

Міністерство освіти і науки України
Запорізький національний технічний університет

ОРГАНІЗАЦІЯ БАЗ ДАНИХ І ЗНАНЬ

методичні вказівки до практичних занять

для студентів напрямку підготовки
6.050103 "Програмна інженерія"
усіх форм навчання



2009

Організація баз даних і знань. Методичні вказівки до практичних занять для студентів напрямку підготовки 6.050103 “Програмна інженерія” усіх форм навчання /Уклад.: Корнієнко С.К., Качан О.І. –Запоріжжя: ЗНТУ, 2009. –30 с.

Укладачі: С.К. Корнієнко, доцент, к.т.н.,
О.І. Качан, асистент

Рецензент: В.П. Пінчук, доцент, к.ф.-м.н.

Відповідальний
за випуск: А.В. Притула, доцент, к.т.н.

Затверджено
на засіданні кафедри
"Програмні засоби"

Протокол № 1 від 28.08.08 р.

ЗМІСТ

Вступ	4
1 Життєвий цикл інформаційної системи	5
2 Концептуальне проектування	8
2.1 Аналіз концептуальних вимог та інформаційних потреб	8
2.2 Виявлення інформаційних об'єктів та зв'язків між ними	9
2.3 Побудова концептуальної моделі	10
3 Логічне проектування	13
3.1 Вибір СКБД	13
3.2 Відображення концептуальної схеми на логічну схему БД	15
4 Основи Мови структурованих запитів SQL	19
4.1 Типи даних	19
4.2 Класифікація команд SQL	22
4.3 Створення таблиць	23
4.4 Проектування простих запитів на вибірку	27
Література	28
Додаток А Словник основних термінів	29

ВСТУП

Проектування інформаційної системи (системи баз даних) уявляє собою трудомісткий, тривалий і, в багатьох випадках, неформалізований процес. Це комплексна проблема, яка торкається в кінцевому рахунку не тільки питань обробки даних, але й усієї організації в цілому.

Якість одержаної в підсумку структури бази даних (БД) визначається загальною методологією проектування, яка використовується на кожному кроці технічними прийомами, обґрунтованістю інформаційних вимог, а також готовністю людських та матеріальних ресурсів організації до проведення цієї роботи.

Методичні вказівки містять основні відомості та практичні рекомендації з концептуального і даталогічного проектування інформаційних систем, а також розрахунково-графічне завдання та рекомендації з їх виконання.

Призначені для використання на практичних і лабораторних заняттях з дисципліни «Організація баз даних та знань», а також в процесі самостійної роботи студентів.

1 ЖИТТЄВИЙ ЦИКЛ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Життєвий цикл інформаційної системи (ІС) являє собою концепцію, в рамках якої корисно і зручно розглядати розвиток системи за термін часу. Ця концепція створює добрі передпосилання для регламентування функцій адміністратора баз даних.

Життєвий цикл системи баз даних поділяється на дві окремі фази: фазу аналізу та проектування, та фазу експлуатації. На протязі першої фази відбувається збір вимог користувачів та проектування баз даних, на протязі другої – машинна реалізація та використання.

З точки зору проектувальника та користувача можна деталізувати зміст робіт, які виконуються на протязі цих фаз життєвого циклу системи. В цьому випадку вказані фази містять такі етапи:

- а) фаза аналізу та проектування:
 - 1) формулювання та аналіз вимог;
 - 2) концептуальне проектування;
 - 3) проектування реалізації;
 - 4) фізичне проектування;
- б) фаза експлуатації:
 - 1) реалізація бази даних;
 - 2) аналіз функціонування та підтримка;
 - 3) модифікація та адаптація.

В літературі [1-5] етапи а, 1 та а, 2 часто поєднуються в етап так званого «паперового» проектування, а етапи а.3 та а.4 в етап програмної реалізації.

Основні мети проектування баз даних складаються в тому, щоб забезпечити користувачів точними даними, необхідними для виконання ними своїх службових обов'язків, а також в тому, щоб забезпечити доступ до даних за допустимий час.

Для досягнення першої мети необхідно, щоб елементи бази даних більш повним образом відповідали вимогам користувачів організації, заснованим на їх загальних метах, внутрішній організаційній структурі та припустимих вимог доступу до даних.

Досягнення другої мети (продуктивності) потребує, щоб структура бази даних, отримана внаслідок процесу проектування, забезпечила достатньо швидкий доступ до даних з тим, щоб ті, хто користується цими даними, могли виконувати свої службові обов'язки достатньо ефективно.

Виходячи з цього, на стадії проектування БД проєктувальник повинен проробити наступну роботу:

- обстежити предметну область автоматизації;
- розробити концептуальну модель предметної області;
- опрацювати технологію обслуговування ІС, тобто визначити порядок збору, збереження даних в БД, частоту та формати вводу-виводу даних, правила роботи всіх груп користувачів. Проєкт повинен забезпечувати простоту та зручність майбутньої експлуатації ІС, захист даних від некоректних поновлень користувачами та від зруйнувань при збоях комп'ютеру;

- вибрати інструментальні засоби (СКБД) для реалізації;
- перевірити коректність проєкту. Проєкт повинен адекватно, на необхідному рівні детальності, відображати предметну область (ПО), тобто всім виділеним об'єктам та процесам ПО повинні відповідати дані та процедури обробки в комп'ютері.

На стадії програмної реалізації необхідно виконати наступні пункти:

- описати засобами СКБД та ввести в комп'ютер схеми всіх відносин;

- розробити інтерфейси користувачів з БД. Сюди входять розробка екранних форм для вводу та відображення даних, зручних екранних засобів звернення та доступу до даних в БД, порядку вводу та поновлення даних; визначення розмірів та змісту порцій даних, які одночасно відображаються на екрані, порядку їх розміщення. Кожна екранна форма повинна забезпечувати максимальну інформативність та зручність сприйняття, створювати звичне для користувача середовище;

- розробити програмне забезпечення ІС для всіх додатків;
- заповнити ІС налагодженими даними та відлагодити їх;
- провести тестування системи та скоректувати технологію її обслуговування;
- скласти необхідні інструкції за системою.

Стадія експлуатації починається з наповнення системи реальними даними, після чого відбувається безпосередньо використання ІС, підтримка її функціонування. Стадія експлуатації може включати вдосконалення системи та розробку нових додатків в разі розвитку та змінення відповідної предметної області.

Контрольні питання

1. З яких основних етапів складається життєвий цикл ІС?
2. Які питання вирішуються на стадії проектування?
3. Які питання вирішуються на стадії програмної реалізації?

2 КОНЦЕПТУАЛЬНЕ ПРОЕКТУВАННЯ

Аналіз предметної області є першим етапом проектування БД будь-якого типу і закінчується побудованням інформаційної структури (концептуальної моделі). На даному етапі аналізуються запитання користувачів, вибираються інформаційні об'єкти та їх характеристики, і на основі проведеного аналізу структурується предметна область. Аналіз ПО є етапом, який не залежить від програмних і технічних середовищ, в яких буде реалізовуватися БД.

Аналіз предметної області доцільно розбити на три фази:

- аналіз концептуальних вимог та інформаційних потреб;
- виявлення інформаційних об'єктів та зв'язків між ними;
- побудова концептуальної моделі ПО і проектування концептуальної схеми БД.

2.1 Аналіз концептуальних вимог та інформаційних потреб

На даному етапі встановлюються мети організації, визначаються специфічні вимоги до бази даних, які витікають з цих мет або сформульовані безпосередньо керуючим персоналом організації. При цьому необхідно вирішити наступні завдання:

- аналіз вимог користувачів до БД (концептуальних вимог);
- виявлення заданих завдань по обробці інформації (аналіз додатків);
- виявлення перспективних завдань(перспективних додатків);
- документування наслідків аналізу.

Ці вимоги документуються у формі, доступній як кінцевому користувачу, так і проектувальнику бази даних. При цьому використовуються методи інтерв'ювання персоналу різних рівнів керування та головних спеціалістів організації, зв'язаних з процесом обробки даних.

Вимоги користувачів до БД, яка розробляється, являють собою список запиту з показом їх інтенсивності та об'ємів даних. Тут же з'ясовуються вимоги до вводу, поновлення та коректування інформації. Вимоги користувачів уточнюються та доповнюються при аналізі існуючих і перспективних напрямків.

2.2 Виявлення інформаційних об'єктів та зв'язків між ними

Друга фаза аналізу предметної області складається з вибору інформаційних об'єктів, завдання необхідних властивостей для кожного об'єкту, виявлення зв'язків між об'єктами, визначення обмежень, що накладаються на інформаційні об'єкти, типів зв'язку між ними, характеристик інформаційних об'єктів.

При виборі інформаційних об'єктів бажано відповісти на ряд запитань:

- 1) На які класи можна розбити дані, що зберігаються в БД?
- 2) Яке ім'я можна надати кожному класу даних?
- 3) Які найбільш цікаві характеристики (з точки зору користувача) кожного класу даних можна виділити?
- 4) Які імена можна присвоїти обраним наборам характеристик?

Виділення інформаційних об'єктів – процес ітеративний. Він виконується за основою аналізу інформаційних потоків та інтерв'ювання споживачів. Характеристики інформаційних об'єктів визначаються такими ж методами.

Далі виділяються зв'язки між інформаційними об'єктами. Під час цього процесу слід відповісти на наступні запитання:

- 1) Які типи зв'язків між інформаційними об'єктами?
- 2) Яке ім'я можна присвоїти кожному типу зв'язків?
- 3) Які можливі типи зв'язків, що можуть бути використані?
- 4) Чи мають рацію які-небудь комбінації типів зв'язків?

При необхідності слід задати обмеження на об'єкти, їх характеристики та зв'язки, відповідаючи на наступні запитання:

- 1) Яка область значень для числових характеристик?
- 2) Які функціональні залежності між характеристиками одного інформаційного об'єкту?
- 3) Який тип відображення відповідає кожному типу зв'язків?

Під обмеженням цілісності звичайно розуміють логічні обмеження, які накладаються на дані. Обмеження цілісності – це така властивість, яку ми задаємо для деякого інформаційного об'єкту або його характеристики і яка повинна зберігатися для кожного їх стану.

Наприклад, твердження, що оклад професора ВНЗу більше окладу доцента, може розглядатися як обмеження цілісності тільки в тому випадку, якщо воно справедливо у будь-який момент та не

залежить від процедури підвищення заробітної платні або зміни, наприклад, тарифної сітки.

В загальному випадку розглядається три види обмежень: *внутрішні, явні та змішані*. Внутрішні обмеження в цілому зв'язані з моделлю даних, яка підтримується СКБД.

Між різними інформаційними об'єктами, а також між інформаційним об'єктом та його характеристиками виникають певні асоціації, які називаються зв'язками. При проектуванні БД прийнято розглядати взаємозв'язки між інформаційними об'єктами трьох типів: *«один до одного», «один до багатьох», «багато до багатьох»*.

2.3 Побудова концептуальної моделі

Заключна фаза аналізу ПО полягає в проектуванні її інформаційної структури (концептуальної схеми, або концептуальної моделі).

Концептуальна модель вживається для структурування ПО з урахуванням не тільки інформаційних інтересів користувачів системи, але й інформаційних витрат самої предметної області.

В рамках кожної БД концептуальні вимоги узагальнюються в концептуальну модель, яка виражена абстрактними засобами, що дозволяють побачити зміст предметної області.

При концептуальному проектуванні як основна модель використовується модель типу *«сутність - зв'язок»* (ER – модель), побудова якої вміщує декілька етапів:

- визначення сутностей;
- визначення атрибутів сутностей;
- ідентифікація джерельних атрибутів сутностей;
- визначення зв'язків між сутностями.

На мові **ER** - моделі інформаційний об'єкт називають сутністю. При цьому розрізняють тип сутності та екземпляр сутності. Під типом сутності розуміють набір однорідних об'єктів, які виступають як одне ціле. Поняття *«екземпляр сутності»* відноситься до конкретного об'єкту.

Наприклад:

тип сутності – службовець;

екземпляр сутності – Петренко, Іваненко та інші

Сутність прийнято визначати поїменованими характеристиками, які називаються атрибутами. Найменування повинно бути унікальним для кожного екземпляру сутності, але воно може повторюватись для різних типів сутностей.

В кожному наборі атрибутів, які характеризують сутність, необхідно обрати ключеві атрибути, тобто атрибути, які однозначно ідентифікують кожний екземпляр сутності.

Після визначення сутностей та атрибутів виявляються залежності між двома та більше сутностями та атрибутами, зв'язки атрибутів між собою.

Далі визначається, які зв'язки необхідні, а які зайві. Для виявлення зв'язків визначаються їх характеристики, кожний тип зв'язку «*сутність - сутність*», «*атрибут - атрибут*» іменуються.

Моделювання ПО закінчується графічним оформленням всіх виявлених сутностей, зв'язків між ними, атрибутів та утворення всіх специфікацій. При моделюванні предметної області проєктувальник розбиває її на ряд локальних областей, моделює кожне локальне уявлення, а потім їх з'єднує.

Вибір локального уявлення залежить від масштабів ПО. Для зручності проєктування в окремому локальному уявленні бажано використовувати 6-7 типів сутностей.

Найчастіше за все, локальне уявлення відповідає окремому зовнішньому застосуванню, наприклад, окремому функціональному завданню або окремому користувачу. Але воно може відповідати також цілій області даних, яка використовується декількома засобами.

При виборі області для локального уявлення проєктувальник відшукує компромісне рішення, оскільки вузька область веде до зниження рівня інтеграції даних та їх подрібненню, а велика область - до нечіткості та складності проєктування.

Після формування локальних уявлень виконується їх об'єднання в єдину глобальну інформаційну структуру. При цьому зазвичай використовуються такі основні концепції об'єднання, як агрегація та узагальнення.

Контрольні питання

1. В чому полягає аналіз предметної області?
2. Що розуміють під обмеженням цілісності?
3. Який інструментарій використовується на етапі побудови концептуальної моделі?
4. Чим відрізняється тип сутності від екземпляру сутності?
5. Яким чином відбувається об'єднання локальних уявлень?

3 ЛОГІЧНЕ ПРОЕКТУВАННЯ

Основним завданням логічного проектування є розробка логічної схеми, орієнтованої на обрану СКБД. Процес логічного проектування вміщує наступні етапи:

- 1) Вибір конкретної СКБД.
- 2) Відображення концептуальної схеми на логічну схему.
- 3) Вибір джерел.
- 4) Опис мови запитань.

3.1 Вибір СКБД

При виборі **СКБД**, яка реалізує конкретну **БД**, виділяють два підходи до її оцінки. Перший підхід пов'язаний з поглядом на кінцеву тему, а другий - технічний, пов'язаний з продуктивністю системи. З урахуванням цієї системи розглянемо сім груп параметрів для характеристики і наступного вибору **СКБД**. Дані параметри, не претендуючи на вичерпну повноту, дають достатньо повну картину про можливості **СКБД**.

Загальні характеристики:

- тип комп'ютеру;
- максимальна кількість записів у файлу;
- максимальний об'єм файлу;
- максимальна кількість копій у запису;
- максимальна кількість символів на запис;
- максимальна кількість індексів на файл;
- максимальна кількість таблиць в операції з'єднання;
- максимальна кількість файлів, доступних для однієї команди;
- максимальна кількість одночасно відкритих файлів;
- максимальна кількість файлів у **БД**;
- максимальна кількість записів у **БД**;
- максимальна кількість змінних;
- максимальна кількість символів в одному полі;
- імпорт - експорт даних;
- тип внутрішньої моделі даних;

- фірма – виробник;
- версія.

Управління файлами та пошук:

- * тип зв'язку;
- * модифікація декількох файлів;
- * двонаправлене з'єднання таблиць;
- * мова маніпулювання даними;
- * тип пошуку.

Засоби підтримки додатків:

- * каталог даних;
- * генератор додатків;
- * процедурна мова;
- * підпрограми;
- * макроси;
- * налагоджувач;
- * система підтримки виконання;
- * шифровка програм та даних;
- * розмежування доступу;
- * графіка;
- * графіка кольорова;
- * текстовий редактор;
- * статистика.

Введення та підтримка цілісності:

- * управління за допомогою команд;
- * управління за допомогою меню;
- * перевірка цілісності за таблицею;
- * перевірка унікальності ключа;
- * перевірка за датою;
- * незалежність даних.

Звіти:

- * звіти за декількома файлами;
- * зберігання форматів звіту;
- * видача звіту на екран;
- * видача звіту на магнітний носій;
- * поля, що обчислюються;
- * перевизначення формату дати;
- * генератор звітів;
- * підсумкові поля;
- * максимальна ширина звіту;
- * звіт в матричній формі.

Операційне середовище:

- * тип операційної системи;
- * об'єм необхідної оперативної пам'яті;
- * мова системи.

Додатки:

- * наявність варіанту мережі;
- * вартість.

3.2 Відображення концептуальної схеми на логічну схему БД

Для відображення інформаційної схеми, одержаної на етапі концептуального проектування, на логічну схему БД необхідно мати :

- розмір БД (кількість сутностей та атрибутів);
- частоту звернення до БД (кількість виконання додатків за одиницю часу);
- правила побудови логічних схем БД для наданих у розпорядження проектувальника СКБД;
- перелік додатків з вказівкою даних, які використовуються;
- конфігурацію, швидкодію та об'єм пам'яті, наданих у розпорядження проектувальника технічних засобів.

При відображенні інформаційної системи за допомогою

реляційної моделі даних кожний прямокутник схеми для ER – моделі предметної області відображається таблицею, яка є одним відношенням. При цьому слід враховувати обмеження на розмір таблиць, які накладає конкретна СКБД.

Інформаційні об'єкти (сутності) будуть відповідати найменуванням першого рівня меню, атрибути сутності – другому рівню меню.

Нехай на етапі концептуального проектування було складено дворівневе меню:

I рівень

1. Споживач
2. Виріб
3. Склад
4. Виробник
5. Заява

II рівень

1. Споживач

- 1.1 Адреса
- 1.2 Телефон
- 1.3 Профіль магазину

2. Виріб

- 2.1 Найменування
- 2.2 Вартість
- 2.3 Артикул
- 2.4 Шифр
- 2.5 Матеріал

3. Склад

- 3.1 № складу
- 3.2 Адреса
- 3.3 Телефон
- 3.4 Профіль

4. Виробник

- 4.1 Найменування
- 4.2 Адреса
- 4.3 Телефон
- 4.4 Профіль

5. Заява

- 5.1 № заяви
- 5.2 Дата
- 5.3 Найменування виробу

Проведемо відображення концептуальної схеми на реляційну схему та зробимо оцінку необхідного об'єму пам'яті для зберігання БД. В результаті отримаємо наступні відношення:

Відношення СПОЖИВАЧ

Найменування	Адреса	Телефон	Профіль магазину
20	30	7	23

На одну таблицю необхідно $20+30+7+23=80$ байт. Усього споживачів 200, тобто на зберігання всіх таблиць з заданим відношенням потрібно $80*200=16000$ байт.

Відношення ВИРІБ

Найменування	Вартість	Артикул	Шифр	Матеріал
30	5	5	5	15

На одну таблицю необхідно $30+5+5+5+15=60$ байт. Усього виробів 400, тобто на зберігання всіх таблиць з заданим відношенням потрібно $60*400=24000$ байт.

Відношення СКЛАД

№ складу	Адреса	Телефон	Профіль
5	18	7	30

Одна таблиця займає 60 байт. Усього складів 40, у підсумку $60*40=2400$ байт.

Відношення ВИРОБНИК

Найменування	Адреса	Телефон	Профіль
5	18	7	30

Одна таблиця займає 70 байт. Усього підприємств – виробників 15, у підсумку $70*15=1050$ байт.

Відношення ЗАЯВА

№ заяви	Дата	Найменування виробу
5	6	30

Одна таблиця займає 41 байт. Усього заяв 1000, у підсумку $41 \cdot 1000 = 41000$ байт.

Таким чином, у першому наближенні на зберігання **БД** необхідно $16000 + 24000 + 2400 + 1050 + 41000 = 84450$ байт.

Після визначення всіх відношень необхідно зробити нормалізацію бази даних, яка міститься в декомпозиції вихідних відношень **БД** на інші, більш дрібні та прості відношення.

Нормалізація усуває небажані функціональні залежності між атрибутами та забезпечує мінімальне дублювання даних за рахунок раціональної групування атрибутів.

Контрольні питання

1. На які етапи поділяється процес логічного проектування бази даних?
2. Якими розуміннями слід керуватися при виборі СКБД?
3. Які моделі даних існують?
4. Що називається об'єктним відношенням?
5. Що називається зв'язковим відношенням?
6. Яким чином виробляється відображення концептуальної моделі на реляційну модель даних?

4 ОСНОВИ МОВИ СТРУКТУРОВАНИХ ЗАПИТІВ SQL

4.1 Типи даних

SQL підтримує основні типи даних, які наведені в таблиці 4.1.

Таблиця 4.1 –Типи даних

Тип даних 1	Розмір 2	Опис 3
BINARY	1 байт на знак	У полі цього типу можуть зберігатися дані будь-якого типу. Дані відображаються в тому ж вигляді, у якому вони вводяться в це поле.
BIT	1 байт	Значення «Так» (Yes) і «Ні» (No), а також поля, що містять одне з двох можливих значень.
TINYINT	1 байт	Ціле значення від 0 до 255.
MONEY	8 байтів	Грошовий
DATETIME	8 байтів	Дата або час; припустимий будь-який рік від 100 до 9999.
UNIQUEIDENTIFIER	128 бітів	Унікальний ідентифікатор, який використовується при викликах вилучених процедур.
REAL	4 байти	Число з крапкою, що плаває, та одинарною точністю
FLOAT	8 байтів	Число з крапкою, що плаває, та подвійною точністю
SMALLINT	2 байти	Коротке ціле від –32 768 до 32 767 (див. «Примітки»).
INTEGER	4 байти	Довге ціле від –2 147 483 648 до 2 147 483 647.
DECIMAL	17 байтів	Тип даних для збереження точних числових значень від 10^{28-1} до 10^{28-1} .

Продовження таблиці 4.1

1	2	3
TEXT	2 байти на знак	Від 0 до 2,14 Гбайт (поле MEMO)
IMAGE	Не обмежено	Від 0 до 2,14 Гбайт. Використовується для об'єктів OLE.
CHARACTER	2 байти на знак	Від 0 до 255 знаків.

Окремі версії мови SQL мають певні відмінності. У таблиці 4.2 наведені типи даних ANSI SQL, еквівалентні ним типи даних мови SQL ядра бази даних **Microsoft Jet** і припустимі синоніми. Крім того, представлені еквівалентні типи даних **Microsoft® SQL Server™**.

Таблиця 4.2 – Типи даних окремих версій SQL

Типи даних ANSI SQL	Типи даних SQL Microsoft Jet	Синонім	Типи даних Microsoft SQL Server
BIT, BIT VARYING	BINARY	VARBINARY, BINARY VARYING BIT VARYING	BINARY, VARBINARY
Не підтримується	BIT	BOOLEAN, LOGICAL, LOGICAL1, YESNO	BIT
Не підтримується	TINYINT	INTEGER1, BYTE	TINYINT
Не підтримується	COUNTER	AUTOINCREMENT	Не підтримується
Не підтримується	MONEY	CURRENCY	MONEY
DATE, TIME, TIMESTAMP	DATETIME	DATE, TIME	DATETIME
Не підтримується	UNIQUEIDENTIFIER	GUID	UNIQUEIDENTIFIER
DECIMAL	DECIMAL	NUMERIC, DEC	DECIMAL
REAL	REAL	SINGLE, FLOAT4, IEEE SINGLE	REAL
DOUBLE PRECISION, FLOAT	FLOAT	DOUBLE, FLOAT8, IEEE DOUBLE, NUMBER	FLOAT
SMALLINT	SMALLINT	SHORT, INTEGER2	SMALLINT
INTEGER	INTEGER	LONG, INT, INTEGER4	INTEGER
INTERVAL	Не підтримується		Не підтримується
Не підтримується	IMAGE	LONGBINARY, GENERAL, OLEOBJECT	IMAGE
Не підтримується	TEXT	LONGTEXT, LONGCHAR, MEMO, NOTE, NTEXT	TEXT
CHARACTER, CHARACTER VARYING, NATIONAL CHARACTER, NATIONAL CHARACTER VARYING	CHAR	TEXT(n), ALPHANUMERIC, CHARACTER, STRING, VARCHAR, CHARACTER VARYING, NCHAR, NATIONAL CHARACTER, NATIONAL CHAR, NATIONAL CHARACTER VARYING, NATIONAL CHAR VARYING	CHAR, VARCHAR, NCHAR, NVARCHAR

4.2 Класифікація команд SQL

Команди SQL використовуються для виконання різноманітних дій над реляційними БД. Для зручності роботи вони розділяються на наступні групи (табл.. 4.3):

- команди визначення даних (*Data Definition Commands*);
- команди маніпуляції даними (*Data Manipulation Commands*);
- команди вибірки даних (*Data Query Commands*);
- команди керування транзакціями (*Transaction Control Commands*);
- команди керування даними (*Data Control Commands*)

Таблиця 4.3 – Команди мови SQL

Команда	Призначення
1	2
Команди визначення даних	
<i>alter table</i>	Змінює структуру таблиці
<i>create index</i>	Створює індекс
<i>create table</i>	Створює таблицю
<i>create view</i>	Створює подання
<i>Drop</i>	Вилучає таблицю, індекс, подання
Команди маніпуляції даними	
<i>Delete</i>	Видаляє запису таблиці
<i>Insert</i>	Додавляє запису в таблицю
<i>Update</i>	Змінює дані таблиці
Команда вибірки даних	
<i>Select</i>	
Команди керування транзакціями	
<i>commit</i>	Робить зміни, проведені з початку транзакції, постійними
<i>rollback</i>	Відкочує всі проведені зміни до крапки зберігання або до початку транзакції
<i>savepoint</i>	Встановлює контрольну крапку, до котрого згодом можна буде виконати відкат

Продовження таблиці 4.3

1	2
Команди керування даними	
<i>check database</i>	Перевіряє цілісність бази даних
<i>grant</i>	Надає привілеї
<i>revoke</i>	Скасовує надані раніше привілеї

4.3 Створення таблиць**4.3.1 Основний формат**

Щоб створити таблицю треба зробити, щонайменше, наступне.

- задати ім'я таблиці;
- задати імена складових її стовпчиків;
- визначити тип даних для кожного стовпчика;
- визначити (або використовувати по умовчання) нульовий статус для кожного стовпчика - припускається або забороняється використання в стовпчику нульових значень.

```
CREATE TABLE ім'я_таблиці  
( ім'я_стовпчика тип_даних [NULL | NOT NULL]  
[ , ім'я_стовпчика тип_даних [NULL | NOT NULL] ] ...);
```

Статус стовпчика **NOT NULL** означає обов'язкове заповнення відповідного стовпчика. Статус **NULL** - означає, що значення стовпчика можуть бути не визначені Наприклад,

```
create table customs  
(name varchar(25) not null,  
adress varchar(35) not null,  
phone char(8) null);
```

Типи даних **char** і **varchar** відрізняються тим, що для даних типу **char** виділяється дисковий простір фіксованої довжини незалежно від реальної потреби, а для даних типу **varchar** дисковий

простір виділяється по мірі необхідності в рамках зазначеного розміру, що дозволяє заощаджувати дисковий простір.

4.3.2 Обмеження на множину припустимих значень

Більшість комерційних систем підтримують пропозиції (обмеження) **PRIMARY KEY**, **UNIQUE**, **DEFAULT**, **CHECK**, **REFERENCES** і **FOREIGN KEY** у команді **CREATE TABLE** у відповідності зі стандартом **SQL/92**. Ці елементи забезпечують захист цілісності даних.

PRIMARY KEY помічає стовпчик (у якому не можуть бути присутніми нульові значення) у якості первинного ключа таблиці. Кожне значення, що вводиться в цей стовпчик, повинно бути унікальним. Якщо первинний ключ включає декілька стовпчиків, **PRIMARY KEY** визначається на рівні таблиці.

```
create table salespeople
(snum int not null primery key,
sname char(10) not null,
city char(10));
```

У наступному прикладі обмеження на первинний ключ є обмеженням на всю таблицю, оскільки використовує більш одного стовпчика.

```
create table namefields
(firstname char(10) not null,
lastname char(10) not null,
city char(10),
primery key (firstname, lastname));
```

UNIQUE – альтернативний, або можливий (*candidate keys*) ключ. Також гарантує унікальність елементів стовпчика. Це обмеження застосовується до тих полів, що були заявлені **NOT NULL**.

Унікальною можна також зробити й групу полів, вказавши **UNIQUE** в якості обмежень таблиці. При цьому комбінації значень

зазначених стовпчиків повинні бути унікальними, хоча в самих стовпчиках значення можуть бути й неунікальними.

DEFAULT визначає значення, що автоматично вставляється системою, якщо користувач не введе необхідні дані. Наприклад, слово "*невизначений*", якщо значення ще не визначене.

CHECK визначає, які дані можуть вводитися у визначений стовпчик, тобто дозволяє визначити область значень стовпчика. При цьому можна задати список, діапазон або шаблон припустимих значень.

Обмеження на таблицю рекомендується розташовувати після списку всіх її елементів (хоча це й необов'язково). Стовпчики можуть мати більш одного обмеження, котрі не розділяються комами:

```
CREATE TABLE ім'я_таблиці
  (ім'я_стовпчика тип [NULL|NOT NULL]
   [DEFAULT значення_по_умовчання]
   [обмеження_на_стовпчик]
   ...
   [обмеження_на_таблицю]...);
```

Наприклад,

create table товар

```
(код char(6) not null primary key
 check (код like '[A-Я] [A-Я] [0-9] [0-9][0-9] [0-9]'),
 назва varchar(80) not null,
 тип char(12)
 default "невизначений" null
 check (тип in ("економіка", "психологія",
               "програмування", "невизначений")),
 ціна money null,
 кількість int null)
```

Ключове слово ***LIKE*** означає завдання шаблону значень, а ***IN*** – список можливих значень.

4.3.3 Підтримка посилальної цілісності

Таблиці бази даних можуть бути певним чином зв'язані між собою. Такі зв'язки між таблицями встановлюються шляхом завдання посилань (*references*) між окремими полями таблиць.

Коли поле однієї таблиці посилається поле в іншій таблиці, воно називається *зовнішнім ключем (foreign key)*; поле, на яке воно посилається, називається його *батьківським ключем*. Імена зовнішнього та батьківського ключів можуть бути неоднаковими.

Кожне значення зовнішнього ключа повинно бути подане в батьківському ключі один і тільки один раз. У той же час на одне значення батьківського ключа можуть посилатися декілька полів зовнішнього ключа.

Наприклад,

create table покупці

(ном integer not null primary key,

прізвище char(15),

ном_прод integer,

foreign key (ном_прод) references продавці (ном_прод));

Зовнішній ключ, як і первинний, може бути визначений на будь-якій кількості полів, які разом розглядаються як єдине ціле (складовий ключ). Зовнішній і батьківський ключі повинні бути визначені на однаковій множині полів (за кількістю, типам і порядку слідування):

FOREIGN KEY (список стовпчиків) REFERENCES ім'я таблиці (список стовпчиків)

Таким чином, основна ідея *посилальної цілісності* полягає в тому, що всі значення зовнішніх ключів відсилають до визначеного рядка батьківського ключа. Як тільки значення розміщується в зовнішньому ключі, батьківський ключ перевіряє для того, щоб переконатися в тому, що таке значення в ньому є присутнім; у протилежному випадку команда відхиляється. Батьківський ключ повинний мати обмеження ***PRIMARY KEY*** або ***UNIQUE*** для гарантії того, що значення не подане більш ніж один раз.

4.4 Проектування простих запитів на вибірку

4.4.1 Загальний формат команди *SELECT*

SELECT [***ALL***|***DISTINCT***] список полів, що вибираються | *
FROM список таблиць
 [***WHERE*** умова вибірки]
 [***GROUP BY*** умова угруповання
 [***HAVING*** умова вибірки групи]]
 [***ORDER BY*** умова впорядкування];

DISTINCT – аргумент, який усуває дублювання значень із результату виконання речення ***SELECT***. Наприклад:

Символ "*" використовується, якщо необхідно вивести всі стовпчики.

```
select distinct прізвище
from продавці;
```

Якщо оператор ***SELECT*** витягає декілька полів, то ***DISTINCT*** виключає рядки, в яких всі вибрані поля ідентичні, тобто він діє на всю строку, а не на окремі поля.

Альтернативою ***DISTINCT*** є ключове слово ***ALL***, що має протилежне значення, тобто дозволяє виведення повторюваних рядків у запиті. По умовчанню використовується ***ALL***.

При роботі з декількома таблицями, що містять однойменні поля, перед іменами стовпчиків треба задавати імена відповідних таблиць. У більшості діалектів ***SQL*** для спрощення набору таблицям дозволяється задавати *псевдоніми* (*aliases*). Псевдонім указується після імені таблиці в списку таблиць:

```
select T.код, T.назва, T.ціна
from Товари [as] T;
```

Буква ***T*** перед іменами стовпчиків заміняє повне ім'я таблиці (*товари*). Цей запит еквівалентний наступному:

```
select Товари.код, Товари.назва, Товари.ціна
from Товари;
```

ЛІТЕРАТУРА

1. Боуман Дж., Эмерсон С., Дарновски М. Практическое руководство по SQL. - К: Диалектика, 1997. - 320 с.
2. Бруншир Дж. Введение в компьютерные науки. Общий обзор: Пер. с англ. - М.:Вильямс, 2001. -688 с.
3. Бхамидипати К. SQL. Справочник программиста. – М.:ЭКОМ,2003. – 304 с.
4. Дейт К. Введение в системы баз данных: Пер. с англ. –К.: Диалектика, 2002. –803 с.
5. Карпова Т.С. Базы данных: модели, разработка. реализация. – СПб.: Питер, 2001. – 304 с.
6. Конноли Т., Бегг К., Страчан А. Базы данных: проектирование, реализация и сопровождение. Теория и практика: - М.: Издательский дом "Вильямс", 2000. – 1120 с.
7. Корнієнко С. К. Системи баз даних: організація та проектування: Навч. Посібник. –Запоріжжя: ЗНТУ, 2006. –252 с.
8. Пасічник В.В., Резніченко В.А. Організація баз даних і знань. – К.:ВНУ, 2006. – 384 с.



ДОДАТОК А СЛОВНИК ОСНОВНИХ ТЕРМІНІВ

Адміністратор бази даних – особа, яка відповідає за виконання функцій *адміністрування* бази даних, тобто координацію дій по збору відомостей, проектуванню та експлуатації бази даних, а також по забезпеченню захисту даних.

Альтернативний ключ – атрибут (або група атрибутів), які не співпадають із первинним ключем і унікально ідентифікують кожний рядок у таблиці.

Атрибут – інформаційне відображення властивостей об'єкту.

База даних – поійменована структурована сукупність взаємозв'язаних даних, які відносяться до конкретної предметної області та призначені для задовольняння інформаційних потреб багатьох користувачів.

Безпека БД - захист від несанкціонованого доступу, зміни або руйнування даних.

Домен – набір значень елементів даних одного типу, який відповідає поставленим умовам.

Екземпляр сутності – опис конкретного об'єкту в наборі.

Зв'язок – функціональна залежність між сутностями.

Зв'язне відношення – відношення, яке зберігає ключі двох або більше об'єктних відношень, за якими встановлюється зв'язок між цими відношеннями.

Ключовий елемент даних – елемент, по якому можна визначити значення інших елементів даних.

Курсор (віртуальна таблиця) – об'єкт, який не містить власних даних. Це свого роду віртуальна таблиця, що не існує як незалежний об'єкт в базі даних і вміст якої береться з інших таблиць шляхом виконання запиту.

Нормалізація відношень – процес побудови оптимальної структури таблиць і зв'язків у реляційній базі даних.

Об'єкт – елемент інформаційної системи, інформацію про який зберігається.

Об'єктне відношення – відношення, яке зберігає дані про об'єкти (екземпляри сутності) предметної області.

Первинний ключ – атрибут (або група атрибутів), які однозначно ідентифікують кожний рядок у таблиці.

Посилальна цілісність – забезпечення співвідношення значень зовнішнього ключа екземпляра дочірній сутності значенням первинного ключа в батьківській сутності.

Предметна область – частина реальної системи, яка представляє цікавість для даного дослідження.

Словник даних – централізоване сховище відомостей про об'єкти, елементи даних, які їх складають, взаємозв'язки між об'єктами, їх джерела, значення, використання та формати представлення.

Ступінь відношення – кількість атрибутів (стовпців) відношення.

Сутність – це збиральне поняття, деяка абстракція реально існуючого об'єкта навколишнього світу, процесу або явища.

Таблиця (відношення) – деяка регулярна структура, яка складається з кінцевого набору однотипних записів.

Тип сутності – набір однорідних об'єктів, які виступають як єдине ціле.

Транзакція – це логічна одиниця роботи, яка виконується без порушення логічної цілісності бази даних. Якщо в процесі виконання транзакції виникла помилка виконання, то система, яка підтримує процес транзакції, гарантує повернення до первинного стану.

Тригер – попередньо визначена дія або послідовність дій, які автоматично здійснюються під час виконання операцій оновлення, додавання або видалення даних.

ODBC (Open Database Connectivity) – відкритий доступ до баз даних – загальне визначення мови та набір протоколів, які дозволяють клієнтському додатку працювати з командами та функціями, що підтримуються сервером.

OLE (Object Linking and Embedding) – зв'язування та впровадження об'єктів – технологія, яка дозволяє використовувати в додатку об'єкти, розроблені в іншому додатку. OLE-об'єктами можуть бути зву, рисунки, діаграми, відеокліпи тощо.

OLE-сервер – програма, яка може надавати іншим програмам можливість використання своїх об'єктів.

SQL (Structured Query Language) – мова структурованих запитів, офіційний стандарт мови для роботи з реляційними базами даних.