2. gyakorlat

Történelem
Az SQL alapjait az IBM-nél fektették le, még az 1970-es években. Elvi alapja a relációs adatmodell.
Az SQL nyelv deklaratív nyelv, ezért alapvetően más, mint a megszokott C++ vagy Java.

DDL (Data Definition Language) – Adat definíciós utasítások
Ide tartoznak az adatbázis létrehozásához, átalakításához tartozó utasítások.

• CREATE - az adatbázisban valamilyen elem, pl. tábla, elsődleges vagy külső kulcs, view, létrehozása
• ALTER – minden, amit a CREATE-tel létrehoztunk, ezzel módosítható
• DROP - ezzel meg törölhető

DML (Data Manipulation Language) – Adatkezelő utasítások
Ide tartoznak az adatbázisban tárolt adatokhoz közvetlenül kapcsolódó utasítások

• INSERT - a definiált adatbázisba ezzel az utasítással helyezhetők el adatok
• UPDATE - ezzel meg frissíthetők
• DELETE - vagy törölhetők
• SELECT - talán az egész SQL legfontosabb utasítása: ezzel tudunk az adatbázisból adatokat lekérdezni

Egyebek
Számos beépített függvény található. Általában a feladatokat ezeknek a segítségével érdemes megoldani.
Lehet definiálni saját függvényeket és eljárásokat is, később ezekről is lesz szó gyakorlatokon.

I. Rész – CREATE

CREATE TABLE táblanév
(<attribútum típus[megszorítás]>
[,[attribútum típus[megszorítás]
[, stb]]
);
<kötelező paraméter> [opcionális paraméter]

Pl.:

```
CREATE TABLE dolgozo(
nev VARCHAR2(40),
kor NUMBER,
munkakor VARCHAR2(60)
);
```

A leggyakoribb Oracle adattípusok:
zárójelben (szélesség) – hány karakter hosszú
VARCHAR(size) – változó hosszú, max 2000byte
VARCHAR2(size) -változó hosszú, max 4000byte
CHAR(size) – fix hosszú
(NCHAR(size) – nemzeti karakterek is)
NUMBER - számok
DATE – dátum és idő (!!) típus - legfeljebb az ember az időt nem használja, de erre külön figyelni kell!
ezen kívül még számos egyéb típus:
http://docs.oracle.com/cd/B28359_01/server.111/b28318/datatype.htm
Megszorítások (constraints):
NOT NULL – Az adott oszlopban nem lehet 'NULL' érték.
UNIQUE – Az oszlopban lévő összes érték különböző
CHECK – Az oszlopban lévő összes érték megfelel valamilyen kritériumnak
PRIMARY KEY – Egyértelműen azonosítja a táblázatban az adott sort
FOREIGN KEY – Az adott érték valamelyik táblázatban elsődleges kulcs

---

1. Feladat. Hozd létre az alábbi táblákat sql-ben! Melyik oszlopnak mi lesz a típusa?

patient

patient_id	p_name	sex	admission_date	alzheimer_diagnosis
P1500	Irvin Brody	male	24-10-2004	mild
P9700	Clifton Norman	male	02-08-2010	severe
P9500	Arden Rodger	female	04-09-2010	moderate
P4000	Harland Wilbur	male	17-06-2008	moderate
P8000	Henry Kip	male	28-07-2009	severe

treatment

Treatment_id	Patient_id	Drug	DCost	Adm_time	consultant
T001	P1500	Donepezil	57	24-10-2004 21:18:27	Dr. Green
T002	P9700	Memantine	128	02-08-2010	Dr. Green
T003	P1500	Donepezil	55	31-10-2004 09:12:43	Dr. Green

SQL kód megformázása itt:
http://www.dpriver.com/pp/sqlformat.htm

II. Rész – ALTER

- új attribútum felvétele egy adott táblához
- attribútum törlése egy adott táblából
- megszorítás (constraint) hozzáadása a táblához

ALTER TABLE táblanév
<ADD attrib típus megszorítás [, attrib típus megszorítás]
| MODIFY attribnév adattípus megszorítás[, attribnév adattípus megszorítás]
| DROP COLUMN attribnév>;

Példák:
ALTER TABLE Tanar ADD szuletesiev NUMBER(4);
ALTER TABLE dolgozo DROP COLUMN kor;
ALTER TABLE dolgozo MODIFY szemelyiszam PRIMARY KEY;

1. **Feladat.** A *patient* táblához add hozzá az *age* attribútumot!

patient_id	p_name	sex	admission_date	alzheimer_diagnosis	age
P1500	Irvin Brody	male	24-10-2004	mild	46
P9700	Clifton Norman	male	02-08-2010	severe	85
P9500	Arden Rodger	female	04-09-2010	moderate	72
P4000	Harland Wilbur	male	17-06-2008	moderate	69
P8000	Henry Kip	male	28-07-2009	severe	73

2. **Feladat.** Módósítsd úgy a *patient* táblát, hogy a p_*name* attribútum megadása kötelező legyen!
3. **Feladat.** A *treatment* nevű táblából töröld a *consultant* attribútumot!

III. Rész – DROP

DROP TABLE táblanév [CASCADE CONSTRAINT][PURGE];
pl.
DROP TABLE Tanar;
DROP TABLE dolgozo PURGE - eldobja a táblát és nem rakja kukába
DROP TABLE Tanar CASCADE CONSTRAINT - eldobja a táblát és a függőségeket is.

4. **Feladat.** Töröljétek a *patient* táblát!

IV. Rész – INSERT

A létrehozott táblába ezzel lehet adatot elhelyezni.
INSERT INTO táblanév [(attr1[,attr3[,attrx]])] VALUES (ertek[,ertek[,ertek]]);

pl.:
INSERT INTO dolgozo (nev, kor, munkakor) VALUES ('Béla', 40, 'Raktáros');
INSERT INTO dolgozo VALUES ('István', 32, 'Targoncás');

Dátum attribútum megadásához lehet használni a to_date() függvényt!
CREATE TABLE temp_date(
a DATE
)
INSERT INTO temp_date (a) VALUES(
to_date('1962-05-20', 'yyyy-mm-dd')
);

5. **Feladat.** Töltsétek fel az alábbi adatokkal a korábban definiált táblát (ha töröltétek, akkor hozzátok újra létre)!

patient_id	name	sex	age	admission_date	alzheimer_diagnosis
P1500	Irvin Brody	male	46	24-10-2004	mild
P9700	Clifton Norman	male	85	02-08-2010	severe
P9500	Arden Rodger	female	72	04-09-2010	moderate
P4000	Harland Wilbur	male	69	17-06-2008	moderate
P8000	Henry Kip	male	73	28-07-2009	severe

V. Rész – INSERT

SELECT:
Ha csillagot írunk (**attribnevek helyett**), akkor mindent felsorol.
SELECT attribnevek
FROM tablanevek
[WHERE feltételek];
Részletesebben:

SELECT *oszloplista* (mit szeretnénk látni a kimeneten - projekció)
FROM *táblalista* (miből szeretnénk látni, eddig a kötelező rész)
WHERE *logikai kifejezés* (mivel szűrünk - szelekció)
ORDER BY *logikai rendezés* (csökkenő-növekvő sorba rendez)

Példa:
SELECT * FROM dolgozo;

6. **Feladat.** Milyen recordok vannak a patient táblában?
7. **Feladat.** Válaszd ki a patient_id, név és alzheimer_diagnozis oszlopokat!
8. **Feladat.** Listázd ki a 70 évesnél öregebb betegek adatait!
9. **Feladat.** Listázd ki a betegeket az aktuális életkorukkal („age" mező: életkor az első felvételkor)
10. **Feladat.** Listázd ki azokat a pacienseket koruk szerint csökkenő, sorrendbe rendezve (azonos kor esetén névsorban), akik 2005.01.01. után kerültek felvételre!

VI. Rész – Update, Delete

1. Update

Ha valamilyen recordot frissítenünk kell, pl. megváltozott a beteg neve, akkor az update parancsot lehet használni:

```
UPDATE table_name
SET column1=value1,column2=value2,...
WHERE some_column=some_value;
```

Pl.:

```
UPDATE dolgozo
SET    name = 'Géza'
WHERE  name = 'Károly'
```

2. Feladatok

a. P9500 azonosítójú beteg diagnózisa megváltozott moderate-ről severe-re. Ennek megfelelően frissítsd az adatbázist!
b. Opcionális: Frissítsd az életkort úgy, hogy az aktuális életkort mutassa, ne pedig a felvételkorit! (Aktuális dátum: `SYSDATE`)

3. Delete

Szintén gyakori, hogy egy rekordot, vagy rekordokat törölni kell az adatbázisból.

```
DELETE FROM table_name
WHERE some_column=some_value;
```

Pl.: Az összes Károly nevű dolgozó törlése:

```
DELETE FROM dolgozo
WHERE name = 'Károly'
```

4. Feladatok

a. Töröld az összes nőnemű beteget!
b. A Henry Kip nevű beteg meghalt, töröld az adatbázisból!
c. Töröld az össze rekordot a táblából! (Where klóz elhagyható)

VII. Rész – Where klóz (CONT')

1. Pattern matching

Akkor használjuk, ha valamilyen karaktermintázatot akarunk megtalálni. Például azt kérdezzük, hogy mely dolgozókat hívják Károlynak. Ezt fejezhetjük ki a LIKE kulcsszóval.

'%' – tetszőleges és bármilyen hosszúságú stringgel való egyezés (Figyelem, nem a pl. kereséseknél vagy reguláris kifejezésekben szokásos '*')

'_' – egyetlen karakterben való egyezés (Figyelem, nem a szokásos '?'!)

Pl.:

```
SELECT *
FROM   dolgozo
WHERE  name LIKE '%Károly%'
```

Ha arra vagyunk kíváncsiak, hogy melyek azok a nevek, amik 'A' betűvel kezdődnek és a kettővel rákövetkező karakter is 'A':

```
SELECT *
FROM   dolgozo
WHERE  name LIKE 'A_A%'
```

2. Feladatok pattern matching-hez

Ha az összes rekordot törölted korábban, akkor a wikin lévő sql script alapján visszatöltheted az adatokat a táblába.

a) Mik az 'ar' vagy 'Ar' karaktereket tartalmazó betegek adatai?
b) Mi az azonosítója és a kora a ' Henry' stringgel kezdődő betegeknek?

3. Null értékek kezelése

SQL-ben, ha csak máshogy nem rendelkezünk, lehetnek 'NULL' értékek. Ekkor azonban az összehasonlító operátorokat (<>,=,<) és a függvényeket óvatosan kell használni. Erre megoldás az IS NOT NULL és az IS NULL használata.

```
SELECT *
FROM   patient
WHERE  name IS NULL
```

4. Feladatok null-ok kezeléséhez:

a) Hozzatok létre, majd töltsetek fel egy táblát úgy, hogy a következő adatokat tartalmazza, majd válasszátok ki azokat a recordokat, amelyek cost attribútuma nem 'üres'!

Treatment_id	Patient_id	Cost
T001	P1500	55
T002	P9700	129
T003	P1500	
T004	P1500	57
T005	P9500	
T006	P4000	81
T007	P4000	82
T008	P8000	82

b) Opcionális: válasszátok ki az összes rekordot, de úgy, hogy a Cost mezőben hiányzó értékeket 0-val helyettesítitek! (Használd az nvl() függvényt)

c)Mit ad vissza a következő lekérdezés?

```
SELECT *
FROM   treatmentvar
WHERE  cost < 60
       OR cost >= 60;
```