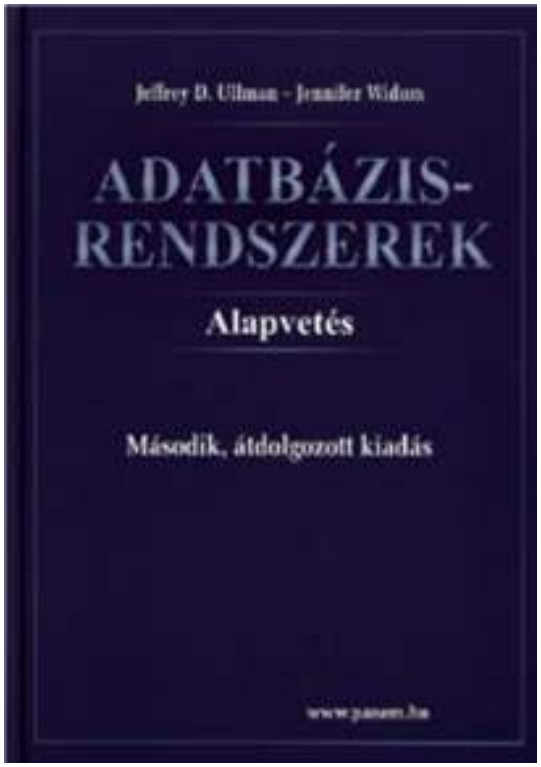
ADATBÁZISOK: TAN7.EA témaköre SQL DDL, DML, DCL, Tranz.kez.

(info. tanárszakon és fizikusoknak)

8.1.-8-2. folyt.SQL DDL. Nézettáblák

10.1. SQL DCL. Biztonság és felhasználói jogosultságok SQL-ben
GRANT és REVOKE

6.6. Tranzakciók az SQL-ben
(Gyakorlaton csak SAVEPOINT, COMMIT és ROLLBACK lesz.)



Relációs lekérdező nyelvek

Adatbázisok-1 kurzuson háromféle nyelvet tanulmányozunk:

- Relációs algebra: procedurális, algebrai megközelítés, megadjuk a kiértékelési tervet, többféle lehetőség összevetése, hatékonysági vizsgálatok.
- Datalog: deklaratív, logika alapú megközelítés, amely az összetett lekérdéseknél, például rekurziónál segítség.
- SQL szabvány relációs lekérdező nyelv: gyakorlatban, SQL története, szabványok, az SQL fő komponensei:
  SQL DDL (sémaleíró nyelv) milyen objektumok lehetnek?
  DML (adatkezelő és lekérdező nyelv) és
  Tranzakció-kezelés,
  DCL (vezérlő nyelv) milyen jogosultságok, privilégiumok?
  SQL-2003/PSM, ezt a gyakorlatban: PL/SQL (Oracle)

Áttekintés: A mai témáink

- Tankönyv 8.1.-8.2. fejezete a nézettáblákról, és az adatok módosításáról a nézettáblákon keresztül, tárolt nézettáblákról.
- Tankönyv 10.1.fejezete a jogosultságokról
- Tankönyv 5.7.fejezete a tranzakció-kezelésről

Mik a nézettáblák?

- A nézettábla olyan reláció, amit tárolt táblák (vagyis alaptáblák) és más nézettáblák felhasználásával definiálunk.
- **`EMPLOYEES` table**

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALA
100	Steven	King	SKING	515.123.4567	17-JUN-87	AD_PRES	240
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	170
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	170
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	90
104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-91	IT_PROG	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-99	IT_PROG	42
124	Kevin	Mourgos	KMOURGOS	650.123.5234	16-NOV-99	ST_MAN	58
141	Trenna	Rajs	TRAJS	650.121.8009	17-OCT-95	ST_CLERK	35
142	Curtis	Davies	CDAVIES	650.121.2994	29-JAN-97	ST_CLERK	31
143	Randall	Matos	RMATOS	650.121.2874	15-MAR-98	ST_CLERK	26
144	Peter	Vargas	PVARGAS	650.121.2004	09-JUL-98	ST_CLERK	25
149	Eleni	Zlotkey	EZLOTKEY	011.44.1344.429018	29-JAN-00	SA_MAN	105
174	Ellen	Abel	EABEL	011.44.1644.429267	11-MAY-96	SA_REP	110
176	Jonathon	Taylor	JTAYLOR	011.44.1644.429265	24-MAR-98	SA_REP	86
178	Kimberely	Grant	KGRANT	011.44.1644.429263	24-MAY-99	SA_REP	70
200	Jennifer	Whalen	JWHALEN	515.123.4444	17-SEP-87	AD_ASST	44
201	Michael	Hartstein	MHARTSTE	515.123.5555	17-FEB-96	MK_MAN	130
202	Pat	Fay	PFAY	603.123.6666	17-AUG-97	MK_REP	60
205	Shelley	Higgins	SHIGGINS	515.123.8080	07-JUN-94	AC_MGR	120
206	William	Gietz	WGIETZ	515.123.8181	07-JUN-94	AC_ACCOUNT	83

20 rows selected.

EMPLOYEE_ID	LAST_NAME	SALARY
149	Zlotkey	10500
174	Abel	11000
176	Taylor	8600

A nézettáblák előnyei

Korlátozható az adatok elérése

Bonyolult lekérdezések egyszerűbben kifejezhetők

Az adatfüggetlenség biztosítása

Ugyanazon adatokat különböző nézőpontokból szemlélhetjük

Virtuális vagy materializált?

- Kétféle nézettábla létezik:
  - Virtuális = nem tárolódik az adatbázisban, csak a relációt megadó lekérdezés.
  - Materializált = kiszámítódik, majd tárolásra kerül.

Nézettáblák létrehozása és törlése

- Létrehozása:

```
CREATE [OR REPLACE][FORCE|NOFORCE]
[MATERIALIZED] VIEW <név>
AS <lekérdezés>

[WITH CHECK OPTION [CONSTRAINT constraint]]
[WITH READ ONLY [CONSTRAINT constraint]];
```

- Alapesetben virtuális nézettábla jön létre.
- Nézettábla megszüntetése:

```
DROP VIEW <név>;
```

Példa: nézettábla létrehozása

- Mit_ihat(név, sör) nézettáblában a sörivók mellett azon söröket tároljuk, amelyeket legalább egy olyan söröző felszolgálnak, amelyet látogat:

```
CREATE VIEW Mit_ihat AS
    SELECT név, sör
    FROM Látogat, Felszolgál
    WHERE L.söröző = F.söröző;
```

Példa: nézettáblákhoz való hozzáférés

- A nézettáblák ugyanúgy kérdezhetők le, mint az alaptáblák.
  - A nézettáblákon keresztül az alaptáblák néhány esetben módosíthatóak is, ha a rendszer a módosításokat át tudja vezetni (lásd módosítások, SQL DML)
- Példa lekérdezés:

```
SELECT sör FROM Mit_ihat
WHERE név = 'Sally';
```

Módosítható nézettáblák

- Az SQL szabvány formálisan leírja, hogy mikor lehet egy nézettáblát módosítani és mikor nem, ezek a szabályok meglehetősen bonyolultak.
  - Ha a nézettábla definíciójában a SELECT után nem szerepel DISTINCT, további kikötések:
  - A WHERE záradékban R nem szerepelhez egy alkérdésben sem
  - A FROM záradékban csak R szerepelhet, az is csak egyszer és más reláció nem
  - A SELECT záradék listája olyan attribútumokat kell, hogy tartalmazzon, hogy az alaptáblát fel lehessen tölteni (vagyis kötelező a kulcsként vagy not null-nak deklarált oszlopok megadása)

Tankönyv példája: nézettáblára

Tk.8.1. Példa: Egy olyan nézettáblát szeretnénk, mely a Film(cím, év, hossz, színes, stúdióNév, producerAzon) reláció egy részét jelképezi, pontosabban a Paramount stúdió által gyártott filmek címét és gyártási évét

```
CREATE VIEW ParamountFilm AS
    SELECT cím, év
    FROM Film
    WHERE stúdióNév = 'Paramount';
```

Nézeteken instead-of-triggerek

Tk.8.8. Példa: Az előző nézettábla módosítható, és hogy az alaptáblába való beszúráskor a stúdióNév attribútum helyes értéke , 'Paramount' legyen, ezt biztosítja ez az INSTEAD OF (helyette) típusú trigger:

```
CREATE TRIGGER ParamountBeszúrás
    INSTEAD OF INSERT ON ParamountFilm
    REFERENCING NEW ROW AS ÚjSor
    FOR EACH ROW
    INSERT INTO Film(cím, év, stúdióNév)
    VALUES(Újsor.cím, ÚjSor.év, 'Paramount');
```

Materializált (tárolt) nézettáblák

- Adattárházaknál használják (MSc kurzusok)
- Probléma: minden alkalommal, amikor az alaptáblák valamelyike változik, a materializált nézettábla frissítése is szükségessé válhat.
  - Ez viszont néha túl költséges.
- Megoldás: Periodikus frissítése a materializált nézettábláknak, amelyek egyébként „nem aktuálisak”.

Példa nézetek használatára

- Képezzük osztályonként az összfizetést, vegyük ezen számok átlagát, és adjuk meg, hogy mely osztályokon nagyobb ennél az átlagnál az összfizetés.

```
CREATE OR REPLACE VIEW osztaly_osszfiz AS
SELECT onev, SUM(fizetes) ossz_fiz
FROM dolgozo d, osztaly o
WHERE d.oazon = o.oazon
GROUP BY onev;

CREATE OR REPLACE VIEW atlag_koltseg AS
SELECT SUM(ossz_fiz)/COUNT(*) atlag
FROM osztaly_osszfiz;

SELECT * FROM osztaly_osszfiz
WHERE ossz_fiz > (SELECT atlag FROM atlag_koltseg) ;
```

Példa nézetek helyett munkatáblák

- Ugyanez WITH átmeneti munkatáblával megadva:

```
WITH osztaly_osszfiz AS
( SELECT onev, SUM(fizetes) ossz_fiz
  FROM dolgozo d, osztaly o
  WHERE d.oazon = o.oazon
  GROUP BY onev),

        atlag_koltseg AS
( SELECT SUM(ossz_fiz)/COUNT(*) atlag
FROM osztaly_osszfiz)

SELECT * FROM osztaly_osszfiz
WHERE ossz_fiz > (SELECT atlag FROM atlag_koltseg);
```

Tk.10.1. Jogosultság-kezelés

- Egy UNIX-szerű fájlrendszerhez hasonlítva az analógiák: Tipikusan írás, olvasás és végrehajtási jogosultságokról van szó.
- Az adatbázisok lényegesen bonyolultabbak a fájlrendszereknél, ezért az SQL szabványban definiált jogosultágok is összetettebbek.
  - Az SQL kilencféle jogosultságot definiál (SELECT, INSERT, DELETE, UPDATE, REFERENCES, USAGE, TRIGGER, EXECUTE, UNDER)
  - Bizonyos „résztvevőkhöz” sorolja a jogosultságokat, például rendszergazda, korlátozott jogosultságokkal rendelkező felhasználó. Spec. PUBLIC (mindenki)

Jogosultságok

- A relációkra vonatkozó jogosultságok:

  SELECT = a reláció lekérdezésének joga.

  INSERT = sorok beszúrásának joga.

  (egyetlen attribútumra is vonatkozhat)

  DELETE = sorok törlésének joga.

  UPDATE = sorok módosításának a joga.

  (szintén egy attribútumra is vonatkozhat)

Példa: jogosultságok

- Az alábbi utasítás esetében:

```
INSERT INTO Sörök(név)
   SELECT sör FROM Felszolgál f
   WHERE NOT EXISTS
        (SELECT * FROM Sörök
         WHERE név = f.sör);
```

azok a sörök, amelyek még nincsenek benne a sörök táblában. A beszúrás után a gyártó értéke NULL.

- Az utasítás végrehajtásához szükséges: SELECT jogosultság a felszolgál és sörök táblába és INSERT jog a Sörök tábla név attribútumára vonatkozóan.

Adatbázis objektumok

- Jogosultságokat nézetekre és materializált nézetekre vonatkozóan is megadhatunk.
- Egy másik fajta jogosultság lehet pl. adatbázis objektumok létrehozásának a joga: pl. táblák, nézetek, triggerek.
- A nézettáblák segítségével tovább finomíthatjuk az adatokhoz való hozzáférést.

Példa: nézettáblák és jogosultságok

- Tegyük fel, hogy nem szeretnénk SELECT jogosultságot adni az Dolgozók(név, cím, fizetés) táblában.
- Viszont a BiztDolg nézettáblán már igen:

```
CREATE VIEW BiztDolg AS
    SELECT név, cím FROM Dolgozók;
```

- A BiztDolg nézettáblára vonatkozó kérdésekhez nem kell SELECT jog a Dolgozók táblán, csak a BiztDog nézettáblán.

Jogosultsági azonosítók

- A felhasználókat egy *jogosultsági azonosító* (*authorization ID*) alapján azonosítjuk, általában ez a bejelentkezési név.
- Külön jogosultsági azonosító a PUBLIC.
  - A PUBLIC jogosultság minden felhasználó számára biztosítja az adott jogot.

Jogosultságok megadása

- A magunk készítette objektumok esetében az összes jogosultsággal rendelkezünk.
- Másoknak is megadhatunk jogosultságokat, a PUBLIC jogosultsági azonosítót is használhatjuk.
- A WITH GRANT OPTION utasításrész lehetővé teszi, hogy aki megkapta a jogosultságot, tovább is adhassa azt.

A GRANT utasítás

- Jogosultságok megadásának szintaktikája:

  GRANT <jogosultságok listája>

  ON <reláció vagy másféle objektum>

  TO <jogosultsági azonosítók listája>;

- Ehhez hozzáadható:

  WITH GRANT OPTION

Példa: GRANT

```
GRANT SELECT, UPDATE(ár)
   ON Felszolgál
   TO Sally;
```

- Ez után Sally kérdéseket adhat meg a Felszolgál táblára vonatkozóan és módosíthatja az ár attribútumot.

Példa: Grant Option

```
GRANT UPDATE ON Felszolgál TO Sally
  WITH GRANT OPTION;
```

- Ez után Sally módosíthatja a Felszolgál táblát és tovább is adhatja ezt a jogosultságot.
  - Az UPDATE jogosultságot korlátozottan is továbbadhatja: `UPDATE(ár)ON Felszolgál`.

Jogosultságok visszavonása

REVOKE <jogosultságok listája>
ON <reláció vagy más objektum>
FROM <jogosultsági azonosítók listája>;

- Az általunk kiadott jogosultságok ez által visszavonódnak.
  - De ha máshonnan is megkapták ugyanazt a jogosultságot, akkor az még megmarad.

Tk. 6.6. Tranzakciók az SQL-ben
Miért van szükség tranzakciókra?

- Az adatbázis rendszereket általában több felhasználó és folyamat használja egyidőben.
  - Lekérdezések és módosítások egyaránt történhetnek.
- Az operációs rendszerektől eltérően, amelyek támogatják folyamatok interakcióját, az adatbázis rendszereknek el kell különíteniük a folyamatokat.

Példa: rossz interakció

- Egy időben ketten töltenek fel 100 dollárt ugyanarra a számlára ATM-en keresztül.
  - Az adatbázis rendszernek biztosítania kell, hogy egyik művelet se vesszen el.
- Ezzel szemben az operációs rendszerek megengedik, hogy egy dokumentumot ketten szerkesszenek egyidőben. Ha mind a ketten írnak, akkor az egyik változtatás elvész (elveszhet).

Tranzakciók

- Tranzakció = olyan folyamat, ami adatbázis lekérdezéseket, módosításokat tartalmaz.
- Az utasítások egy „értelmes egészt” alkotnak.
- Egyetlen utasítást tartalmaznak, vagy az SQL-ben explicit módon megadhatóak.

ACID tranzakciók

ACID tulajdonságok:

- Atomiság (atomicity): a tranzakció egységesen lefut vagy nem, vagy az összes vagy egy utasítás sem hajtódik végre.
- Konzisztencia (consistency): a tranzakció futása után konzisztens legyen az adatbázis, megszorításokkal, triggerekkel biztosítjuk.
- Elkülönítés (isolation): párhuzamos végrehajtás eredménye egymás utáni végrehajtással egyezzen meg
- Tartósság (durability): a befejezett tranzakció eredménye rendszerhiba esetén sem veszhet el
- Opcionálisan: gyengébb feltételek is megadhatóak.

COMMIT és ROLLBACK

- A COMMIT utasítás a tranzakció sikeres befejeződését eredményezi. Egy sikeresen befejeződött tranzakció a kezdete óta végrehajtott utasításainak módosításait tartósan rögzíti az adatbázisban
  - vagyis a módosítások *véglegesítődnek*.
- A ROLLBACK utasítás megszakítja a tranzakció végrehajtását, és annak sikertelen befejeződését eredményezi. Az így befejezett tranzakció SQL utasításai által végrehajtott módosításokat a rendszer meg nem történtekké teszi
  - Vagyis az összes utasítás *visszagörgetésre kerül*, a módosítások nem jelennek meg az adatbázisban.

Példa: egymásra ható folyamatok

- A Felszolgál(bár, sör, ár) táblánál tegyük fel, hogy Joe bárjában csak Bud és Miller sörök kaphatók 2.50 és 3.00 dollárért.
- Sally a Felszolgál táblából Joe legolcsóbb és legdrágább sörét kérdezi le.
- Joe viszont úgy dönt, hogy a Bud és Miller sörök helyett ezentúl Heinekent árul 3.50 dollárért.

Sally utasításai

(max)SELECT MAX(ár) FROM Felszolgál
WHERE bár = 'Joe bárja';

(min) SELECT MIN(ár) FROM Felszolgál
WHERE bár = 'Joe bárja';

Joe utasításai

- Ugyanabban a pillanatban Joe a következő utasításokat adja ki:

(del) DELETE FROM Felszolgál
WHERE bár = 'Joe bárja';

(ins) INSERT INTO Felszolgál
VALUES('Joe bárja', 'Heineken', 3.50);

Átfedésben álló utasítások

- A (max) utasításnak a (min) kell végrehajtódnia, hasonlóan (del) utasításnak az (ins) előtt, ettől eltekintve viszont nincsenek megszorítások a sorrendre vonatkozóan, ha Sally és Joe utasításait nem gyűjtjük egy-egy tranzakcióba.

Példa: egy furcsa átfedés

- Tételezzük fel a következő végrehajtási sorrendet: (max)(del)(ins)(min).

Joe árai:	{2.50,3.00}	{2.50,3.00}		{3.50}
Utasítás:	(max)	(del)	(ins)	(min)
Eredmény:	3.00			3.50

- Mit lát Sally? MAX < MIN!

A probléma megoldása tranzakciókkal

- Ha Sally utasításait, (max)(min), egy tranzakcióba gyűjtjük, akkor az előbbi inkonzisztencia nem történhet meg.
- Joe árait ekkor egy adott időpontban látja.
  - Vagy a változtatások előtt vagy utánuk, vagy közben, de a MAX és a MIN ugyanazokból az árakból számolódik.

Egy másik hibaforrás: a visszagörgetés

- Tegyük fel, hogy Joe a (del)(ins) és utasításokat nem, mint tranzakció hajtja végre, utána viszont úgy dönt, jobb ha visszagörgeti a módosításokat.
- Ha Sally az (ins) után, de visszagörgetés előtt hajtatja végre a tranzakciót, olyan értéket kap, 3.50, ami nincs is benne az adatbázisban végül.

Megoldás

- A (del)(ins) és utasításokat Joe-nak is, mint tranzakciót kell végrehajtatnia, így a változtatások akkor válnak láthatóvá, ha tranzakció egy COMMIT utasítást hajt végre.
  - Ha a tranzakció ehelyett visszagörgetődik, akkor a hatásai sohasem válnak láthatóvá.

Elkülönítési szintek

- Az SQL négy elkülönítési szintet definiál, amelyek megmondják, hogy milyen interakciók engedélyezettek az egy időben végrehajtódó tranzakciók közt.
- Ezek közül egy szint (“sorbarendezhető”) = ACID tranzakciók.
- Minden ab rendszer a saját tetszése szerint implementálhatja a tranzakciókat.

Az elkülönítési szint megválasztása

- Az utasítás:

SET TRANSACTION ISOLATION LEVEL *X*

ahol *X* =

1. SERIALIZABLE
2. REPEATABLE READ
3. READ COMMITTED
4. READ UNCOMMITTED

Sorbarendezhető (serializable) tranzakciók

- Ha Sally a (max)(min), Joe a (del)(ins) tranzakciót hajtatja végre, és Sally tranzakciója SERIALIZABLE elkülönítési szinten fut, akkor az adatbázist vagy Joe módosításai előtt vagy után látja, a (del) és (ins) közötti állapotban sohasem.

Az elkülönítési szint választása

- Ez a döntés csak azt mondja meg, hogy az illető hogyan látja az adatbázist, és nem azt, hogy mások hogy látják azt.
- Példa: Ha Joe sorbarendezhető elkülönítési szintet használ, de Sally nem, akkor lehet, hogy Sally nem talál árakat Joe bárja mellett.
  - azaz, mintha Sally Joe tranzakciójának közepén futtatná a sajátját.

Read-Commited tranzakciók

- Ha Sally READ COMMITTED elkülönítési szintet választ, akkor csak kommitálás utáni adatot láthat, de nem feltétlenül mindig ugyanazt az adatot.
- Példa: READ COMMITTED mellett megengedett a (max)(del)(ins)(min) átfedés amennyiben Joe kommitál.
  - Sally legnagyobb megdöbbenésére: MAX < MIN.

Repeatable-Read tranzakciók

- Hasonló a read-commited megszorításhoz. Itt, ha az adatot újra beolvassuk, akkor amit először láttunk, másodszor is látni fogjuk.
- De második és az azt követő beolvasások után akár *több* sort is láthatunk.

Példa: ismételhető olvasás

- Tegyük fel, hogy Sally REPEATABLE READ elkülönítési szintet választ, a végrehajtás sorrendje: (max)(del)(ins)(min).
  - (max) a 2.50 és 3.00 dollár árakat látja.
  - (min) látja a 3.50 dollárt, de 2.50 és 3.00 árakat is látja, mert egy korábbi olvasáskor (max) már látta azokat.

Kérdés/Válasz

- Köszönöm a figyelmet! Kérdés/Válasz?