і новий продукт (наприклад, Баклажани), який пропонує існуючий постачальник (наприклад, «Полісся»). А як

ввести нове блюдо, якщо в ньому використовується новий продукт («Краби»)?

4. Аномалії видалення. Зворотній проблема виникає при необхідності видалення всіх продуктів, що поставляються даними постачальником або всіх страв, які використовують ці продукти. При таких удаленіях будуть втрачені відомості про такий постачальника.

Багато проблем цього прикладу зникнуть, якщо виділити в окремі таблиці, як в розглянутому вище прикладі, відомості про страви, рецептах, витраті страв, продукти та їх постачальників, а також створити сполучні таблиці «Склад» і «Поставки».

Включення. Простим додаванням рядків (Постачальники; "Нярінга", Вільнюс, Литва) і (Поставки; "Нярінга", Вільнюс, Огірки, 40) можна ввести інформацію про новий постачальника. Аналогічно можна ввести дані про новий продукт (Продукти; Баклажани, 240) і (Поставки; "Полісся", Київ, Баклажани, 50).

Видалення. Видалення відомостей про деякі поставках або стравах не приводить до втрати відомостей про постачальників.

Оновлення. У таблиці 3.2 все ще багато повторюваних даних, що знаходяться в сполучних таблицях (Склад і Поставки). Отже, в даному варіанті БД збереглася потенційна суперечливість: для зміни назви постачальника з "Полісся" на "Дніпро" доведеться змінювати не тільки рядок таблиці Постачальники, але і безліч рядків таблиці Поставки. При цьому не виключено, що в БД будуть одночасно зберігатися: "Полісся", "скотарство", "Дніпро", "Дніпро" та інші варіанти назв. Крім того, що повторюються текстові дані (такі як назва страви "Рулет з телячої грудинки з сосисками і гарніром з різнобарвного пюре" або продукту "Ковбаса московська сирокопчена") істотно збільшують обсяг збережених даних.

Для виключення посилань на довгі текстові значення останні зазвичай нумерують: номер страви у великих кулінарних книгах, товари (продукти) в каталогах і т.д. Скористаємося цим прийомом для виключення надмірного дублювання даних і появи помилок при копіюванні довгих текстових значень (рис. 3.10). Тепер при зміні назви постачальника "Полісся" на "Дніпро" виправляється єдине значення в таблиці Постачальники. І навіть якщо воно вводиться з помилкою ("Дніпро"), то це не може вплинути на зв'язок між постачальниками і продуктами (в сполучній таблиці Поставки використовуються номери постачальників і продуктів, а не їх назви).

Лекція 4. Операції реляційної алгебри та реляційне числення.

4.1 Реляційна алгебра

Реляційна алгебра - замкнута система операцій над відносинами в реляційної моделі даних, або Реляці йна алгебра – відгалуження логіки першого порядку, множина відношень замкнених операторами. Оператори застосовуються до відношень, в результаті застосування отримується нове відношення.

В математиці, алгебра відношень є алгебраїчною структурою щодо математичної логіки та теорії множин.

РМД стала першою працездатною моделлю даних, оскільки мала ефективний інструментарій - операції реляційної алгебри. Основною одиницею обробки є відношення, а не його кортежі.

Реляційна алгебра включає дві групи операцій.

1. Традиційні операції над множинами (модифіковані з урахуванням того, що їх операндами є відношення) - об'єднання, перетин, різниця (віднімання), декартовій твір і розподіл.
2. Спеціальні реляційні операції - вибірка, проекція, з'єднання.

В інструкціях по теорії відносин показано, що безліч відносин замкнуто щодо деяких спеціальних операцій, тобто утворює разом з цими операціями абстрактну алгебру. Це найважливіша властивість відносин було використано в реляційної моделі для розробки мови маніпулювання даними, пов'язаного з вихідної алгеброю. Американський математик Е. Ф. Кодд в 1970 році вперше сформулював основні поняття і обмеження реляційної моделі. Набір основних алгебраїчних операцій по Кодду складається з восьми операцій, які діляться на два класи - теоретико-множинні операції та спеціальні реляційні операції. $\wedge \vee \notin$

Нехай задані два відношення $R_1 = \{ r_1 \}$, $R_2 = \{ r_2 \}$, де r_1 та r_2 – відповідно кортежі відносин R_1 і R_2,

До складу теоретико-множинних операцій входять операції:

1. об'єднання відносин $R_1 \cup R_2 = \{ r \mid r \in R_1 \vee r \in R_2 \}$. де r – кортеж нового відношення, $\vee$– операція логічного складання «АБО».
2. перетин відношення $R_2 = R_1 \cap R_2 = \{ r \mid r \in R_1 \wedge r \in R_2 \}$, де $\wedge$ – операція логічного множення (логічне «І»);
3. взяття різниці відношень $R_4 = RI \setminus R_2 = \{r \mid r \in R_1 \wedge r \notin R_2 \}$;
4. прямого добутку відношень.

Всі попередні операції не змінювали ступеня або арності відносин - це випливає з визначення еквівалентності схем відносин. Операція декартового добутку змінює ступінь результуючого відносини [7, 9].

Розширеним декартових добутком відношення R, степеня n зі схемою

$S_{R1}=(A_1,A_2...,A_n)$ і відношення R_2 степені m зі схемою $S_{R2}=(B_1,B_2, ... , B_m)$ називається відношення R_3 (позначається ступеня n+m зі схемою

$S_{R3} = (A_1, A_2, ... , A_n, B_1, B_2, ..., B_m)$, що містить кортежі, отримані зчепленням кожного кортежу r відношення R_1 з кожним кортежем q відношення R_2.

Тобто, якщо $R_1 = \{ r \}$, $R_2 = \{ q \}$, то $R_1 \otimes R_2 = \{(r, q) \mid r \in R_1 \wedge q \in R_2\}$.

Спеціальні реляційні операції включають:

- обмеження відносини;
- проекцію відносини;
- з'єднання відносин;
- розподіл відносин.

Не всі вони є незалежними, тобто деякі з цих операторів можуть бути виражені через інші реляційні оператори.

Крім того, до складу алгебри включається операція присвоювання, що дозволяє зберегти в базі даних результати обчислення алгебраїчних виразів, і операція перейменування атрибутів, що дає можливість коректно сформувати заголовок (схему) результуючого відносини.

Майже всі операції запропонованого вище набору володіють очевидною і простий інтерпретацією.

1. При виконанні операції об'єднання двох відносин здійснюється ставлення, що включає всі кортежі, що входять хоча б в одне з відносин-операндів.

2. Операція перетину двох відносин виробляє ставлення, що включає всі кортежі, що входять в обидва відносини-операнда.

3. Ставлення, що є різницею двох відносин включає всі кортежі, що входять до ставлення - перший операнд, такі, що жоден з них не входить до відношення, яке є другим операндом.

4. При виконанні прямого твори двох відносин здійснюється ставлення, кортежі якого є конкатенації (зчепленням) кортежів першого і другого операндів. Дуже часто операція розширеного декартова твори використовується для отримання деякого універсуму - т. Е. Ставлення, яке характеризує всі можливі комбінації між елементами окремих множин. Однак самостійного значення результат виконання операції зазвичай не має, він бере участь в подальшій обробці.

5. Результатом обмеження відносини по деякому умові є ставлення, що включає кортежі відносини-операнда, яке задовольняє цій умові.

6. При виконанні проекції відносини на заданий набір його атрибутів виробляється ставлення, кортежі якого виробляються шляхом знаходження відповідних значень з кортежів відносини-операнда.

7. При з'єднанні двох відносин по деякому умові утворюється результуюче ставлення, кортежі якого є конкатенації кортежів першого і другого відносин і задовольняють цій умові.

8. У операції реляційного поділу два операнда - бінарне і унарное відносини. Результуюче відношення складається з одноатрібутних кортежів, що включають значення першого атрибута кортежів першого операнда таких, що безліч значень другого атрибута (при фіксованому значенні першого атрибута) збігається з безліччю значень другого операнда.

9. Операція перейменування виробляє ставлення, тіло якого збігається з тілом операнда, але імена атрибутів змінені.

10. Операція присвоювання дозволяє зберегти результат обчислення реляційного вираження в існуючому відношенні БД.

Оскільки результатом будь-якої реляційної операції (крім операції присвоювання) є певний стосунок, то можна утворювати реляційні вирази, в яких замість відносини-операнда деякої реляційної операції знаходиться вкладене реляційне вираз (замкнутість реляційної алгебри).

Згідно Дейта реляційна модель складається з трьох частин, що описують різні аспекти реляційного підходу: структурної частини, маніпуляційної частини і цілісної частини.

У структурної частини моделі фіксується, що єдиною структурою даних, що використовується в реляційних БД, є нормалізоване n-арное відношення. По суті справи, на попередній лекції ми розглядали саме поняття і властивості структурної складової реляційної моделі.

У маніпуляційної частини моделі затверджуються два фундаментальних механізму маніпулювання реляційними БД - реляційна алгебра і реляційне числення. Перший механізм базується в основному на класичній теорії множин (з деякими уточненнями), а другий - на класичному логічному апараті обчислення предикатів першого порядку. Основною функцією маніпуляційної частини реляційної моделі є забезпечення заходів реляційної будь-якого конкретного мови реляційних БД: мова називається реляційних, якщо він має не меншу виразністю і потужністю, ніж реляційна алгебра або реляційне числення [7,9].

4.2 Операції реляційної алгебри

Операції реляційної алгебри також називають реляційними операціями.

Початковий набір з 8 операцій був запропонований Е. Коддом в 1970-і роки і включав як операції, які до цих пір використовуються (проекція, об'єднання і т.д.), так і операції, які не увійшли до вживання (наприклад, поділ відношень).

У процесі розвитку реляційної теорії і практики було запропоновано декілька нових реляційних операцій, наприклад полусоединеніе (SEMI-JOIN) і полуразность, або анти-полусоединеніе (ANTI-SEMI-JOIN), CROSS APPLY і OUTER APPLY, транзитивне замикання (TCLOSE) та ін

Оскільки багато операцій виразність один через одного, у складі реляційної алгебри можна виділити кілька варіантів базису (набору операцій, через який виразність всі інші). Найбільш відомий і строго певний базис (алгебра А) запропонований Крістофером Дейта і Х'ю Дарвеном.

Доведено, що реляційна алгебра і реляційне обчислення взаємно

еквіваленти.

4.2.1 Замкнутість реляційної алгебри

Реляційна алгебра являє собою набір таких операцій над відносинами, що результат кожної з операцій також є відносинам. Це властивість алгебри називається замкнутістю.

Операції над одним відношенням називаються унарний, над двома відношеннями - бінарними, над трьома - тернарних (такі практично невідомі).

Приклад унарний операції - проекція, приклад бінарної операції - об'єднання.

Реляційну операцію f можна представити функцією з відносинами в якості аргументів: R = f (R 1, R 2,..., R n)

Оскільки реляційна алгебра є замкнутою, в якості операндів в реляційні операції можна підставляти інші вирази реляційної алгебри (підходящі за типом):

R = f (f 1 (R 11, R 12,...), f 2 (R 21, R 22,...), ...)

У реляційних виразах можна використовувати вкладені вирази як завгодно складної структури.

Деякі реляційні операції, зокрема, операції об'єднання, перетину і віднімання, вимагають, щоб відносини мали збігаються (однакові) заголовки (схеми). Це означає, що збігаються кількість атрибутів, назви атрибутів і тип (домен) однойменних атрибутів.

Деякі відносини формально не є сумісними з-за відмінності в назвах атрибутів, але стають такими після застосування операції перейменування атрибутів.

Перерахуємо деякі операції реляційної алгебри, які представляють або історичний, або практичний інтерес. Всі операції перерахувати неможливо, оскільки будь-яка операція, що задовольняє визначенню реляційної, є частиною реляційної алгебри.

Перейменування є унарним оператором, що записується як $\rho_{a/b}(R)$. Результат застосування оператора ідентичний за винятком того, що поле в усіх кортежах перейменовується на поле *a*. Цей оператор застосовується для простого перейменування атрибута відношення, або самого відношення.

В результаті застосування операції перейменування отримуємо нове ставлення, зі зміненими іменами атрибутів.

Синтаксис: *R* RENAME $Atr_1, Atr_2,...$ AS $NewAtr_1, NewAtr_2,...$

де *R* - відношення $Atr_1, Atr_2,...$ - вихідні імена атрибутів $NewAtr_1, NewAtr_2,...$ - нові імена атрибутів

Об'єднання. Відношення з тим же заголовком, що і у сумісних за типом відношень *A* і *B,* і тілом, що складається з кортежів, які належать або *A,* або *B,* або обом відносинам.

Синтаксис:

A UNION *B*

Перетин. Відношення з тим же заголовком, що й у відносин A і B, і тілом,

що складається з кортежів, які належать одночасно обом відносин A і B. Синтаксис:A INTERSECT B

Віднімання. Відношення з тим же заголовком, що і у сумісних за типом відносин A і B, і тілом, що складається з кортежів, що належать відношенню A і не належать відношенню B.

Синтаксис: A MINUS B

Узагальнена вибірка це унарний оператор, що записується як $\sigma_{\varphi}(R)$, де φ є формулою числення висловлень, що складається із атомів, дозволених у звичайній вибірці та логічних операторів (кон'юнкції), (диз'юнкції) та (заперечення). Така вибірка вибирає всі кортежі із для яких істина.

Або, вібірка – це відношення з тим же заголовком, що і у відносини *A*, і тілом, що складається з кортежів, значення атрибутів яких при підстановці в умову *c* дають значення ИСТИНА. *C* являє собою логічне вираження, до якого можуть входити атрибути відносини *A* і / або скалярні вирази. Синтаксис:

A WHERE *c*

Об'єднання. Операція об'єднання є результат послідовного застосування операцій декартового твору та вибірки. Якщо у відносинах і є атрибути з однаковими найменуваннями, то перед виконанням з'єднання такі атрибути необхідно перейменувати.

Операція об'єднання - збирає разом коржеті декількох відношень у один кортеж , проте у результуюче відношення включаються лише рядки, що задовільняють певним співвідношенням між атрибутами з"єднання відповідних відношень таблиць.

Синтаксис: (A TIMES B) WHERE .

Основні положення реляційної алгебри є основою для реалізації роботи СКБД безпосередньо з базою даних.

До відношень можна застосувати систему операцій, що дозволяють отримати одні відносини з інших. Виняток становлять операції створення та заповнення таблиць, а також операції опису та перейменування стовпців. Результатом запиту до реляційної БД може бути нове ставлення, обчислена на основі наявних відносин.

Операції з даними в реляційній базі даних включають операції над рядками (кортежами відношень) та операції над відношеннями.

Операції, що виконуються на рівні кортежів, - вставка (додається новий рядок), вилучення (знищується рядок), поновлення (здійснюються зміни значень атрибутів у рядках).

Основною одиницею обробки даних у реляційній моделі є відношення (файл).

Ефективність реляційної системи управління базою даних визначається здатністю виконувати над відношеннями операції алгебри відношень. Основними операціями над відношеннями РБД є традиційні операції над множинами: об'єднання, переріз, різниця (віднімання), декартів добуток [4].

Розглянемо основні оператори мови реляційної алгебри (для наочності відношення доповнено колонкою "Кортеж").

Об'єднання - операція виконується над двома сумісними відношеннями R1, R2 (з ідентичною структурою – d1, d2,dn) (табл. 4.1 і 4.2).

Таблиця 4.1 – База даних «Семінар 1»

Кортеж	Прізвище та ініціали	Група
С11	Іванов І. П.	12
С12	Соломахін М. В.	25
С13	Кряк І. В.	40
С14	Вовчук В. І.	31

Таблиця 4.2 – База даних «Семінар 2»

Кортеж	Прізвище та ініціали	Група
С21	Іванов І. В.	40
С22	Дорошенко І. П.	12
С23	Якубів Н. З.	18

У результаті операції об'єднання будується нове відношення:$R_1 \cup R_2$.

Відношення R має той самий склад атрибутів і сукупність кортежів вихідних відношень. Причому в цю сукупність не включаються дублікати. Результат представлено в табл.4.3.

Таблиця 4.3 – БД «Семінар» з відношенням R

Кортеж	Прізвище та ініціали	Група
С11 (С22)	Іванов І. П.	12
С12	Соломахін М. В.	25
С13 (С21)	Крят І. В.	40
С14	Вовчук В. І.	31

С23	Якубів Н. З.	18

У нове відношення не ввійшли кортежі С21 і С22, оскільки вони дублюють кортежі С11 і С13.

Переріз - операція, яка виконується над двома сумісними відношеннями R1,R2 (табл. 4.2, табл.4.3) Результуюче відношення$R_1 \cap R_2$ містить однакові кортежі, які є в кожному з двох вихідних. Результат перетину має той самий склад атрибутів, що й у вихідних наведено в табл. 4.4.

Таблиця 4.4 – Результат БД «Семінар» після перетину відношень

Кортеж	Прізвище та ініціали	Група
С11 (С22)	Іванов І. П.	12
С13 (С21)	Крят І. В.	40

Різниця - операція виконується над двома сумісними відношеннями R1, R2 з ідентичним набором атрибутів. У результаті операції віднімання будується нове відношення RV = R1 - R2 з ідентичним набором атрибутів, яке містить лише ті кортежі відношення Rj, які не повторюються в другому відношенні R. Результат наведено в табл.4.5.

Таблиця 4.5 – Результат БД «Семінар» після операції різниця

Кортеж	Прізвище та ініціали	Група
С12	Соломахін М. В.	25
С14	Вовчук В. І.	31

Лекція 5. Апаратні та програмні складові для проектуваня БД.

5.1 Апаратні складові БД

Для роботи СУБД і програм необхідно деяке апаратне забезпечення. Воно може варіювати в дуже широких межах - від єдиного персонального комп'ютера чи одного мейнфрейма до мережі з багатьох комп'ютерів. Апаратне забезпечення залежить від вимог даної організації і СУБД, яка використовується. Одні СУБД призначені для роботи тільки з конкретними типами операційних систем чи устаткування, інші можуть працювати із широким колом апаратного забезпечення і різних операційних систем. Найбільш поширеною системою вважається така, що складається з мережі мінікомп'ютерів з одним центральним комп'ютером. На центральному комп'ютері працює серверна частина СУБД (backend), що облуговує і контролює доступ до бази даних. До нього мають доступ інші комп'ютери, які можуть бути розташовані і в інших регіонах. На них працюють клієнтські частини СУБД (frontend), що здійснюють взаємодію з користувачами. Подібна архітектура зветься клієнт/сервер (client-server), де сервером є комп'ютер із серверною астиною СУБД, а клієнтами – комп'ютери з клієнтськими частинами БД (рис.3.1).

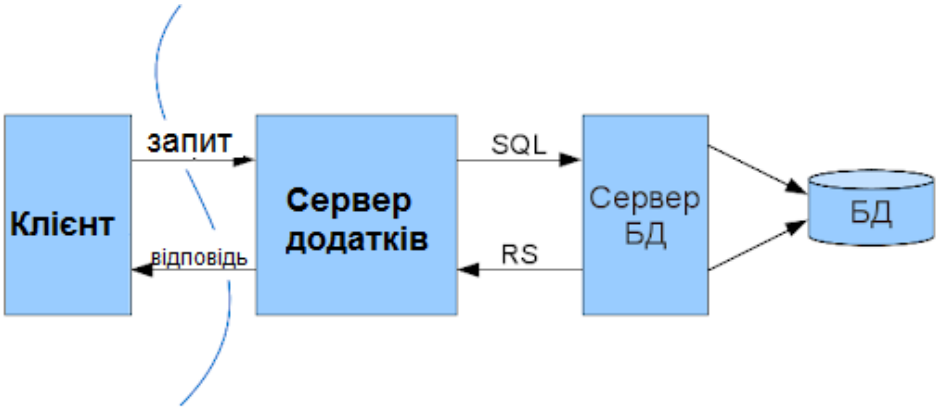

Рисунок 3.1 – Клієнт-серверна БД

Цей компонент охоплює програмне забезпечення самої СУБД і прикладних програм, разом з операційною системою, включаючи і мережне програмне забезпечення, якщо СУБД використовується в мережі. Як правило програми створюються мовами третього покоління, такими як C, Pascal чи VB, а також мовами четвертого покоління, таких як SQL, оператори яких впроваджують у програми на мовах третього покоління. Утім, СУБД може мати з власні інструменти четвертого покоління, призначені для швидкої розробки програм з використанням убудованих непроцедурних мов запитів, генераторів звітів, форм, графічних зображень і навіть повномасштабних програм. Використання інструментів четвертого покоління може істотно підвищити продуктивність системи і сприяти створенню більш зручних для обслуговування програм [8,9].

Імовірно, найважливішим компонентом середовища БД (з погляду кінцевих

користувачів) є дані. База даних містить як робочі дані, так і мета-дані, тобто "дані про дані". Структура бази даних називається схемою (schema). У нашому простому прикладі схема бази даних складається з чотирьох файлів, чи таблиць (table) (рис.3.2):

– офісні меблі;
– вітальні;
– дитячі меблі;
– кухні;
– дачні меблі.

Кожна з наведених таблиць має відповідні поля або атрибути. Крім своєї основної функції характеристики конкретних даних деякі атрибути виконують роль зв’язку з іншими таблицями.

Наприклад атрибут Власник у таблиці Об’єкти нерухомості моделює зв'язок між цією таблицею і таблицею Власник, тобто деякий власник володіє деякою нерухомістю, що здається в оренду.

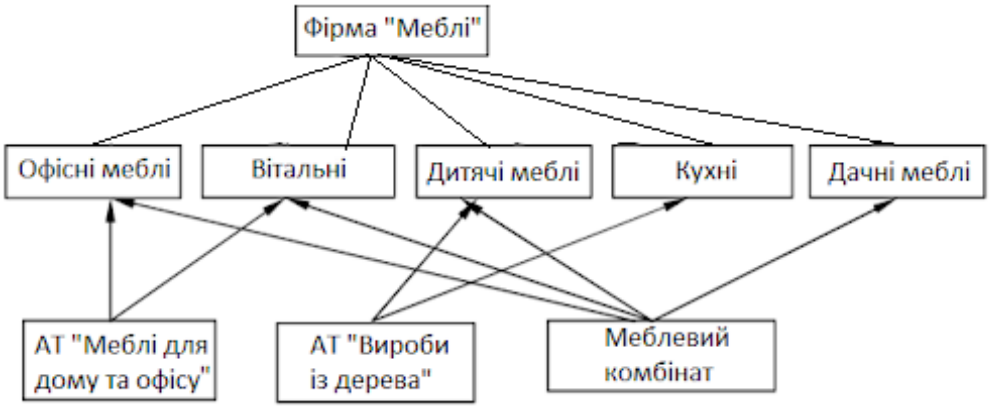

Рисунок 3.2 – Структура БД «Меблі»

У системному каталозі містяться наступні зведення:

– імена, типи і розміри елементів даних;
– імена зв'язків;
– обмеження цілісності даних;
– імена зареєстрованих користувачів, яким надані деякі права доступу до даних;
– індекси і структури збереження - наприклад, інвертовані файли чи дерева B+.

До процедур відносяться інструкції і правила, що повинні враховуватися при проектуванні і використанні бази даних. Користувачам і обслуговуючому персоналу бази даних необхідно надати документацію, що містить докладний опис процедур використання і супроводи даної системи, включаючи інструкції про правила виконання приведених нижче дій:

– реєстрація в БД;
– використання окремого інструмента БД чи програми;
– запуск та зупинка БД;
– створення резервних копій БД;

– обробка збоїв апаратного і програмного забезпечення, включаючи процедури ідентифікації компонента, що вийшов з ладу, виправлення компонента, що відмовив, (наприклад, за допомогою виклику фахівця з ремонту апаратного забезпечення), а також відновлення бази даних після усунення несправності;

– зміна структури таблиці, реорганізація бази даних, розміщеної на декількох дисках, способи поліпшення продуктивності і методи архівації даних на вторинних пристроях збереження.

Останнім, ще не розглянутим нами компонентом середовища БД, є користувачі системи.

Цей компонент докладно описується далі.

Існує чотири групи користувачів:

- адміністратори даних і баз даних;
- розроблювачі баз даних;
- прикладні програмісти;
- кінцеві користувачі.

База даних і БД є корпоративними ресурсами, якими варто управляти так само, як і будь-якими іншими ресурсами. Звичайне керування даними і базою даних передбачає управління і контроль за СУБД і розміщеними в неї даними. Адміністратор даних, чи АД (Data Administrator -DA), відповідає за керування даними, включаючи планування бази даних, розробку і супровід стандартів, бізнес правил і ділових процедур, а також за концептуальне і логічне проектування бази даних. АД консультує і дає свої рекомендації керівництву вищої ланки, контролюючи відповідності загального напрямку розвитку бази даних установленим корпоративним цілям.

Адміністратор бази даних, чи АБД (Database Administrator - DBA), відповідає за фізичну реалізацію бази даних, включаючи фізичне проектування і втілення проекту, за забезпечення безпеки і цілісності даних, за супровід операційної системи, а також за забезпечення максимальної продуктивності програм і користувачів. У порівнянні з АД, обов'язки АБД носять більш технічний характер, для чого необхідне знання конкретної СУБД. і системного оточення. В одних організаціях між цими ролями не робиться розходжень, а в інших важливість корпоративних ресурсів відбита саме у виділенні окремих груп персоналу з зазначеним колом обов'язків.

У проектуванні великих баз даних беруть участь два різних типи розроблювачів: розроблювачі логічної бази даних і розроблювачі фізичної бази даних. *Розроблювач логічної бази даних* займається ідентифікацією даних (тобто сутностей і їх атрибутів), зв'язків між даними і встановлює обмеження, що накладаються на збережені дані. Розроблювач логічної бази даних повинний мати всебічне і повне розуміння структури даних організації і їх бізнес правил. Бізнес правила описують основні характеристики даних з погляду організації. Нижче приводяться приклади типових бізнес правил:

1. Будь який співробітник не може відповідати одночасно більш ніж за десять об'єктів нерухомості що здаються в оренду чи продаються.

2. Будь який співробітник не має права продавати чи здавати в оренду

свою власну нерухомість.

3. Довірена особа не може виступати одночасно і як покупець, і як продавець нерухомості.

Для ефективної роботи розроблювач логічної бази даних повинен якомога раніше включити всіх передбачуваних користувачів бази даних у процес створення моделі даних і його робота поділяється на два етапи.

Концептуальне проектування бази даних, що зовсім не залежить від таких деталей її втілення, як конкретна цільова СУБД, програми, мови програмування чи будь-якої іншої фізичної характеристики.

Логічне проектування бази даних, що проводиться з урахуванням особливостей обраної моделі даних: реляційної, мережної, ієрархічної чи об'єктно-орієнтованій.

Розроблювач фізичної бази даних одержує готову логічну модель даних, займається її фізичною реалізацією, у тому числі:

– перетворенням логічної моделі даних у набір таблиць і обмежень цілісності даних;

– вибором конкретних структур збереження і методів доступу до даних, що забезпечують необхідний рівень продуктивності, при роботі з базою даних;

– проектуванням будь-яких необхідних мір захисту даних.

Багато етапів фізичного проектування бази даних у значній мірі залежать від обраної цільової СУБД, а тому може існувати кілька різних способів реалізації необхідної схеми. Якщо концептуальне і логічне проектування бази даних відповідає на запитання "що?", то фізичне проектування відповідає на запитання "як?". Для рішення цих задач вимагаються різні навички роботи, якими найчастіше володіють різні люди.

Відразу після створення бази даних варто приступити до розробки програм, що надають користувачам необхідні їм функціональні можливості. Саме цю роботу і виконують прикладні програмісти. Звичайно прикладні програмісти працюють на основі специфікацій, створених системними аналітиками. Як правило, кожна програма містить деякі оператори, що вимагають від БД виконання визначених дій з базою даних - наприклад, такі як витяг, вставка, чи відновлення видалення даних. Як уже згадувалося в попередньому розділі, ці програми можуть створюватися на різних мовах програмування третього чи четвертого покоління.

Користувачі є клієнтами бази даних - вона проектується, створюється і підтримується для того, щоб обслуговувати їх інформаційні потреби. Користувачів можна класифікувати за способами використання ними системи.

Наївні користувачі звичайно навіть і не підозрюють про наявність СУБД. Вони звертаються до бази даних за допомогою спеціальних програм які дозволяють у максимальній мірі спростити операції, які вони виконують. Такі користувачі ініціюють виконання операцій у бази даних, способом введення найпростіших команд чи вибираючи команди меню. Це значить, що таким користувачам не потрібно нічого знати про базу даних БД.

Досвідчені користувачі. Досвідчені кінцеві користувачі, що знайомі зі

структурою бази даних і можливостями СУБД. Для виконання необхідних операцій вони можуть використовувати таку мову запитів високого рівня, як SQL. А деякі досвідчені користувачі можуть навіть створювати власні прикладні програми.

5.2 Програмні складові БД

Розглянемо програмні продукти проектування БД.

MySQL - вільна система управління базами даних. MySQL є власністю компанії Oracle Corporation, що отримала її разом з поглиненою Sun Microsystems, яка здійснює розробку і підтримку додатку. Розповсюджується під GNU General Public License і під власною комерційною ліцензією, на вибір.

Крім цього розробники створюють функціональність на замовлення ліцензійних користувачів, саме завдяки такому замовленню майже в найраніших версіях з'явився механізм реплікації.

Цю систему управління базами даних з відкритим кодом було створено як альтернатива комерційним системам. MySQL із самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL - одна з найпоширеніших систем управління базами даних. Вона використовується, у першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування.

MySQL є рішенням для малих і середніх додатків. Зазвичай MySQL використовується як сервер, до якого звертаються локальні або віддалені клієнти, проте до дистрибутиву входить бібліотека внутрішнього сервера, що дозволяє включати MySQL до автономних програм. Вихідні коди сервера компілюються на багатьох платформах. Найповніше можливості сервера виявляються в UNIX-системах, де є підтримка багатонитевості, що підвищує продуктивність системи в цілому.

Гнучкість СУБД MySQL забезпечується підтримкою великої кількості типів таблиць: користувачі можуть вибрати як таблиці типу MyISAM, що підтримують повнотекстовий пошук, так і таблиці InnoDB, що підтримують транзакції на рівні окремих записів. Більш того, СУБД MySQL поставляється із спеціальним типом таблиць EXAMPLE, що демонструє принципи створення нових типів таблиць. Завдяки відкритій архітектурі й GPL-ліцензуванню, в СУБД MySQL постійно з'являються нові типи таблиць. MySQL характеризується великою швидкістю, стійкістю і простотою використання [10,11].

Для некомерційного використання MySQL є безкоштовною.

Можливості сервера MySQL:

– простота у встановленні та використанні;

– підтримується необмежена кількість користувачів, що одночасно працюють із БД;

– кількість рядків у таблицях може досягати 50 млн.;

– висока швидкість виконання команд;

– наявність простої та ефективної системи безпеки.

PostgreSQL - об'єктно-реляційна система управління базами даних. Є альтернативою як комерційним СУБД (Oracle Database, Microsoft SQL Server, IBM DB2 та інші), так і СУБД з відкритим кодом (MySQL, Firebird, SQLite).

Порівняно до інших проектів з відкритим кодом, такими як Apache, FreeBSD або MySQL, PostgreSQL не контролюється якоюсь однією компанією, її розробка можлива завдяки співпраці багатьох людей та компаній, які хочуть використовувати цю СУБД та впроваджувати у неї найновіші досягнення.

СУБД Oracle - це найпотужніший програмний комплекс, що дозволяє створювати додатки будь-якої складності. Ядром цього комплексу є база даних, що зберігає інформацію, кількість якої за рахунок наданих засобів масштабування практично безмежна. З високою ефективністю працювати з цією інформацією одночасно може практично будь-яка кількість користувачів (за наявності достатніх апаратних ресурсів), не проявляючи тенденції до зниження продуктивності системи при різкому збільшенні їхньої кількості.

Механізми масштабування в СУБД Oracle останньої версії дозволяють безмежно збільшувати потужність і швидкість роботи сервера Oracle і своїх додатків, просто додаючи нові й нові вузли кластеру. Це не вимагає зупинки працюючих додатків, не вимагає переписування старих додатків, розроблених для звичайної одномашинної архітектури. Крім того, вихід з ладу окремих вузлів кластера також не призводить до зупинки програми.

Вбудовування до СУБД Oracle JavaVM, повномасштабної підтримки серверних технологій (Java Server Pages, Java-сервлети, модулі Enterprise JavaBeans, інтерфейси прикладного програмування CORBA), призвели до того, що Oracle на сьогоднішній день де-факто є стандартом СУБД для Internet.

Ще однією складовою успіху СУБД Oracle є те, що вона поставляється практично для всіх існуючих на сьогодні операційних систем. Працюючи під Sun Solaris, Linux, Windows або на інший операційній системі з продуктами Oracle не буде виникати ніяких проблем у роботі. СУБД Oracle однаково добре працює на будь-якій платформі. Таким чином, компаніям, які розпочинають роботу з продуктами Oracle не доводиться міняти мережеве оточення. Існує лише невелика кількість відмінностей при роботі з СУБД, обумовлених особливостями тієї або іншої операційної системи. У цілому ж це завжди та ж сама безпечна, надійна і зручна СУБД Oracle [12].

Microsoft SQL Server - система управління реляційними базами даних, розроблена корпорацією Microsoft. Основна використовувана мова запитів - Transact-SQL, створена спільно Microsoft та Sybase. Transact-SQL є реалізацією стандарту ANSI / ISO щодо структурованої мови запитів (SQL) із розширеннями. Використовується для роботи з базами даних розміром від персональних до великих баз даних масштабу підприємства, конкурує з іншими СУБД у цьому сегменті ринку.

При взаємодії з мережею Microsoft SQL Server і Sybase ASE використовують протокол рівня додатків під назвою Tabular Data Stream (TDS, протокол передачі табличних даних). Протокол TDS також був реалізований у проекті FreeTDS з метою забезпечити різні додатки можливістю взаємодії з

базами даних Microsoft SQL Server і Sybase.

Для забезпечення доступу до даних Microsoft SQL Server підтримує Open Database Connectivity (ODBC) - інтерфейс взаємодії додатків з СУБД. SQL Server надає можливість підключення користувачів через веб-сервіси, що використовують протокол SOAP. Це дозволяє клієнтським програмам, не призначеним для Windows, кросплатформно з'єднуватися з SQL Server [4, 11].

Лекція 6. Мова запитів SQL.

6.1 Стандарт мови запитів SQL

У реалізаціях конкретних реляційних БД зараз не використовується в чистому вигляді ні реляційна алгебра, ні реляційне числення. Фактичним стандартом доступу до реляційних даних стала мова SQL (Structured Query Language). Мова SQL являє собою суміш операторів реляційної алгебри і виразів реляційного числення, що використовує синтаксис, близький до фраз англійської мови і розширений додатковими можливостями, відсутніми в реляційної алгебри та реляційному численні. Взагалі, мова доступу до даних називається реляційно повним, якщо він по виразною силою не поступається реляційної алгебри (або, що-те ж саме, реляційному обчисленню), тобто будь-який оператор реляційної алгебри може бути виражений засобами цієї мови. Саме таким і є мова SQL.

SQL (Structured Query Language) – Структурований Мова Запитів - стандартна мова запитів по роботі з реляційними БД. Робота була розпочата відразу після появи статті Е.Кодда в 1970р. в лабораторіях компанії IBM для перевірки можливостей реляційної моделі. Основні віхи становлення і розвитку мови SQL:

СУБД System R - експериментальна дослідницька система з мовою SEQUEL (пізніше SQL), створена IBM:

SQL в комерційних реалізаціях:

–1979 - Oracle (Relation Software Inc.- Oracle corp.;

–1981-1982 - DB2 (IBM), Ingres - CA-OpenIngres (Relation Technology Inc. - Computer Associates);

–1984 - Informix (Informix Sofrware);

–1986 - Sybase (Sybase Corp.).

Стандартизація SQL:

–міжнародний стандарт 1989 р (SQL 1);

–міжнародний стандарт 1992 р (SQL 2) ;

–стандарт 1998 р (SQL 3).

А навіщо взагалі потрібні ці стандарти? Навіщо їх винаходять і чому треба вивчати їх? Текст стандарту SQL 2 займає 600 станиць сухого формального тексту, це дуже багато, і здається, що це просто підступи розробників стандартів, а не те, що необхідно, рядовим розробникам. Однак жоден серйозний розробник, який працює з базами даних, не повинен ігнорувати стандарт, і для цього існують вагомі причини.

Стандарти - це вірний орієнтир для розробників, так як всі постачальники СУБД в своїх перспективних розробках обов'язково слідують стандарту, і можна бути уве¬ренним, що врешті-решт стандарт буде реалізований практично у всіх перспективних СУБД. Так сталося зі стандартом SQL 1, так відбувається зі стандартом SQL 2, і так буде відбуватися зі стандартом SQL 3.

З використанням будь-яких стандартів зв'язані не лише багаточисельні і сповна очевидні переваги, але і певні недоліки. Перш за все, стандарти

направляють в певне русло розвиток відповідної індустрії; в разі мови SQL наявність твердих засадничих принципів наводить, кінець кінцем, до сумісності його різних реалізацій і сприяє як підвищенню переносимості програмного забезпечення і баз даних в цілому, так і універсальності роботи адміністраторів баз даних. З іншого боку, стандарти обмежують гнучкість і функціональні можливості конкретної реалізації. Під реалізацією мови SQL розуміється програмний продукт SQL відповідного виробника. Для розширення функціональних можливостей багато розробників, що дотримуються прийнятих стандартів, додають до стандартної мови SQL різні розширення. Слід зазначити, що стандарти вимагають від будь-якої закінченої реалізації мови SQL наявності певних характеристик і у загальних рисах відображають основні тенденції, які не лише наводять до сумісності між всіма конкуруючими реалізаціями, але і сприяють підвищенню значущості програмістів SQL і користувачів реляційних баз даних на сучасному ринку програмного забезпечення.

Всі конкретні реалізації мови декілька відрізняються один від одного.

На користь самих же виробників гарантувати, аби їх реалізація відповідала сучасним стандартам ANSI в частині переносимості і зручності роботи користувачів. Проте, кожна реалізація SQL містить удосконалення, що відповідають вимогам того або іншого сервера баз даних. Ці удосконалення або розширення мови SQL є додатковими командами і опціями, що є додаваннями до стандартного пакету і доступні в даній конкретній реалізації.

Сьогодні мова SQL підтримується багатьма десятками СУБД різних типів, розроблених для найрізноманітніших обчислювальних платформ, починаючи від персональних комп'ютерів і закінчуючи мейнфреймами.

Всі мови маніпулювання даними, створені для багатьох СУБД до появи реляційних баз даних, були орієнтовані на операції з даними, представленими у вигляді логічних записів файлів. Зрозуміло, це вимагало від користувача детального знання організації зберігання даних і серйозних зусиль для вказівки того, які дані необхідні, де вони розміщуються і як їх отримати.

Дана мова SQL орієнтована на операції з даними, представленими у вигляді логічно взаємозв'язаних совокупностей таблиць-стосунків.

Найважливіша особливість його структур – орієнтація на кінцевий результат обробки даних, а не на процедуру цієї обробки. Мова SQL сам визначає, де знаходяться дані, індекси і навіть які найбільш ефективні послідовності операцій слід використовувати для здобуття результату, а тому вказувати ці деталі в запиті до бази даних не потрібно [12, 13].

6.2 Типи команд SQL

Реалізація в SQL концепції операцій, орієнтованих на табличне представлення 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. Основні категорії команд мови SQL призначені для виконання різних функцій, включаючи побудову об'єктів бази

даних і маніпулювання ними, початкове завантаження даних в таблиці, оновлення і видалення існуючої інформації, виконання запитів до бази даних, управління доступом до неї та її загальне адміністрування [4].

Основні категорії команд мови SQL:

- DDL – мова визначення даних;
- DML – мова маніпулювання даними;
- DQL – мова запитів;
- DCL – мова управління даними;
- команди адміністрування даних;
- команди управління транзакціями.

6.2.1 Визначення структур бази даних (DDL)

На відміну від реляційної алгебри, де були представлені тільки операції запитів до БД, SQL є повною мовою, в ньому присутні не тільки операції запитів, але і оператори, відповідні DDL - Data Definition Language - мови опису даних. Крім того, мова містить оператори, пред¬назначенние для управління (адміністрування) БД.

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

6.2.2 Маніпулювання даними (DML)

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

6.2.3 Вибірка даних (DQL)

Мова запитів DQL найбільш відома користувачам реляційної бази даних, не дивлячись на те, що він включає одну команду: SELECT. Ця команда разом зі своїми багаточисельними опціями і пропозиціями використовується для формування запитів до реляційної бази даних.

6.2.4 Мова управління даними (DCL)

Команди управління даними дозволяють управляти доступом до інформації, бази даних, що знаходиться усередині. Як правило, вони використовуються для створення об'єктів, пов'язаних з доступом до даних, а також служать для контролю над розподілом привілеїв між пользователями.

Команди управління даними наступні: GRANT, REVOKE.

6.2.5 Команди адміністрування даних

За допомогою команд адміністрування даних користувач здійснює контроль за виконуваними діями і аналізує операції бази даних; вони також можуть

виявитися корисними при аналізі производительности системи. Не слід плутати адміністрування даних з адмініструванням бази даних, яке є загальним управлінням базою даних і має на увазі використання команд всіх рівнів.

SQL містить розділи, представлені в таблиці 6.1 [4, 10].

Таблиця 6.1 – SQL розділи

Оператор	**Суть**	**Дія**
Оператори визначення даних DDL		
CREATE TABLE	створити таблицю	Створює нову таблицю в БД
DROP TABLE	видалити таблицю	Видаляє таблицю з БД
ALTER TABLE	змінити таблицю	Змінює структуру існуючої таблиці або обмеження цілісності, що задаються для даної таблиці
CREATE VIEW	створити уявлення	Створить віртуальну таблицю, відповідну деякого SQL-запит
ALTER VIEW	змінити уявлення	Змінює раніше створене подання
DROP VIEW	видалити уявлення	Видаляє раніше створене подання
CREATE INDEX	створити індекс	Створює індекс для деякої таблиці для забезпечення швидкого доступу по атрибутам, що входять в індекс
DROP INDEX	видалити індекс	Видаляє раніше створений індекс
Оператори маніпулювання даними Data Manipulation Language		
DELETE	Видалити рядки	Видаляє одну або кілька рядків, відповідних умовам фільтрації, з базової таблиці. Застосування оператора узгоджується з принципами підтримки цілісності, тому цей оператор не завжди може бути виконаний коректно, навіть якщо синтаксично він записаний правильно
SELECT	Вибрати рядки	Оператор, який замінює всі оператори реляційної алгебри і дозволяє сформувати

Оператор	Суть	Дія
		результуюче відношення, відповідне запитом
Засоби управління транзакціями		
COMMIT	Завершити транзакцію	Завершити комплексну взаємозв'язану обробку інформації, об'єднану в транзакцію
ROLLBACK	Відкотити транзакцію	Скасувати зміни, проведені в ході виконання транзакції
SAVEPOINT	Зберегти проміжну крапку виконання транзакції	Зберегти проміжний стан БД, позначити його для того, щоб можна було в подальшому до нього повернутися
Засоби адміністрування даних		
ALTER DATABASE	змінити БД	Змінити набір основних об'єктів в базі даних, обмежень, що стосуються бази даних
ALTER DBAREA	Змінити область зберігання БД	Змінити раніше створену область зберігання
ALTER PASSWORD	змінити пароль	Змінити пароль для всієї бази даних
CREATE DATABASE	створити БД	Створити нову базу даних, визначивши основні параметри для неї
CREATE DBAREA	Створити область зберігання	Створити нову область зберігання і зробити її доступною для розміщення даних
DROP DATABASE	видалити БД	Видалити існуючу базу даних (тільки в тому випадку, коли ви маєте право виконати цю дію)
DROP DBAREA	Видалити область зберігання БД	Видалити існуючу область зберігання (якщо в ній на зараз не розташовуються активні дані)
GRANT	надати права	Надати права доступу на ряд дій над деякими об'єктом БД
REVOKE	позбавити прав	Позбавити прав доступу до деякого об'єкту або деяких дій над об'єктом

Оператор	Суть	Дія
Програмний SQL		
DECLARE	Визначає курсор для запиту	Задає деякий ім'я і визначає пов'язаний з ним запит до БД, який відповідає віртуального набору даних
OPEN	відкрити курсор	Формує віртуальний набір даних, відповідає опису зазначеного курсору і поточному стану БД
FETCH	Вважати рядок з безлічі рядків, визначених курсором	Зчитує чергову рядок, задану параметром команди з віртуального набору даних, відповідає відкритому курсору
CLOSE	Закрити курсор	Припиняє доступ до віртуального набору даних, відповідному зазначеним курсору
PREPARE	Підготувати оператор SQL до динамічного виконання	Згенерувати план виконання запиту, відповідного заданому оператору SQL
EXECUTE	Виконати оператор SQL, раніше підготовлений до динамічного виконання	Виконує раніше підготовлений план запиту

Лекція 7. Структура мови SQL. Типи даних.

7.1 Типи даних

Поняття тип даних в реляційної моделі даних повністю адекватно поняттю типу даних в мовах програмування. Зазвичай в сучасних реляційних БД допускається зберігання символьних, числових даних, бітових рядків, спеціалізованих числових даних (таких як "гроші"), а також спеціальних "темпоральних" даних (дата, час, часовий інтервал). Досить активно розвивається підхід до розширення можливостей реляційних систем абстрактними типами даних (відповідними можливостями володіють, наприклад, системи сімейства Ingres / Postgres).

Ведення на попередній лекції поняття домену більш специфічно для баз даних, хоча і має деякі аналогії з підтипами в деяких мовах програмування. У найзагальнішому вигляді домен визначається завданням деякого базового типу даних, до якого відносяться елементи домену, і довільного логічного виразу, який застосовується до елемента типу даних. Якщо обчислення цього логічного виразу дає результат "істина", то елемент даних є елементом домену.

Найбільш правильною інтуїтивною трактуванням поняття домену є розуміння домену як допустимого потенційного безлічі значень даного типу. Наприклад, домен "Прізвища" в прикладі попередньої лекції визначено на базовому типі рядків символів, але в число його значень можуть входити тільки ті рядки, які можуть зображати прізвище (зокрема, такі рядки не можуть починатися з м'якого знака).

Слід зазначити також семантичне навантаження поняття домену: дані вважаються порівнянними тільки в тому випадку, коли вони відносяться до одного домену. Наприклад, щодо «Облік пропусків студентів ВНЗ» значення доменів "Номери перепусток" і "Номери груп" належать до типу цілих чисел, але не є порівнянними. Зауважимо, що в більшості реляційних СУБД поняття домену не використовується, хоча в Oracle V.7 і в InterBase воно вже підтримується.

У мові SQL підтримуються наступні типи даних:

–CHARACTER (n) или CHAR (n) – символьні рядки постійної довжини в n символів. При завданні даного типу під кожне значення завжди відводиться n символів, і якщо реальне значення займає менш, ніж n символів, то СУБД автоматично доповнює відсутні символи пробілами;

– NUMERIC [(n, m)] - точні числа, тут n - загальна кількість цифр в чіс¬ле, m - кількість цифр коду точки;

– DECIMAL [(n, m)] - точні числа, тут n - загальна кількість цифр в чіс¬ле, m - кількість цифр коду точки;

– DEC [(n, m)] - те саме, що і DECIMAL [(n, m)];

– INTEGER або INT - цілі числа;

– SMALLINT - цілі числа меншого діапазону.

Невелике пояснення: незважаючи на те, що в стандарті SQL 1 не визначається точно, що мається на увазі під типом INT і SMALLINT (це віддано на

відкуп реалізації), зазначено тільки співвідношення між цими типами даних, в більшості реалізацій тип даних INTEGER відповідає цілим числам, збереженим в чотирьох байтах, а SMALLINT - відповідає цілим числам, збереженим в двох байтах. Вибір одного з цих типів визначається розміром числа.

– FLOAT [(n)] - числа великої точності, збережені в формі з плаваючою точ¬кой. Тут n - число байтів, що резервуються під зберігання одного числа. Діа¬пазон чисел визначається конкретною реалізацією;

– REAL - речовинний тип чисел, який відповідає числам з плаваючою точкою, меншою точності, ніж FLOAT;

– DOUBLE PRECISION специфікує тип даних з певною в реалізації точністю більшої, ніж визначена в реалізації точність для REAL.

У стандарті SQL 92 додані наступні типи даних:

– VARCHAR (n) - рядки символів змінної довжини;
– NCHAR (N) - рядки локалізованих символів постійної довжини;
– NCHAR VARYING (n) - рядки локалізованих символів змінної довжини;
– BIT (n) - рядок бітів постійної довжини;
– BIT VARYING (n) - рядок бітів змінної довжини;
– DATE - календарна дата;
– TIMESTAMP (точність) - дата і час;
– INTERVAL - часовий інтервал.

Більшість комерційних СУБД підтримують ще додаткові типи даних, які не специфіковані в стандарті. Так, наприклад, практично всі СУБД в тому чи іншому вигляді підтримують тип даних для подання неструктурованого тексту великого обсягу. Цей тип аналогічний типу MEMO в настільних СУБД. Називаються ці типи по-різному, наприклад в Oracle цей тип називається LONG, в DB 2 - LONG VARCHAR, в Sybase і MS SQL Server - TEXT.

Однак слід зазначити, що специфіка реалізації окремих типів даних серйозним чином впливає на результати запитів до БД. Особливо це стосується реалізації типів даних DATE і TIMESTAMP. Тому при перенесенні пріло¬женій слід бути уважним, на різних платформах вони можуть працювати по-різному, і однією з причин може бути відмінність в інтерпретації типів даних.

Константи дати, часу і часового інтервалу в реляційних СУБД представляються у вигляді строкових констант. Формати цих констант відрізняються в різних СУБД. Крім того, формат представлення дати різний в різних країнах. У більшості СУБД реалізовані способи настройки форматів представлення дат спеціальні функції перетворення форматів дат, як зроблено, наприклад, в СУБД Oracle. Наведемо приклади констант в MS SQL Server:

March 15 1999 , Маг 15 1999 , 3/15/1999 , 3-15-99 , 1999 MAR 15 . У СУБД Oracle та ж константа запишеться як 15 – MAR – 99.

Крім користувальницьких констант в СУБД можуть існувати і спеціальні системні константи. Стандарт SQL1 визначає тільки одну системну константу USER, яка відповідає імені користувача, під яким ви под¬ключілись до БД [4].

У специфікації SQL :2003 визнані п'ять зумовлених загальних типів, усередині яких можуть бути підтипи (див. таб.7.1) [4].

Таблиця 7.1 – Типи даних SQL

Тип	**SQL реалізація**	
Строковий (символьний)	CHARACTER (чи CHAR)	
	CHARACTER VARYING (чи VARCHAR)	
	CHARACTER LARGE OBJECT (чи CLOB)	
Числовий	точні числові типи	integer; smallint; bigint; numeric; decimal.
	приблизні числові типи	real; double precision; float;
	логічний (булевий)	boolean
	дати-часу	date; time without time zone*; time with time zone; timestamp without time zone; timestamp with time zone

В операторах SQL можуть використовуватися вирази, які будуються за стандартними правилами застосування знаків арифметичних операцій додавання (+), віднімання (-), множення (*) і ділення (/). Однак в ряді СУБД операція ділення (/) інтерпретується як розподіл без остачі, тому при побудові складних виразів можна отримати результат, який не відповідає тра¬діціонной інтерпретації виразу. У стандарт SQL2 включена можливість виконання операцій додавання і віднімання над датами. У більшості СУБД також визначена операція конкатенації (зчеплення) над рядковими даними, позначається вона, на жаль, по-різному. Так, наприклад, для DB 2 операція конкатенації позначається подвійною вертикальною рисою, в MS SQL Server - знаком складання (+), тому два вирази, створені в різних СУБД, еквівалентні [4, 10]:

```
'Mr./Mrs. ' || NAME || ' ' LASTNAME
'Mr./Mrs. ' + NAME + ' ' LASTNAME
```

У стандарті SQL 2 вже введений ряд стандартних вбудованих функцій:

– BIT_LENGTH (рядок) - кількість бітів в рядку;

– CAST (значення AS тип даних) - значення, перетворене в заданий тип даних;

– CHAR _ LENGTH (строка) - довжина рядка символів;

– CONVERT (строка USING функція) - рядок, перетворена відповідно до зазначеної функції;

– CURRENT_DATE - поточна дата;

– CURRENT_TIME (точність) - місцевий час з вказаною точністю;

– CURRENT _ TlMESTAMP (точність) - поточні дата і час з вказаною точністю;

– LOWER (рядок) - рядок, перетворена до верхнього регістру;

– OCTED _ LENGTH (рядок) - число байтів в рядку символів;

– POSITION (перший рядок IN другий рядок) .- позиція, з якої починається входження першого рядка в другу;

– SUBSTRING (рядок FROM n FOR довжина) - частина рядка, що починається з n - го символу і має зазначену довжину;

– TRANSLATE (рядок USING функція) - рядок, перетворена з використанням зазначеної функції;

– TRIM (BOTH символ FROM рядок) - рядок, у якої вилучені всі перші і останні символи;

– TRIM (LEADING символ FROM рядок) - рядок, в якої вилучені всі перші зазначені символи;

– TRIM (TRAILING символ FROM рядок) - рядок, в якої вилучені останні зазначені символи;

– UPPER (рядок) - рядок, перетворена до верхнього регістру.

7.2 Рядковий тип даних

Рядкові дані (послідовності символів) мають три головні строкові типи. Для стовпця таблиці можна вказати тип character (n) або char(рядок фіксованої довжини), де n - максимальна кількість символів, що містяться в рядку. Якщо (n) не вказано, то передбачається, що рядок складається з одного символу. Якщо в стовпець типу character (n) вводиться m < n символів, то позиції, що залишилися, заповнюються пропусками.

Тип даних CHARACTER VARYING (n) або VARCHAR(рядок змінної довжини) застосовується тоді, коли дані, що вводяться, мають різну довжину і небажано доповнювати їх пропусками. При цьому зберігається тільки та кількість символів, яку ввів користувач. В даному випадку вказівка максимальної кількості символів обов'язкова (на відміну від character).

Дані типів CHARACTER і CHARACTER VARYING можуть брати участь в одних і тих же строкових операціях.

Тип даних CHARACTER LARGE OBJECT (CLOB – великий символьний об'єкт) використовується для представлення дуже великих символьних рядків (наприклад, статей, книг і тому подібне). У деяких БД цей тип називається

memo, а в інших - text. За даними цього типу можна виконувати не усі операції, передбачені для типів CHARACTER і CHARACTER VARYING. Так, їх не можна використовувати в операціях порівняння, за винятком рівності і нерівності. Крім того, стовпці цього типу не можуть бути первинними і зовнішніми ключами, а також бути оголошені як унікальні значення, що мають. Інакше кажучи, при створенні таблиць за допомогою оператора CREATE і оголошенні стовпців типу CLOB не можна використовувати ключові слова PRIMARY KEY, FOREIGN KEY і UNIQUE [4].

У наступному прикладі створюється таблиця із звичайним символьним стовпцем і стовпцем типу CLOB, значення якого можуть містити 100 000 символів:

```
CREATE TABLE myTable
( ( FIELD1 CHARACTER (60)
FIELD2 CLOB (100000));
```

7.3 Числовий тип даних

Числовий тип даних може бути двох видів точний і приблизний. Точні числові типи дозволяють точно виразити значення числа. Деякі величини мають дуже великий діапазон значень, і в таких випадках досить обмежитися деяким наближеним їх представленням з урахуванням технічних можливостей комп'ютера (розмірів регістра).

До ***точних числових*** відносяться наступні п'ять типів: integer, smallint, bigint, numeric та decimal.

Integer - ціле (без дробової частини) число. Кількість розрядів (точність) залежить від реалізації SQL. У деяких реалізаціях числа цього типу лежать в діапазоні від - 2 147 483 648 до 2 147 483 647 (четырехбайтное ціле число).

Smallint - мале ціле число. Кількість розрядів залежить від реалізації SQL, але не більше кількості розрядів integer в цій же реалізації. У деяких реалізаціях числа цього типу лежать в діапазоні від - 32 768 до 32 767 (двобайтове ціле число).

Bigint - велике ціле число. Кількість розрядів залежить від реалізації SQL і перевищує кількість розрядів числа типу integer.

Numeric (x, y) - число, в якому усього x розрядів (точність), з яких y розрядів (масштаб) відводиться для дробової частини. Якщо y не вказано (numeric (x)), то для дробової частини відводиться кількість розрядів, встановлена в системі за умовчанням. Якщо не вказані ні x, ні y (numeric), то приймаються обоє ці величини, встановлені за умовчанням. Наприклад, якщо вказаний тип numerc (6, 2), то максимальне значення числа дорівнює 9999.99.

Decimal (x, y) - десяткове число, в якому усього x розрядів, з яких y розрядів відводяться для дробової частини. Якщо x або/і y не вказані, то набувають значень за умовчанням. Цей тип дуже схожий на numeric.

Відмінність полягає в тому, що якщо в decimal (x, y) вказані x і y менше, ніж допустимі реалізацією SQL, то використовуватимуться останні. Якщо x і y не вказані, то застосовується система умовчань. Наприклад, ви задали для стовпця тип

decimal (6, 2). Якщо реалізація SQL дозволяє, то в цей стовпець можна ввести числа, що перевищують 9999.99. На відміну від decimal (x, y), тип numeric (x, y) жорстко задає діапазон можливих значень числової величини.

До ***приблизних числових*** типів відносяться наступні три типи: real, double precision та Float.

Real - дійсне число одинарної точності з плаваючою розділовою точкою (ця точка "плаває", з'являючись в різних місцях числа). Наприклад, 5.25, 5.257, 5.2573. Точність представлення числа залежить від реалізації SQL і устаткування. Наприклад, 32-бітовий комп'ютер дає велику точність, чим 16-бітовий;

Double precision - дійсне число подвійної точності з плаваючою розділовою точкою. Точність представлення числа залежить від реалізації SQL і устаткування. Застосовується для представлення наукових даних (наприклад, результатів вимірів) в широкому діапазоні значень, тобто як дуже малих (близьких до 0), так і дуже великих;

Float (x) - дійсне число з плаваючою розділовою точкою і мінімальною точністю x, що займає не більше 8 байтів. Якщо комп'ютер може підтримати вказану точність, використовуючи апаратну одинарну точність, то система використовуватиме арифметику одинарної точності. Якщо вказана точність вимагає арифметики з подвійною точністю, то система використовуватиме її.

Цей тип слід застосовувати, якщо передбачається можливість перенесення бази даних на іншу апаратну платформу, що відрізняється розмірами регістрів. Приклад значення типу float : 5.318E-24 (тобто 5.318, помножене на 10 в мірі - 24). Таку ж форму представлення мають і числа типу real і double precision.

При створенні таблиць цілочисельні типи застосовуються для стовпців, що містять різного роду ідентифікатори, наприклад, номера (коди) клієнтів, товарів, замовлень і тому подібне. Зрозуміло, чи її вміст стовпця має бути цілим числом (наприклад, кількість ящиків, пляшок, штук і тому подібне), то тип цього стовпця природно визначити як integer, smallint або bigint [10].

7.4 Логічні дані

У SQL тип даних boolean (булевий) має три значення - true, false і unknown. Значення unknown (невідоме) було введене для позначення результату, що виходить при срав¬нении зі значенням null (невизначене).

Якщо користувач ще не ввів в елемент таблиці ніякого значення, то цей "порожній" осередок містить значення null, що інтерпретується як невідоме або невизначене значення.

Результатом будь-якої операції порівняння true або false з null або з unknown завжди являється unknown.

У SQL -выражениях логічні значення полягають в кавичкики, наприклад, 'TRUE' або 'true'.

7.5 Дата і час

Тип data (дата) призначений для зберігання значень дати, елементи яких розташовані в наступному порядку: рік (4 цифри), дефіс (-), місяць (2 цифри), дефіс, день (2 цифри).

Таким чином, значення дати займають 10 позицій, наприклад, 2005-10-02.

Дані цього типу можуть містити будь-яку дату з 0001 року по 9999 рік.

Для ***представлення часу*** передбачено два типи: time without time zone та time with time zone.

Time without time zone (час без часового поясу) призначена для зберігання значень часу, елементи яких розташовані в наступному порядку :годинник, двокрапка, мину¬ти, двокрапка, секунди. Годинник і хвилини представляються двома цифрами, а секунди можуть бути представлені двома і бо¬леї цифрами (якщо вимагається дробова частина), наприклад 18:35:19.547. Довжина дробової частини секунд залежить від реалізації, але внутрішнє представлення часу повинне мати не менше 6 цифр. За змовчанням час цього типу представляється без дробової частини секунд.

Time with time zone (час з часовим поясом) - такий же тип даних, як і time without time zone. Відмінність заклю¬чается лише в тому, що до значення часу додається ще і інформація про різницю між місцевим і всесвітнім часом. Всесвітній час (Universal Time Coordinated, UTC) – це час по Грінвічу, тобто час нульового меридіана, що ходить через м.Грінвіч у Великобританії (Greenwich Mean Time, GMT). Значення різниці між локальним і всесвітнім часом знаходиться в діапазоні від -12:59 до 13:00.

Довжина даних даного типу дорівнює довжині даних типу time without time zone плюс 6, оскільки до-полнительная інформація про різницю часів займає 6 позицій (дефіс, знак (+) або (-), 2 цифри для годинника, двійкова, 2 цифри для хвилин).

Для ***одночасного представлення дати*** і часу служать наступні два типи: timestamp without time zone та timestamp with time zone.

Timestamp without time zone (дата і час без часового поясу). Елементи даних цього типу мають такі ж характеристики, як і для даних типу date і time without time zone, за винятком одного: дані типу timestamp without time zone по умовчанню мають 6 цифр в дробовій частині секунд, а не 0, як в типі time without time zone. Для вказівки кількості цифр в дробовій частині використовується синтаксис timestamp without time zone (л). Якщо дробової частини немає, то дані займають 19 позицій: 10 позицій для дати, один пропуск і 8 позицій для часу. Якщо визначена дробова частина, то довжина даних дорівнює 20 плюс количествоцифр в дробовій частині секунд;

Timestamp with time zone (дата і час з часовим поясом) - такий же тип даних, як і timestamp with time zone. Відмінність полягає в тому, що до значення часу додається ще і інформація про різницю між місцевим і всесвітнім часом. Додаткова інформація займає 6 позицій. Дані типу timestamp with time zone без

дробової частини займають 25 позицій, з дробовою частиною - 26 плюс кількість цифр в дробовій частині секунд.

Щоб представити в SQL -выражении дату, час або дату-час, необхідно використовувати функцію cast об приведення до заданого типу. Допустимо, в таблиці продажу є стовпець дата типу data.

Щоб отримати відомості з цієї таблиці за період після 2005-09-30, слід виконати такий запит:

SELECT * FROM Продажі WHERE Дата > CAST ('2005-09-30' AS DATE);

Тут рядок, що містить дату, приводиться до типу data, результат бере участь в операції порівняння з даними стовпця Дата.

У мові SQL є три функції, які ***повертають поточну дату*** і час :

- current_date - повертає поточну дату (тип date).

Наприклад, 2005-06-18;

- current_time (число) - повертає поточний час (тип time).

Цілочисельний параметр число вказує точність представлення секунд. Наприклад, при число = 2 секунди бу¬дутий представлені з точністю до сотих (дві цифри в дроб¬ний частині) : 12:39:45.27;

- current_timestamp (число) - повертає дату і час (тип TIMESTAMP), наприклад 2005-06-18 12:39:45.27. Цілочисельний параметр число вказує точність представлення секунд [4,10].

7.6 Структура мови SQL

Мова SQL має два основних компоненти (див.рис.7.1):

–мову DDL (Data Definition Language), призначену для визначення структур бази даних та управління доступом до даних (оператори CREATE TABLE; DROP TABLE; ALTER TABLE; CREATE INDEX; DROP INDEX);

–мову DML (Data Manipulation Language), призначену для вибірки і оновлення даних (оператори SELECT, INSERT, UPDATE, DELETE, COMMIT – фіксація змін, ROLLBACK – відміна внесених змін) (рис.7.2).

Після створення загальної структури бази даних можна приступити до створення таблиць, якими є стосунки, що входять до складу проекту бази даних.

Базовий синтаксис оператора створення таблиці має наступний вигляд:

CREATE TABLE ім’я_таблиці (ім’я_стовбця тип_даних [NULL | NOT NULL] [,...n])

Головне в команді створення таблиці – визначення імені таблиці і опис набору імен полів, які вказуються у відповідному порядку. Крім того, цією командою обмовляються типи даних і розміри таблиці [4, 12, 13].

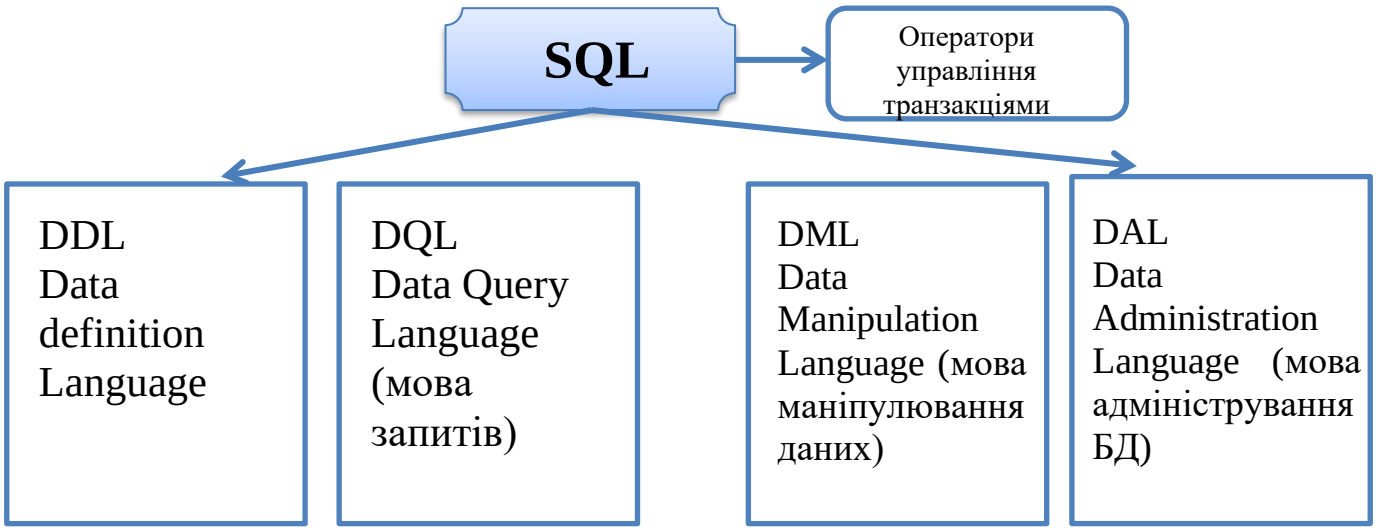

Рисунок 7.1 – Структура SQL

Ключове слово NULL використовується для вказівки того, що в даному стовпці можуть міститися значення NULL. Значення NULL відрізняється від пропуску або нуля – до нього удаються, коли необхідно вказати, що дані недоступні, опущені або недопустимі. Якщо вказано ключове слово NOT NULL, то будуть відхилені будь-які спроби помістити значення NULL в даний стовпець. Якщо вказаний параметр NULL, приміщення значень NULL в стовпець дозволене. За умовчанням стандарт SQL передбачає наявність ключового слова NULL.

Ми використовували спрощену версію оператора CREATE TABLE стандарту SQL. Його повна версія наводиться при обговоренні питань забезпечення цілісності даних.

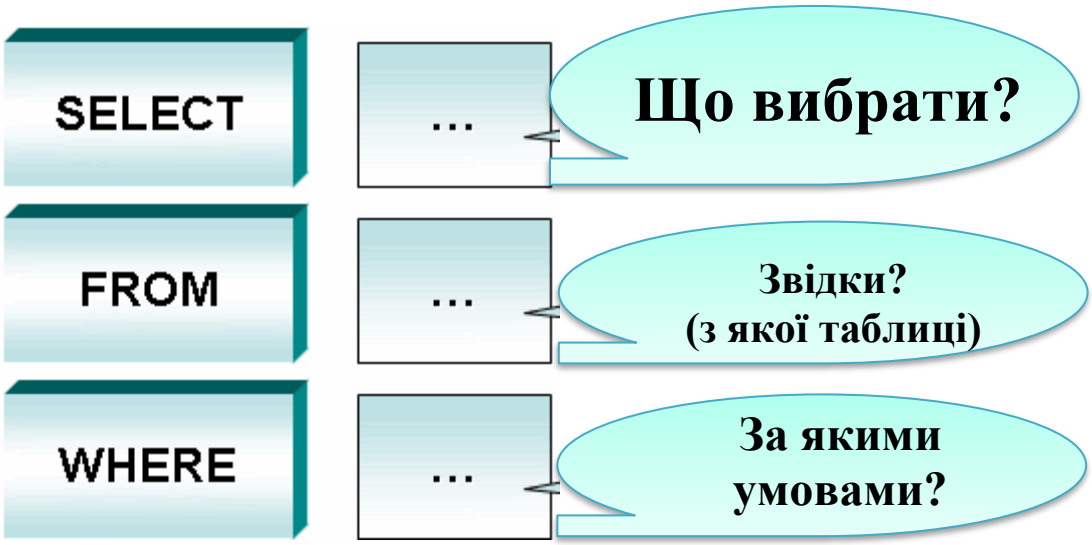

Рисунок 7.2 – Структура простого запиту в БД

Розглянемо приклад. Створити таблицю для зберігання даних про товари, що надходять у продаж в деякій торгівельній фірмі. Необхідно врахувати таківідомості, як назва і тип товару, його ціна, сорт і місто, де товар виробляється.

Реалізація приведена нижче.

```
CREATE TABLE Товар
(Назва VARCHAR(50) NOT NULL
Ціна MONEY NOT NULL
Тип VARCHAR(50) NOT NULL
Сорт VARCHAR(50)
МістоТовару VARCHAR(50))
```

7.7 Транзакція БД

Транзакція - це послідовність операцій над БД, розглянутих СУБД як єдине ціле. Або транзакція успішно виконується, і СУБД фіксує (COMMIT) зміни БД, вироблені цією транзакцією, у зовнішній пам'яті, або жодне з цих змін ніяк не відбивається на стані БД. Поняття транзакції необхідне для підтримки логічної цілісності БД. Підтримка механізму транзакцій є обов'язковою умовою навіть одного користувача СУБД (якщо, звичайно, така система заслуговує назви СУБД). Але поняття транзакції набагато важливіше в багатокористувацьких СУБД.

Те 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 управлінні паралельно виконуються транзакціями з боку СУБД кожен з користувачів може в принципі відчувати себе єдиним користувачем СУБД (насправді, це дещо ідеалізований погляд, оскільки в деяких випадках користувачі багатокористувацьких СУБД можуть відчути присутність своїх колег).

З управлінням транзакціями в багатокористувацької СУБД пов'язані важливі поняття сериализации транзакцій і серіального плану виконання суміші транзакцій. Під Серіалізация паралельно виконуються транзакцій розуміється такий порядок планування їх роботи, при якому сумарний ефект суміші транзакцій еквівалентний ефекту їх деякого послідовного виконання. Серіальний план виконання суміші транзакцій - це такий план, який призводить до сериализации транзакцій. Зрозуміло, що якщо вдається домогтися дійсно серіального виконання суміші транзакцій, то для кожного користувача, з ініціативи якого утворена транзакція, присутність інших транзакцій буде непомітно (якщо не брати до уваги деякого уповільнення роботи в порівнянні з однокористувацький режимом).

Існує кілька базових алгоритмів сериализации транзакцій. У централізованих СУБД найбільш поширені алгоритми, засновані на синхронізаційних захопленнях об'єктів БД. При використанні будь-якого алгоритму сериалізації можливі ситуації конфліктів між двома або більше транзакціями з доступу до об'єктів БД. В цьому випадку для підтримки сериалізації необхідно виконати відкат (ліквідувати всі зміни, вироблені в БД) однієї або більше транзакцій.

Отже, SQL можна в повній мірі віднести до традиційних мов програмування, він не містить традиційні оператори, що керують ходом виконання

програми, оператори опису типів і багато іншого, він містить тільки набір стандартних операторів доступу до даних, що зберігаються в базі даних. Оператори SQL вбудовуються в базова мова програмування, яким може бути будь-який стандартний мову типу C ++, Pascal, COBOL і т. Д. Крім того, оператори SQL можуть виконуватися безпосередньо в інтерактивному режимі.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Стандарт вищої освіти України: перший (бакалаврський) рівень, галузь знань 12 − Інформаційні технології, спеціальність 126 − Інформаційні системи та та технології. Затверджено Наказом Міністерства освіти і науки України 12.12.2018 р. № 1380. – 17 с.
2. Ben Forta. SQL in 10 Minutes a Day, Sams Teach Yourself: Sams Publishing; 5th edition. – 18 Aug. 2020. – 256p. ISBN-10 : 0135182794, ISBN-13 : 978-0135182796.
3. Гайдаржи В., Ізварін І.Бази даних в інформаційних системах: Навчальний посібник. – Тернопіль: Навчальна книга.– 2018.– 418 с.
4. Jamie Chan. SQL: Learn SQL (using MySQL) in One Day and Learn It Well. SQL for Beginners with Hands-on Project. – 2018. – 166p. ASIN : B07K374J19.
5. Мулеса О.Ю. Інформаційні системи та реляційні бази даних. Навч.посібник. – Електронне видання, 2018. – 118 с.
6. Корнієнко С.К. Проектування інформаційного забезпечення автоматизованих систем. Навч. Посібник. – Запоріжжя: ЗНТУ, 2015. – 210 с.
7. Карпуша В.Д., Панченко Б.Є. Моделювання та проектування реляційних баз даних : навч. посіб. – Суми : СДУ, 2010.
8. Сенів М. М., Яковина В. С. Безпека програм та даних : навч. посіб. Львів : Видавництво Львівської політехніки, 2015.
9. Завадський І.О. Основи баз даних: [Навч. посіб.] / І.О. Завадський. – К. : Видавець І.О. Завадський, 2011. – 192 с.
10. Балик Н., Мандзюк В.Бази даних MySQL: Навчальний посібник. – Тернопіль: Навчальна книга – Богдан, 2010.– 160 с.
11. Гайдаржи В.І., Дацюк О.А. Основи проектування та використання баз даних : Навч. посібник Київ : Політехніка, 2004. – 166с.
12. Адміністрування баз даними // Режим доступу: http://firebirdsql.org/manual/ru/migration- mssql-db-admin-ru.html.
13. Управління і адміністрування баз даними // Режим доступу: http://www.interface.ru/home.asp?artId=50&cId=3.

Навчальне видання

Каштан Віта Юріївна
Іванов Денис Валерійович

Конспект лекцій з дисципліни "Бази даних в інформаційних системах". Частина 1.

Для студентів галузі знань 12 "*Інформаційні технології*" *спеціальності* 126 "*Інформаційні системи та технології*"

Електронний ресурс

Видано
у Національному технічному університеті
«Дніпровська політехніка».
Свідоцтво про внесення до Державного реєстру ДК №1842 від 11.06.2004.
49005, *м. Дніпро, просп. Дмитра Яворницького,* 19.