**Лекція 9. Елементи мови SQL**

Розглянемо елементи мови SQL (Structured Query Language). Однією з останніх версій

була версія стандарту мови SQL, яку прийнято в 1992 г. (Міжнародний стандарт мови баз даних SQL - International Standart Database Language SQL (1992)). Її неофіційна назва - SQL/92, або SQL-92, або SQL2). Документ, що описує стандарт, містить більше 600 сторінок тексту. Розберемо лише деякі поняття мови SQL.

Мова SQL стала фактично стандартною мовою доступу до баз даних. Усі СКБД, які можна називати "реляційні", реалізують той або інший діалект SQL. Багато з нереляційних систем також мають засоби доступу до реляційних даних. Метою стандартизації є можливість переносити додатки між різними СКБД.

Треба відмітити, що зараз жодна система не реалізує стандарт SQL у повному обсязі. Крім того, в усіх діалектах мови є можливості, які не вважаються стандартними. Таким чином, можно говорити, що кожний діалект - це надмножина деякою підмножини стандарту SQL. Однак, це ускладнює процес переносу додатків з одних СКБД на інші.

Мова SQL оперує термінами, що дещо відрізняються від термінів реляційної теорії, наприклад, замість "відношень" використовуються "таблиці", замість "кортежів" - "рядки", замість "атрибутів" - "колонки" або "стовбці".

Стандарт мови SQL, хоча і заснований на реляційній теорії, але в багатьох місцях відходить від неї. Для прикладу, відношення в реляційній моделі даних не допускає наявності однакових кортежів, а таблиці у термінології SQL можуть мати однакові рядки.

Мова SQL є реляційно повною. Це означає, що будь-який оператор реляційної алгебри може бути виражений відповідним оператором мови SQL.

Оператори SQL

Основу мови SQL складають оператори, умовно розділені на декілька груп за функціями, що виконуються.

Виділяють такі групи операторів (наведемо основні оператори SQL):

Оператори DDL (Data Definition Language) - оператори визначення об’єктів бази даних

* + CREATE SCHEMA - створити схему бази даних
  + DROP SHEMA - видалити схему бази даних
  + CREATE TABLE - створити таблицю
  + ALTER TABLE - змінити таблицю
  + DROP TABLE - видалити таблицю
  + CREATE DOMAIN - створити домен
  + ALTER DOMAIN - змінити домен
  + DROP DOMAIN - видалити домен
  + CREATE COLLATION - створити послідовність
  + DROP COLLATION - видалити послідовність
  + CREATE VIEW - створити представлення
  + DROP VIEW - видалити представлення

Оператори DML (Data Manipulation Language) - оператори маніпулювання даними

* + SELECT - відібрати рядки з таблиць
  + INSERT - добавити рядки в таблицю
  + UPDATE - змінити рядки в таблиці
  + DELETE - видалити рядки в таблиці
  + COMMIT - зафіксувати внесені зміни
  + ROLLBACK – відмовитись від внесених змін

Оператори захисту і керування даними

* + CREATE ASSERTION - створити обмеження
  + DROP ASSERTION - видалити обмеження
  + GRANT - надати привілеї користувачу або додатку для

маніпулювання об’єктами

* + REVOKE - відмінити привілеї користувачу або додатку

Виділяють також групи операторів встановлення параметрів сеансу, отримання інформації про базу даних, оператори статичного SQL, оператори динамічного SQL.

Найбільш важливими для користувачів є оператори маніпулювання даними (DML).

Приклади застосування операторів маніпулювання даними

**INSERT - вставка рядків у таблицю**

**Приклад1**. Вставка одного рядка у таблицю:

INSERT INTO

P (PNUM, PNAME) VALUES (4, "Іван");

**Приклад 2**. Вставка у таблицю декількох рядків, обраних з іншої таблиці (у таблицю TMP\_TABLE вставляються дані про постачальників з таблиці P, які мають номери, що більше 2):

INSERT INTO

TMP\_TABLE (PNUM, PNAME)

SELECT PNUM, PNAME FROM P

WHERE P.PNUM>2;

**UPDATE - оновлення рядків у таблиці**

**Приклад 3**. Оновлення декількох рядків у таблиці:

UPDATE P

SET PNAME = "Пушкаренко"

WHERE P.PNUM = 1;

DELETE - видалення рядків у таблиці

**Приклад 4**. Видалення декількох рядків у таблиці:

DELETE FROM P WHERE P.PNUM = 1;

**Приклад 5**. Видалення усіх рядків у таблиці:

DELETE FROM P;

Приклади застосування оператора SELECT

Одним з найбільш важливим і вживаним є оператор SELECT. Одночасно він є найбільш складним оператором SQL. Він призначений для вибірки даних з таблиць, тобто, власне, і реалізує головне призначення бази даних - надавати інформацію користувачеві.

Оператор SELECT завжди виконується над деякими таблицями, що входять до бази даних.

Зауваження. Насправді в базах даних можуть бути не лише постійно збережені таблиці, а й тимчасові таблиці і так звані представлення. Представлення - це просто збережені у базі дані SELECT-вирази. З точки зору користувача представлення - це таблиця, яка не зберігається постійно в базі даних, а "виникає" у момент звернення до неї. З точки зору оператора SELECT і постійно збережені таблиці, і тимчасові таблиці й представлення виглядають абсолютно однаково.

Результатом виконання оператора SELECT завжди є таблиця. Таким чином, за результатами дій оператор SELECT схожий на оператори реляційної алгебри. Будь-який оператор реляційної алгебри може бути виражений відповідним образом, що формулюється оператором SELECT. Складність оператора SELECT визначається тим, що він містить в собі усі можливості реляційної алгебри, а також додаткові можливості, яких у реляційній алгебрі немає.

Відбір даних з однієї таблиці

**Приклад 6**. Вибрати усі дані з таблиці постачальників (ключові слова ***SELECT***… ***FROM***…):

SELECT \* FROM P;

Зауваження. В результаті отримуємо нову таблицю, що містить повну копію даних з вихідної таблиці P.

**Приклад 7**. Вибрати усі рядки з таблиці постачальників, що задовольняють деяку умову (ключове слово ***WHERE***…):

SELECT \* FROM P

WHERE P.PNUM > 2;

Зауваження. В якості умови в розділі WHERE можна використовувати складні логічні вирази, що визначають поля таблиць, константи, операції порівняння (>, <, = і т.д.), скобки, логічні AND та OR, заперечення NOT.

**Приклад 8**. Вибрати деякі колонки (стовбці) з вихідної таблиці (вказується список колонок):

SELECT P.NAME FROM P;

Зауваження. В результаті отримуємо таблицю з однією колонкою, що містить усі найменування постачальників.

Зауваження. Якщо у вихідній таблиці присутне декілька постачальників з різними номерами, але з однаковими найменуваннями, то у результуючій таблиці *будуть рядки з повтореннями* - дублікати рядків автоматично не відкидаються.

**Приклад 9**. Вибрати деякі колонки з вихідної таблиці, видалив з результату рядки, що повторюються (ключове слово ***DISTINCT***):

SELECT DISTINCT P.NAME FROM P;

Зауваження. Застосування ключового слова DISTINCT приводить до того, що в результуючій таблиці будуть видалені усі рядки, що повторюються.

**Приклад 10**. Використання скалярних виразів й перейменування колонок в запитах

(ключове слово ***AS***…):

SELECT

TOVAR.TNAME,

TOVAR.KOL,

TOVAR.PRICE,

"=" AS EQU,

TOVAR.KOL\*TOVAR.PRICE AS SUMMA FROM TOVAR;

В результаті отримуємо таблицю з колонками, яких не було у вихідній таблиці

TOVAR:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TNAME** | **KOL** | **PRICE** | **EQU** | **SUMMA** |
| Болт | 10 | 100 | = | 1000 |
| Гайка | 20 | 200 | = | 4000 |
| Гвинт | 30 | 300 | = | 9000 |

**Приклад 11**.Впорядкування результатів запиту (ключове слово ***ORDER BY***…):

SELECT

PD.PNUM,

PD.DNUM, PD.VOLUME

FROM PD

ORDER BY DNUM;

В результаті отримуємо таку таблицю, яка впорядкована за атрибутом DNUM:

|  |  |  |
| --- | --- | --- |
| **PNUM** | **DNUM** | **VOLUME** |
| 1 | 1 | 100 |
| 2 | 1 | 150 |
| 3 | 1 | 1000 |
| 1 | 2 | 200 |
| 2 | 2 | 250 |
| 1 | 3 | 300 |

**Приклад 12**. Впорядкування результатів запиту за деякими полями з зростанням або спаданням (ключові слова ***ASC***, ***DESC***):

SELECT

PD.PNUM,

PD.DNUM,

PD.VOLUME

FROM PD ORDER BY

DNUM ASC, VOLUME DESC;

В результаті отримуємо таблицю, в якій рядки йдуть у порядку зростання значень поля DNUM, а рядки з однаковими значеннями DNUM йдуть у порядку спадання значень поля VOLUME:

|  |  |  |
| --- | --- | --- |
| **PNUM** | **DNUM** | **VOLUME** |
| 3 | 1 | 1000 |
| 2 | 1 | 150 |
| 1 | 1 | 100 |
| 2 | 2 | 250 |
| 1 | 2 | 200 |
| 1 | 3 | 300 |

Зауваження. Якщо явно не вказані ключові слова ASC або DESC, то за замовчанням приймається впорядкування за зростанням (ASC).

Відбір даних з декількох таблиць

**Приклад 13**. Природне з’єднання таблиць (спосіб 1 - явне задання умови з’єднання):

SELECT

P.PNUM, P.PNAME, PD.DNUM, PD.VOLUME

FROM P, PD

WHERE P.PNUM = PD.PNUM;

В результаті отримуємо нову таблицю, у який рядки з даними про постачальників з’єднано з рядками з даними про поставки деталей:

|  |  |  |  |
| --- | --- | --- | --- |
| **PNUM** | **PNAME** | **DNUM** | **VOLUME** |
| 1 | Іван | 1 | 100 |
| 1 | Іван | 2 | 200 |
| 1 | Іван | 3 | 300 |
| 2 | Петро | 1 | 150 |
| 2 | Петро | 2 | 250 |
| 3 | Микола | 1 | 1000 |

Зауваження. Таблиці, що з’єднуються, перераховуються у розділі FROM оператора, умова з’єднання наведена у розділі WHERE. Розділ WHERE, крім умови з’єднання таблиць, може також містити й умови відбору рядків.

**Приклад 14**. Природне з’єднання таблиць (спосіб 2 - ключові слова ***JOIN… USING…***):

SELECT

P.PNUM,

P.PNAME,

PD.DNUM,

PD.VOLUME

FROM P JOIN PD USING PNUM;

Зауваження. Ключове слово USING дозволяє *явно вказувати*, за яким з *загальних* атрибутів у таблицях буде проводитися з’єднання.

**Приклад 15**. Природне з’єднання таблиць (спосіб 3 - ключове слово ***NATURAL JOIN***):

SELECT

P.PNUM,

P.PNAME,

PD.DNUM,

PD.VOLUME

FROM P NATURAL JOIN PD;

Зауваження. В розділе FROM не вказано, за якими полям відбувається з’єднання. NATURAL JOIN автоматично з’єднує *за усіма однаковими полями* у таблицях.

**Приклад 16**. Природне з’єднання трьох таблиць:

SELECT

P.PNAME, D.DNAME, PD.VOLUME

FROM

P NATURAL JOIN PD NATURAL JOIN D;

В результаті отримуємо таку таблицю:

|  |  |  |
| --- | --- | --- |
| **PNAME** | **DNAME** | **VOLUME** |
| Іван | Болт | 100 |
| Іван | Гайка | 200 |
| Іван | Гвинт | 300 |
| Петро | Болт | 150 |
| Петро | Гайка | 250 |
| Микола | Болт | 1000 |

**Приклад 17**. Прямий добуток таблиць:

SELECT

P.PNUM,

P.PNAME,

D.DNUM,

D.DNAME

FROM P, D;

В результаті отримуємо таку таблицю:

|  |  |  |  |
| --- | --- | --- | --- |
| **PNUM** | **PNAME** | **DNUM** | **DNAME** |
| 1 | Іван | 1 | Болт |
| 1 | Іван | 2 | Гайка |
| 1 | Іван | 3 | Гвинт |
| 2 | Петро | 1 | Болт |
| 2 | Петро | 2 | Гайка |
| 2 | Петро | 3 | Гвинт |
| 3 | Микола | 1 | Болт |
| 3 | Микола | 2 | Гайка |
| 3 | Микола | 3 | Гвинт |

Зауваження. Виходячи з того, що не вказано умову з’єднання таблиць, *кожний рядок* першої таблиці з’єднується з *кожним рядком* другої таблиці.