# Бази даних та інформаційні системи

## Тема 10. Data Manupulation Language (DML)

СумДУ, каф.КН

2020

# Задачі

- Сьогодні ми маємо дізнатися:
  - Як вставити дані в таблицю;
  - Як оновити дані в таблиці;
  - Як видалити дані з таблиці.

# Оператори SQL

| Оператори | Призначення |
|---|---|
| **SELECT** | Отримання даних |
| **INSERT**<br>**UPDATE**<br>**DELETE** | Мова маніпулювання даними<br>Data manipulation language (DML) |
| **CREATE**<br>**ALTER**<br>**DROP**<br>**RENAME**<br>**TRUNCATE** | Мова визначення даних<br>Data definition language (DDL) |
| **COMMIT**<br>**ROLLBACK**<br>**SAVEPOINT** | Управління транзакціями<br>Transaction control (TC) |
| **GRANT**<br>**REVOKE** | Мова управління даними<br>Data control language (DCL) |

# INSERT

Оператор **INSERT** дає змогу вводити дані в таблицю.

Є два різновиди цього оператора:

```
1) INSERT INTO … VALUES (…)
2) INSERT …  SELECT
```

1. За один раз вставляється тільки один рядок.

```
INSERT INTO table [(column [, column...])]
VALUES    (value [, value...]);
```

# Insert into … values (…);

Вставляє 1 рядок

```
INSERT INTO table [(column [, column...])]
VALUES    (value [, value...]);
```

Певні правила:

- Типи даних, що вставляються, мають збігатися з типами даних відповідних стовпців;
- Розміри даних мають відповідати розмірам стовпців;
- Порядок даних у фразі **VALUES** має відповідати порядку стовпців у фразі **INSERT INTO**

```
SQL> INSERT INTO dept (depno,dname)
   2        VALUES (1, 'New dep')
```

# Insert into … values (…)

- Які значення можна використовувати:
  - Вирази (в т.ч. з функціями);
  - **`Null`**;
  - **`Default`**.

```
SQL> INSERT INTO dept(depno, dname,loc)
   2   VALUES (20, 'New dep', DEFAULT)
```

# Як перевірити, вставлені значення чи ні?

```
SQL> INSERT INTO dept (depno, dname, loc)
   2   VALUES (20, 'New dep', DEFAULT);
```

```
SQL> SELCT *
2  FROM dept
3  WHERE depno = 20
4  AND  dname = 'New dep';
```

# Додавання рядків з NULL значеннями

- **Неявний** метод: пропустити стовпці з **NULL**-значеннями:

```
INSERT INTO   dept (deptno, dname)
VALUES     (30, 'Purchasing' );

1 rows inserted
```

- **Явний** метод: замість значень підставити константу **NULL**:

```
INSERT INTO   dept
VALUES     (100, 'Finance', NULL);

1 rows inserted
```

# Insert into …

- Якщо в новому записі вказані значення всіх атрибутів перелік атрибутів можна не вказувати

```
SQL> INSERT INTO DEPT (deptno, dname, loc)
   2        VALUES (1, 'New dep', null)
```

```
SQL> INSERT INTO DEPT
   2        VALUES (1, 'New dep', null)
```

# Додавання спеціальних значень

- Функція **SYSDATE** містить поточну дату

```
INSERT INTO empl(empno, ename,hiredate,
                                 mgr, deptno)
VALUES      (113, 'Louis', SYSDATE,
                                      205, 10);
```

```
1 rows inserted
```

# Додавання певних значень часу

```
INSERT INTO empl (empno, ename,
        hiredate,  mng, depno)
VALUES      (113, 'Louis',
      TO_DATE('02 19, 2013', 'MM DD, YYYY'),
            205, 110);
```

```
1 rows inserted
```

# Які поля можна пропустити?

```
SQL> CREATE TABLE STUDENT(
2     STUDENT_ID NUMBER PRIMARY KEY,
3     SURNAME  VARCHAR(25),
4     NAME  VARCHAR(10),
5     STIPEND NUMBER,
6     CITY  VARCHAR(15) DEFAULT 'SUMY',
7     BIRTHDAY DATE NOT NULL,
8     UNIV_ID INTEGER CHECK (UNIV_ID BETWEEN 10 AND 99),
9     CONSTRAINT STUD_CHECK CHECK (
10          STIPEND > 100500 AND CITY = 'SUMY')
11 );
```

# Додавання значень із запиту. INSERT … SELECT

Оператор **INSERT ... SELECT** дає змогу додати до таблиці множину рядків (результат виконання запиту).

Дозволяє копіювати інформацію з однієї чи кількох таблиць.

Команда вставки значень із запиту має синтаксис:

```
INSERT INTO table [(column [, column...])]
select-query
```

# Копіювання рядків з іншої таблиці

- Приклад:

```
INSERT INTO sales_reps(id, name, salary, commission_pct)
  SELECT empno, ename, sal, comm
  FROM   emp
  WHERE  job LIKE 'SAIL%';
4 rows inserted
```

- *Пам'ятайте*:
  - При такому записі **не використовується** специфікатор **VALUES**;
  - Кількість і тип стовпців таблиці, в яку додає значення **INSERT** має збігатися зі стовпцями підзапиту.

# UPDATE

- Для оновлення наявних значень використовується оператор **UPDATE** :

```
UPDATE    table
SET       column = value [, column = value, ...]
[WHERE    condition];
```

- **UPDATE** може змінювати кілька рядків за один раз (якщо це потрібно).

# UPDATE

- Для того щоб вказати, які рядки потрібно оновити використовується умова **WHERE**:

```
UPDATE  emp
SET     depno = 50
WHERE   empno = 113;
1 rows updated
```

- Якщо не вказати умову **WHERE** - оновляться значення всіх рядків:

```
UPDATE    emp
SET       depno = 50;
22 rows updated
```

- Специфікатор **SET** column_name = **NULL** встановлює значення атрибута в **NULL**.

# UPDATE

- Встановити всім співробітникам премію в розмірі 12% старої зарплати і додати до зарплати 15%

```
UPDATE    emp
SET   comm   = sal*0.12, sal=sal+sal*0.15;
```

# Використання підзапитів при оновленні

- Встановити службовцю 113 посаду і зарплату службовця 205.

```
UPDATE    emp
SET       job      = (SELECT   job
                      FROM     emp
                      WHERE    empno = 205),
          sal      = (SELECT   sal
                      FROM     emp
                      WHERE    empno = 205)
WHERE     empno     =  113;
```

```
1 rows updated
```

# Оновлення рядків на основі інших таблиць

- Приклад оновлює таблицю COPY_EMP на основі значень таблиці EMP. Він змінює номера підрозділів усіх працівників, які мають посаду працівника 200, на номер відділу працівника 100.

```
UPDATE   copy emp
SET      depno   =   (SELECT depno
                      FROM emp
                      WHERE empno = 100)
WHERE    job     =   (SELECT job
                      FROM emp
                      WHERE empno = 200);
```

```
1 rows updated
```

# DELETE

- Оператор **DELETE** видаляє рядки з таблиці:

```
DELETE [FROM] table
[WHERE  condition];
```

# Видалення рядків з таблиці

- Для визначення, які рядки будуть видалені використовуйте **WHERE**:

```
DELETE FROM departments
WHERE  department_name = 'Finance';
```

```
1 rows deleted
```

- Якщо не вказати умови видалення будуть видалені всі рядки:

```
DELETE FROM  copy_emp;
```

```
22 rows deleted
```

# Підзапит в умові видалення

Видалення рядків на основі іншої таблиці.
Приклад видаляє всіх працівників у відділі, де назва відділу містить рядок Public.

```
DELETE FROM empl
WHERE  deptno IN
                (SELECT deptno
                 FROM   dept
                 WHERE  dname LIKE '%Public%');
```

```
1 rows deleted
```

# Delete vs Drop

- У чому різниця між :

```
SQL> drop table departments;
SQL> delete from departments;
```

# TRUNCATE

- Очищує таблицю.
- Відноситься до **`data definition language`** (DDL), а не до DML.
- Синтаксис:

```
TRUNCATE TABLE table_name;
```

```
TRUNCATE TABLE emp;
```

# Конструкція MERGE, вставка - оновлення

- Дозволяє доповнювати і оновлювати дані однієї таблиці - даними іншої таблиці.
- При злитті таблиць перевіряється умова, і якщо вона істинна, то виконується **`UPDATE`**, а якщо ні - **`INSERT`**.
- Не можна змінювати поля таблиці в секції **`Update`**, за якими йде зв'язування двох таблиць.
- **`MERGE`** не можна використовувати для оновлення одного рядка більше одного разу, а також для оновлення та видалення одного і того ж рядка.

# Merge

```
MERGE INTO {table_name|view_name}
  USING {table_name|view_name|subquery}
   ON {conditions}
[WHEN MATCHED THEN
   UPDATE SET {col=value, …}
  [WHERE {conditions}]
  [DELETE {conditions}]
]
[WHEN NOT MATCHED THEN
   INSERT [({column, …})]
   VALUES ({expr,…})
   [WHERE {conditions}]
]
```

# Merge

```
CREATE TABLE bonuses(
 employee_id NUMBER,
 bonus NUMBER DEFAULT 100
);

INSERT INTO bonuses(employee_id)
SELECT e.empno FROM emp e;
```

# Merge

```
SELECT * FROM bonuses ORDER BY employee_id;
```

```
EMPLOYEE_ID BONUS
----------- ----------
153             100
154             100
155             100
156             100
158             100
159             100
160             100
161             100
163             100
```

# Merge

```
MERGE INTO bonuses D
USING (
   SELECT empno, sal salary, deptno
   FROM emp
   WHERE deptno = 80) S
ON (D.employee_id = S.empno)
WHEN MATCHED THEN
   UPDATE SET D.bonus = D.bonus + S.salary*.01
   DELETE WHERE (S.salary > 8000)
WHEN NOT MATCHED THEN
   INSERT (D.employee_id, D.bonus)
   VALUES (S.empno, S.salary*0.01)
   WHERE (S.salary <= 8000);
```

# Merge

```
SELECT * FROM bonuses ORDER BY employee_id;
```

```
EMPLOYEE_ID BONUS
----------- ----------
153         180
154         175
155         170
159         180
160         175
161         170
164         72
165         68
166         64
167          62
```