



# Бази даних та Інформаційні системи

## 4.2 Посилальна цілісність

СумДУ, каф. КН  
2020

# Задачі заняття

---

- ▶ Після завершення заняття ви повинні вміти і знати наступне:
  - ▶ встановлювати обмеження на значення атрибутів;
  - ▶ створювати і підтримувати обмеження;
  - ▶ створювати первинні і зовнішні ключі;
  - ▶ перетворювати ERD в таблицю.



# Перетворення ERD в структуру таблиць

---

1. Кожна проста сутність стає таблицею.
2. Кожен атрибут стає стовпчиком таблиці. Обов'язковим атрибутам встановлюють обмеження **not null**.
3. Ідентифікатори сутностей стають ключами. Один з ідентифікаторів стає первинним ключем.
4. Зв'язки «один-до-одного» і «один-до-багатьох» перетворюються в зовнішні ключі.



# Співвідношення між таблицями

- Кожен запис однозначно визначається первинним ключем (**PK**)
- Ви можете логічно пов'язати дані різних таблиць за допомогою зовнішнього ключа (**FK**)

Ім'я таблиці : EMP

EMPNO	ENAME	JOB	DEPTNO
7839	KING	PRESIDENT	10
7698	BLAKE	MANAGER	30
7782	CLARK	MANAGER	10
7566	JONES	MANAGER	20

Ім'я таблиці : DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

Первинний ключ

Зовнішній ключ

Первинний ключ

# Що таке обмеження?

---

- ▶ Визначає правило на рівні таблиці.
- ▶ Запобігає видаленню таблиці в разі наявності залежностей.

У Oracle існують такі види обмежень :

- ▶ **NOT NULL**
- ▶ **UNIQUE Key**
- ▶ **PRIMARY KEY**
- ▶ **FOREIGN KEY**
- ▶ **CHECK**



# Рекомендації до використання

---

- ▶ Давайте імена обмеженням, інакше Oracle Server створить ім'я у вигляді **SYS\_Cn**.
- ▶ Створити обмеження можна:
  - ▶ При створенні таблиці;
  - ▶ Після того, як таблиця була створена.
- ▶ Обмеження можна задати на рівні таблиці і на рівні стовпця.
- ▶ Обмеження можна переглянути в каталозі даних.





# Визначення обмежень

---

- ▶ На рівні стовпця

```
column datatype[CONSTRAINT constraint_name] constraint_type,
```

- ▶ На рівні таблиці

```
column datatype, ...  
[CONSTRAINT constraint_name] constraint_type
```

```
CREATE TABLE NEW_EXAM_MARKS (  
    STUDENT_ID INTEGER NOT NULL,  
    SUBJ_ID INTEGER,  
    MARK INTEGER,  
    CONSTRAINT EXAM_PR_KEY PRIMARY KEY (STUDENT_ID, SUBJ_ID));
```



# Таблиці, що використовуються в курсі

## EMP

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		17-NOV-81	5000		10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	1500		10
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30

## DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO



# Обмеження NOT NULL

- ▶ Гарантує, що в стовпці не можуть бути використані значення **Null**

EMP

EMPNO	ENAME	JOB	...	COMM	DEPTNO
7839	KING	PRESIDENT			10
7698	BLAKE	MANAGER			30
7782	CLARK	MANAGER			10
7566	JONES	MANAGER			20
...					

Обмеження NOT NULL  
(Ні один рядок не може містити null в цьому стовпці)

Обмеження NOT NULL  
не встановлене  
(Будь-який рядок може містити null в цьому стовпці)

Обмеження NOT NULL

# Обмеження NOT NULL

---

- Визначаються тільки на рівні стовпця

```
SQL> CREATE TABLE emp (  
2      empno      NUMBER (4) ,  
3      ename      VARCHAR2 (10) NOT NULL ,  
4      job        VARCHAR2 (9) ,  
5      mgr        NUMBER (4) ,  
6      hiredate   DATE ,  
7      sal        NUMBER (7,2) ,  
8      comm       NUMBER (7,2) ,  
9      deptno     NUMBER (7,2) NOT NULL) ;
```

# NULL

---

- ▶ **NULL** – позначення порожнечі (нестачі інформації).
- ▶ З **NULL** не можна порівнювати

Умова	Значення A	Результат
a IS NULL	10	FALSE
a IS NOT NULL	10	TRUE
a IS NULL	NULL	TRUE
a IS NOT NULL	NULL	FALSE
a = NULL	10	UNKNOWN
a != NULL	10	UNKNOWN
a = NULL	NULL	UNKNOWN
a != NULL	NULL	UNKNOWN
a = 10	NULL	UNKNOWN
a != 10	NULL	UNKNOWN

# Обмеження UNIQUE (унікальний ключ)

Обмеження UNIQUE

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

Забезпечує, щоб всі значення в стовпці були різними

Вставка

50	SALES	DETROIT
60		BOSTON

Не буде **вставлено** (DNAME-SALES вже існує)

Буде **вставлено**

# Обмеження UNIQUE

---

- Визначається як на рівні стовпця, так і на рівні таблиці.

```
SQL> CREATE TABLE dept(  
2     deptno      NUMBER(2) ,  
3     dname       VARCHAR2(14) ,  
4     loc         VARCHAR2(13) UNIQUE ,  
5     CONSTRAINT dept_dname_uk UNIQUE(dname)) ;
```

```
CREATE TABLE Student(  
    STUDENT_ID INTEGER,  
    Name VARCHAR(20) ,  
    Surname VARCHAR(20) ,  
    CONSTRAINT STUDENT_UNIQ UNIQUE (Name, Surname)) ;
```

# Обмеження PRIMARY KEY

**PRIMARY KEY**

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

Комбінація NOT NULL і  
UNIQUE.

Унікально ідентифікує  
кожен рядок у таблиці.

**Вставка**

20	MARKETING	DALLAS
	FINANCE	NEW YORK

**Не буде додано**  
(DEPTNO–20 вже існує)

**Не буде додано**  
(DEPTNO має  
значення null)

# Обмеження PRIMARY KEY

Визначається як на рівні стовпця, так і на рівні таблиці.

```
SQL> CREATE TABLE dept (  
  2      deptno      NUMBER(2) PRIMARY KEY,  
  3      dname       VARCHAR2(14) ,  
  4      loc         VARCHAR2(13)  
  ) ;
```

```
SQL> CREATE TABLE dept (  
  2      deptno      NUMBER(2) ,  
  3      dname       VARCHAR2(14) ,  
  4      loc         VARCHAR2(13) ,  
  5      CONSTRAINT dept_deptno_pk PRIMARY KEY(deptno) ) ;
```



# Обмеження FOREIGN KEY

**PRIMARY KEY**

**DEPT**

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
...		

Однозначно  
ідентифікує  
рядок / запис в  
іншій таблиці.

**EMP**

EMPNO	ENAME	JOB	...	COMM	DEPTNO
7839	KING	PRESIDENT			10
7698	BLAKE	MANAGER			30
...					

**FOREIGN KEY**

**Не буде додано**  
(DEPTNO=9  
відсутнє в  
таблиці DEPT)

Вставка

7571	FORD	MANAGER	...	200	9
7571	FORD	MANAGER	...	200	

**Дозволено**

# Обмеження FOREIGN KEY

---

- Визначається як на рівні стовпця, так і на рівні таблиці.

```
SQL> CREATE TABLE emp (  
2      empno      NUMBER(4) ,  
3      ename      VARCHAR2(10) NOT NULL ,  
4      job        VARCHAR2(9) ,  
5      mgr        NUMBER(4) ,  
6      hiredate   DATE ,  
7      sal        NUMBER(7,2) ,  
8      comm       NUMBER(7,2) ,  
9      deptno     NUMBER(7,2) NOT NULL ,  
10     CONSTRAINT emp_deptno_fk FOREIGN KEY (deptno)  
11                REFERENCES dept (deptno)) ;
```

# Ключові слова обмеження FOREIGN KEY

---

- ▶ **FOREIGN KEY**

Визначає стовпець дочірньої таблиці на рівні обмежень таблиці.

- ▶ **REFERENCES**

Визначає батьківську таблицю і стовпець в ній.

- ▶ **ON DELETE CASCADE**

Дозволяє видаляти рядки батьківської таблиці разом з пов'язаними елементами дочірньої таблиці.

- ▶ **ON DELETE SET NULL**

При видаленні з батьківської таблиці в дочірній таблиці значення полів зі зовнішніми ключами встановлюється **NULL**.



# Обмеження CHECK

---

- ▶ Визначає умову, якої повинні задовольняти всі рядки в таблиці
- ▶ Не допускаються такі вирази:
  - ▶ Посилання на псевдостовпці **CURRVAL**, **NEXTVAL**, **LEVEL**, і **ROWNUM**;
  - ▶ Виклики функцій **SYSDATE**, **UID**, **USER**, і **USERENV**;
  - ▶ Запити до значень в інших рядках.

```
... ,  
deptno  NUMBER(2) ,  
CONSTRAINT emp_deptno_ck CHECK (DEPTNO > 10 ) ,  
...
```



# Обмеження CHECK

---

```
CREATE TABLE Student  
( Kod_stud      integer NOT NULL PRIMARY KEY,  
  Name         varchar2 (30) NOT NULL UNIQUE,  
  Bal          integer CHECK ( Bal > 60),  
  Forma_Navch  varchar2(12) CONSTRAINT CHK_St_Formnavch CHECK  
  (Forma_Navch IN ( 'Денна', 'Заочна', 'Дістанційна' ));
```



# Додавання обмежень

---

- ▶ **ALTER TABLE** – оператор модифікації структури таблиці.

```
ALTER TABLE table  
ADD [CONSTRAINT constraint] type (column);
```

- ▶ Обмеження можна додати або видалити, але не модифікувати.
- ▶ Обмеження можна активувати і деактивувати.
- ▶ Обмеження **NOT NULL** додається за допомогою специфікатора **MODIFY**.



# Додавання обмежень

---

- ▶ Додавання в таблицю EMP обмеження **FOREIGN KEY**, яке регламентує, що менеджер повинен вже існувати в таблиці EMP.

```
SQL> ALTER TABLE      emp
      2  ADD CONSTRAINT  emp_mgr_fk
      3                FOREIGN KEY (mgr) REFERENCES emp (empno) ;
Table altered.
```



# Видалення обмежень

---

- ▶ Видалення обмеження на менеджера з таблиці EMP:

```
SQL> ALTER TABLE      emp
      2  DROP CONSTRAINT  emp_mgr_fk;
Table altered.
```

- ▶ Видалення обмеження **PRIMARY KEY** в таблиці DEPT з видаленням всіх залежних обмежень **FOREIGN KEY**:

```
SQL> ALTER TABLE      dept
      2  DROP PRIMARY KEY CASCADE;
Table altered.
```



## Деактивація обмежень

---

- ▶ Використовуйте специфікатор **DISABLE** оператора **ALTER TABLE** для деактивації обмеження.
- ▶ Використовуйте параметр **CASCADE** для видалення всіх залежних обмежень.

```
SQL> ALTER TABLE          emp  
      2  DISABLE CONSTRAINT  emp_empno_pk CASCADE;  
Table altered.
```



# Активация ограничений

---

- ▶ Для активации ограничения, что є деактивоване в даний час, використовуйте специфікатор **ENABLE**.

```
SQL> ALTER TABLE          emp
      2  ENABLE CONSTRAINT    emp_empno_pk;
Table altered.
```

- ▶ Індекси **UNIQUE** і **PRIMARY KEY** автоматично формуються при активції обмежень **UNIQUE** або **PRIMARY KEY**.



# Перегляд обмежень

---

- ▶ Запити до таблиці **USER\_CONSTRAINTS** дозволяють переглянути всі обмеження.

```
SQL> SELECT constraint_name, constraint_type c,  
2         search_condition  
3 FROM   user_constraints  
4 WHERE  table_name = 'EMP';
```

CONSTRAINT_NAME	C SEARCH_CONDITION
-----	- - - - -
SYS_C00674	C EMPNO IS NOT NULL
SYS_C00675	C DEPTNO IS NOT NULL
EMP_EMPNO_PK	P
...	

# Перегляд стовпців, що пов'язані з обмеженнями

- ▶ Переглянути стовпці, що задіяні в обмеженні за ім'ям обмеження, можна за допомогою запиту до **USER\_CONS\_COLUMNS**

```
SQL> SELECT    constraint_name, column_name
      2  FROM      user_cons_columns
      3  WHERE     table_name = 'EMP';
```

CONSTRAINT_NAME	COLUMN_NAME
EMP_DEPTNO_FK	DEPTNO
EMP_EMPNO_PK	EMPNO
EMP_MGR_FK	MGR
SYS_C00674	EMPNO
SYS_C00675	DEPTNO

# Висновки

---

- ▶ Створюйте наступні типи обмежень :
  - ▶ **NOT NULL**
  - ▶ **UNIQUE key**
  - ▶ **PRIMARY KEY**
  - ▶ **FOREIGN KEY**
  - ▶ **CHECK**
- ▶ Запити до таблиці **USER\_CONSTRAINTS** дозволяють переглянути всі обмеження та їх імена.

