Adatbázisok

4. labor: DDL utasítások, megszorítások

SQL utasítások kategóriái

- DQL (Data Query Language)
  - SELECT
- DDL (Data Definition Language)
  - CREATE, ALTER, DROP, TRUNCATE, RENAME
- DML (Data Manipulation Language)
  - INSERT, UPDATE, DELETE
- DCL (Data Control Language)
  - GRANT, REVOKE
- TCL (Transaction Control Language)
  - SAVEPOINT, COMMIT, ROLLBACK

DDL és DML utasítások

- DDL: Data Definition Language
  - **táblák, nézetek** létrehozása, módosítása, törlése
- DML: Data Manipulation Language
  - **sorok** beszúrása, módosítása, törlése

Tábla létrehozása

- Egyszerűbb eset: tábla létrehozása egy lekérdezés eredménye alapján:

```
CREATE TABLE employees2 AS
  SELECT * FROM employees;
```

- Másolat készül!

Tábla létrehozása

- Most azokat másoljuk át új táblába, akiknek 5000 dollárnál több a fizetése!

```
CREATE TABLE employees3 AS
  SELECT * FROM employees
  WHERE salary>5000;
```

Teljesen új tábla létrehozása

- Kötelezően megadandó:
  - a tábla neve
  - az oszlopok (mezők) neve, adattípusa
- Alakja:
  CREATE TABLE táblanév
  (oszlopnév adattípus [, oszlopnév adattípus ...]);
- Példa:

```
CREATE TABLE újtábla (
      számoszlop NUMBER(5,2),
      szövegoszlop VARCHAR2(10),
      dátumoszlop DATE
);
```

A fontosabb adattípusok 1.

- CHAR(n): fix (*n* karakter) hosszú szöveg
  Default: n=1.
- VARCHAR2(n): változó, legfeljebb *n* karakter hosszú szöveg
- VARCHAR(n): a VARCHAR2 adattípussal ekvivalens jelenleg, de jelentése változhat, ezért VARCHAR2 használata javasolt
- CLOB: változó, kb. 8 TB méretű szöveg

A fontosabb adattípusok 2.

- NUMBER(n, m): fixpontos szám, max. *n* decimális számjegy, ebből *m* tizedesjegy
- NUMBER(n): max. *n* számjegyes egész,
  ugyanaz, mint NUMBER(n, 0)
- NUMBER: lebegőpontos, 38 számjegy pontosság

A fontosabb adattípusok 3.

- DATE: dátum és idő (másodperc pontosság)
- TIMESTAMP(n): dátum és idő (a másodperc *n* tizedesjeggyel, default: n=6)
- BLOB: binary large object, „nyers” bináris adat (max. 4 GB)
- **Nincs logikai adattípus!**
- **A használható adattípusok az egyes Oracle verziókban eltérhetnek!**

Alapértelmezett érték megadása

- Példa:

```
CREATE TABLE hallgato (
        nev VARCHAR2(30),
        szul_dat DATE,
        evfolyam NUMBER(1)  DEFAULT 1);
```

- Az alapértelmezett érték nemcsak konstans lehet, hanem kifejezés is, pl.

```
...
belep_dat DATE DEFAULT
TRUNC(SYSDATE)
...
```

Egy tábla szerkezetének lekérdezése

- Példa:
  DESCRIBE hallgato;
- Vagy:
  DESCR hallgato;
- Vagy:
- DESC hallgato;

Módosítás

- Hozzáadás táblához

```
ALTER TABLE táblanév
ADD (...)
```

Példák:

```
ALTER TABLE employees3 ADD
(cardID NUMBER(5));
```

Módosítás

- Módosítás táblában

```
ALTER TABLE táblanév
MODIFY (...)
```

Példák:

```
ALTER TABLE employees3 MODIFY
 cardID NUMBER(7);
```

- A meglévő adatoknak illeszkedni kell az új adattípushoz!

Módosítás

- Oszlop törlése

```
ALTER TABLE táblanév
DROP COLUMN oszlopnév
```

- Oszlop átnevezése

```
ALTER TABLE táblanév
RENAME COLUMN Réginév TO Újnév
```

Módosítás

- Adjunk hozzá egy új oszlopot a másolat-táblánkhoz, amely a dolgozók kedvenc színét tárolja!

```
ALTER TABLE employees3 ADD
  (színe VARCHAR2(10));
```

Tábla átnevezés, törlés

- Tábla átnevezése:

```
RENAME employees3 TO
  employees23;
```

- Tábla kiürítése:

```
TRUNCATE TABLE employees23;
```

- Tábla törlése:

```
DROP TABLE employees23;
```

Saját tábláink adatai

- USER_TABLES nézet
- A táblák felsoroltatása: SELECT table_name FROM user_tables;
- Sok adminisztratív információhoz is hozzá lehet itt férni

Gyakorlás

- Hozzuk létre a következő táblát **kutya** néven:
  - **ID**: legfeljebb 3 jegyű egész szám
  - **név**: legfeljebb 20 karakter hosszú szöveg
  - **nem**: legfeljebb 1 jegyű egész szám
  - **szüldátum**: dátum

Gyakorlás

- Az előbb létrehozott **kutya** nevű táblához adjunk egy oszlopot gazd_ID névvel, mely legfeljebb 6 jegyű egész számot tartalmaz.
- Nevezzük át a táblát **kutyusok**-ra
- Töröljük ki a táblát!

Megszorítások

- PRIMARY KEY – elsődleges kulcs
- FOREIGN KEY – idegen kulcs

Előadás!

- NOT NULL – az értéke nem lehet NULL
- UNIQUE – minden érték csak egyszer szerepel
- CHECK – az értéknek meg kell felelnie a megadott feltételnek

PRIMARY KEY

```
CREATE TABLE departments2(
  department_id NUMBER(4),
  department_name VARCHAR2(30),
  manager_id NUMBER(6),
  location_id NUMBER(4),
  CONSTRAINT d_pk
  PRIMARY KEY (department_id)
);
```

- out-of-line deklaráció

PRIMARY KEY

```
CREATE TABLE departments2(
  department_id NUMBER(4)
  CONSTRAINT d_pk PRIMARY KEY,
  department_name VARCHAR2(30),
  manager_id NUMBER(6),
  location_id NUMBER(4)
);
```

- inline deklaráció, megszorításnév megadásával

PRIMARY KEY

```
CREATE TABLE departments2 (
 department_id NUMBER(4)
 PRIMARY KEY,
 department_name VARCHAR2(30),
 manager_id NUMBER(6) ,
 location_id NUMBER(4)
);
```

- inline deklaráció, megszorításnév megadása nélkül (ABK automatikusan generálja)

FOREIGN KEY

```
CREATE TABLE departments2 (
 department_id NUMBER(4),
 department_name VARCHAR2(30),
 manager_id NUMBER(6),
 location_id NUMBER(4),
 CONSTRAINT
 department_manager_id
 FOREIGN KEY (manager_id)
 REFERENCES employees
 (employee_id)
);
```

FOREIGN KEY

```
CREATE TABLE departments2(
  department_id NUMBER(4),
  department_name VARCHAR2(30),
  manager_id NUMBER(6) REFERENCES
  employees (employee_id),
  location_id NUMBER(4)
);
```

- inline deklaráció esetén nem kell a FOREIGN KEY rész
- a hivatkozott oszlopot nem kötelező megadni, a hivatkozott tábla elsődleges kulcsát feltételezi

FOREIGN KEY opciók

- Alapértelmezett viselkedés: szülő rekord nem törölhető, amíg van rá hivatkozó gyermek rekord (Tábla eldobását is akadályozhatja!)
- ON DELETE CASCADE
  - A szülő rekord törlésekor a gyermek rekordok is törlődnek.
- ON DELETE SET NULL
  - A szülő rekord törlésekor a gyermek rekordokban NULL értékre állítódik a szülőre hivatkozó mező.

ON DELETE CASCADE - példa

```
CREATE TABLE supplier (
supplier_id number(10) not null,
supplier_name varchar2(50) not null,
contact_name varchar2(50),
CONSTRAINT supplier_pk PRIMARY KEY (supplier_id) );

CREATE TABLE products (
product_id number(10) not null,
supplier_id number(10) not null,
CONSTRAINT fk_supplier FOREIGN KEY (supplier_id)
REFERENCES supplier(supplier_id) ON DELETE CASCADE );
```

UNIQUE

```
CREATE TABLE employees(
employee_id NUMBER(6) ,
first_name VARCHAR2(20) ,
last_name VARCHAR2(25),
email VARCHAR2(25),
…
department_id NUMBER(4) ,
CONSTRAINT emp_email_uk
UNIQUE (email));
```

NOT NULL

```
CREATE TABLE departments3(
department_id NUMBER(4),
department_name VARCHAR2(30)
CONSTRAINT dept_name_nn3 NOT
NULL ,
manager_id NUMBER(6) ,
location_id NUMBER(4));
```

- csak inline deklarálható
- ha mégis mindenképp out-of-line akarjuk, akkor CHECK

CHECK

```
CREATE TABLE employees(
employee_id NUMBER(6) ,
first_name VARCHAR2(20) ,
last_name VARCHAR2(25),
salary NUMBER(8,2) ,
…,
department_id NUMBER(4) ,
CONSTRAINT emp_salary_min
CHECK (salary > 0) );
```

Gyakori feltételek ellenőrzésnél 1.

- Egyszerű összehasonlítás
  - CHECK (x <= 0) -- x nem pozitív
  - CHECK (x = y) -- x = y
  - CHECK (x <> ‘C’) -- x nem egyenlő ‘C’-vel
- Intervallum belseje
  - CHECK (x >= 0 AND x <= 10) -- x>=0 és x<=10
  - CHECK (x BETWEEN 0 AND 10) -- ugyanaz
- Intervallum külseje
  - CHECK (x < 0 OR x > 10) -- x<0 vagy x>10
  - CHECK (x NOT BETWEEN 0 AND 10) -- ugyanaz

Gyakori feltételek ellenőrzésnél 2.

- Szöveg hasonlítása mi
  - CHECK (x LIKE ‘A_b%’)
    (x első karaktere A, harmadik karaktere b)
- Előfordulás felsorolásban
  - CHECK (x IN (‘a’, ‘b’, ‘c’)) -- a vagy b vagy c
  - CHECK (x IN (SELECT nev FROM hallgato))
    (x előfordul a HALLGATO tábla NEV oszlopában)
- Kizárás felsorolásból
  - CHECK (x NOT IN (1, 2, 3)) -- nem 1, 2

_ egy tetszőleges karak
% akárhány tetszőleges

Jó tudni …

- A rendszer a kényszerek teljesülését minden rekord létrehozásakor, törlésekor vagy módosításakor ellenőrzi
- Tömeges adatmódosításnál (pl. adat-import) sok időt vesz igénybe
- A kényszerek átmenetileg kikapcsolhatók!

Megszorítás hozzáadása

```
ALTER TABLE táblanév
ADD CONSTRAINT ...

vagy

ALTER TABLE táblanév
MODIFY ...
```

Megszorítás hozzáadása

Példák:

```
ALTER TABLE employees ADD
  CONSTRAINT minimálbér CHECK
  (salary>2000);
```

vagy

```
ALTER TABLE employees
MODIFY salary CHECK
  (salary>1000);
```

Megszorítás hozzáadása

```
ALTER TABLE employees ADD
  CONSTRAINT minimálbér CHECK
  (salary>3000);
```

- Nem működik! Csak olyan megszorítást adhatunk meglévő táblához (vagy engedélyezhetünk rá), ami nem mond ellent a benne levő adatoknak!

Egyéb műveletek

Egyéb ALTER TABLE utasításhoz kapcsolódó megszorítás műveletek:

- Megszorítás törlése
  DROP CONSTRAINT megszorításnév;
- Megszorítás engedélyezése
  ENABLE CONSTRAINT megszorításnév;
- Megszorítás tiltása
  DISABLE CONSTRAINT megszorításnév;

Példa:

```
ALTER TABLE employees DROP
CONSTRAINT minimálbér;
```

Megszorítások lekérdezése

- A beépített `user_constraints` nézetből lekérdezhetők a megszorítások, azok típusa és állapota.
- Lényeges mezők:
  - constraint_name: a megszorítás neve
  - constraint_type: a megszorítás típusa
  - table_name: melyik táblához kapcsolódik

Megszorítások lekérdezése

```
SELECT constraint_name,
  constraint_type,
  table_name
FROM user_constraints
WHERE lower(table_name) IN
('employees',
  'departments');
```

Gyakorlás

- A már meglévő kutyusok táblában állítsuk be a következőket:
  - Elsődleges kulcs: **ID**
  - A **nem** csak 0 vagy 1 lehet.
  - A **név** nem maradhat üresen.
  - Idegen kulcs: gazd_ID, mely az employees tábla employee_ID oszlopára mutat
- Adjunk hozzá egy új oszlopot a táblához, melyre UNIQUE típusú megszorítás rakható. Adjuk is ezt hozzá!
- Végezzük el az előző feladatot egyetlen CREATE TABLE utasítás kiadásával!

Megoldás

- ALTER TABLE kutyusok ADD CONSTRAINT k_pk PRIMARY KEY (ID);
- ALTER TABLE kutyusok ADD CONSTRAINT k_ck CHECK (nem IN(0, 1));
- ALTER TABLE kutyusok ADD CONSTRAINT k_ck2 CHECK (nev IS NOT NULL);
- ALTER TABLE kutyusok ADD CONSTRAINT k_fk FOREIGN KEY (gazd_ID) REFERENCES employees (employee_id);
- ALTER TABLE kutyusok ADD (email VARCHAR2(20) UNIQUE);

Megoldás

```
CREATE TABLE kutyusok (
        ID NUMBER(3),
        nev VARCHAR2(20),
        nem NUMBER(1),
        szuldatum DATE,
        gazd_ID NUMBER,
        email VARCHAR2(25),
        CONSTRAINT k_pk PRIMARY KEY (ID),
        FOREIGN KEY (gazd_id)
        REFERENCES employees (employee_id),,
        CONSTRAINT k_ck CHECK (nem IN(0, 1)),
        CONSTRAINT k_ck2 CHECK (nev IS NOT NULL),
        CONSTRAINT k_un UNIQUE (email)
);
```

Indexek

Alapelvek

- Az indexek az adott mező(k) szerinti keresést és rendezést gyorsító segédobjektumok (nem adattáblák!)
- Azokat a mezőket indexeljük, amik szerint gyakran keresünk vagy rendezünk
- A jó index gyorsítja a lekérdezést
- A kulcs szerint automatikusan készül index is
- A felesleges index lassítja az adatbázist - az indexek karbantartása is időbe kerül!
- **Az idegen kulcsokat általában érdemes indexelni (join-okra tekintettel)**

Index létrehozása

- Formája:
  CREATE [UNIQUE] INDEX indexnév
  ON táblanév (mezőnév [sorrend]
  [, mezőnév [sorrend] …]);
  - sorrend: ASC = növekvő (alapértelmezés), DESC = csökkenő
  - UNIQUE: az index alapjául szolgáló mezők kötelezően egyediek
- Példa:
  CREATE INDEX hallg_nevsor_idx
  ON hallgato (nev, evfolyam DESC);
  - név szerint növekvő, azonos nevűeknél évfolyam szerint csökkenő sorrend

Értékek egyediségének ellenőrzése indexeléssel

- A mezőre UNIQUE indexet készítünk, pl. CREATE UNIQUE INDEX hallg_nev_idx ON hallgato (nev);
- **1:N kapcsolat esetén az idegen kulcshoz sose készítsünk UNIQUE indexet!**

Index törlése

- Formája:
  DROP INDEX indexnév;
- Példa:
  DROP INDEX hallg_nevsor_idx;

Jó tudni ...

- A rendszer az indexeket minden rekord létrehozásakor, törlésekor vagy módosításakor aktualizálja
- Tömeges adatmódosításnál (pl. adat-import) sok időt vesz igénybe
- Sokszor érdemesebb az indexeket törölni, majd az adatmódosítások után újra létrehozni!