Adatbázis rendszerek I tételsor

Kidolgozta: Huzinec Erik és Pozsgay Máté

Vizsga formátuma:

- Írásbeli (60 perc): rövid kérdések (8 db)
- kifejtős gyakorlati kérdések (2 db)

Követelmény: min 50%

Verzió: 2.5 (2012.01.13)

Kérdéskörök

Jel	Magyarázat
(*)	Csak annak kell tudnia, aki jelest szeretne
(!)	Jelenleg nincs kidolgozva
(-)	Gyakorlati feladat, nincs példával illusztrálva
(+)	Gyakorlati feladat példával

1 Információ és adat fogalma

Az információ olyan jelsorozatok által hordozott hír, mely egy rendszer számára új ismeretet jelent.

Adat a közlés formája, valamilyen rögzített információ. Az adat valamilyen jelrendszerben ábrázolt jelek, sorozatok összessége.

2 Információs rendszer fogalma és vetületei

Az információs rendszer egymással szoros összefüggésben lévő objektumok összessége, a közöttük fennálló kapcsolatokkal, ezek struktúrájával és dinamikájával.

Vetületei:

- Adatvetület
- Feldolgozás vetület
- Környezeti vetület

3 Alapvető fájlszervezési módszerek; szekvenciális, random, rekord, mező, kulcs

Fájlszervezési módszerek

- **Soros:** A rekordok között nincs kapcsolati elem
- **Láncolt:**
  - egy vagy több szintű
  - egy vagy két irányú
- **Indexelt:** egy külön struktúra tárolja a rekordok sorrendjét(kulcs, pozíció)
- **Indexszekvenciális:**
  - az állományban rendezetten helyezkednek el a rekordok
  - egy tartomány van fenntartva minden érték intervallumra
  - az index csak az intervallumokat tartalmazza
- **Hash:** A hash elérési módszer alapelve, hogy a kulcs értékéből valamilyen egyszerűbb eljárással, egy h() hash függvényt alkalmazva meghatároznak egy pozíciót

Fájlelérési módszerek

- **Szekvenciális:** Előnye, hogy nem szükséges a teljes fájlt végignézni adott kulcsértékű rekord keresésekor, mivel a sorrendbe rendezés miatt egy adott kulcstól jobbra csak a tőle nagyobb (növekvő rendezést feltételezve) kulcsértékű elemek helyezkedhetnek el
- **Random:** A rekordok közvetlenül elérhetőek, beolvashatóak anélkül, hogy az előtte lévő rekordokat is át kellene olvasni.
- **Rekord:** Adatbázis struktúra elem, mely a logikailag összetartozó, és egységként kezelhető elemi adatértékek együttesét jelöli. A mezősorrend rögzített (séma), köthetők hozzá integritási feltételek.
- **Mező:** Az adatbázis struktúra azon egysége, melyből a rekordok felépülnek; a mező rendszerint a legkisebb DB struktúra egység (egyértékű, atomi). A mezők megadásánál meg kell adni a domain-t (típust) és az integritási feltételeket.
- **Kulcs:** A keresés, az azonosítás szempontjából meghatározó mező vagy mezőcsoport, mely a rekord azonosítására szolgál; azaz értéke nem ismétlődik és egyetlen egy rekordban sem üres az értéke. Fontosabb típusai: elsődleges kulcs, jelölt kulcs, idegen kulcs, szuper kulcs, index kulcs.

4 Adatkezelési követelmények

- Nagy mennyiségű adatok hatékony kezelése (VLDB, időbeli hatékonyság)
- Konkurens hozzáférés támogatása: egyidejűleg több felhasználó, lost update léphet fel
- Integritásőrzés (adathelyesség)
- Védelem: adatvesztés, illetéktelen hozzáférés ellen
- Hatékony programfejlesztés: rugalmasnak kell lennie

5 Fájlszintű adatkezelés jellemzése

Fájlokban tárolódnak az adatok, ezeken műveletek végezhetők, leggyakoribb a query. Beolvasás: fejmozgatás, fejkiválasztás, forgatás. Az állományokon belül az adatok blokkokban tárolódnak. Több blokk együtt egy extendet alkot. A blokkok nyilvántartása láncolással és címlistával történik. A fájl belső logikai struktúrája stream vagy rekord jellegű.

6 Excel adatkezelési parancsok (jellemzés, rendezés, szűrés, csoportosítás)

A parancsok az "Adatok" menüpontból érhetők el.

- **Rendezés:** A rendezendő adatok tartományának kijelölése után az alábbi ablakban adható meg, hogy mely mezők (max. 3) szerint kívánjuk rendezni az adatokat és milyen sorrendben (növekvő ill. csökkenő).
- **Szűrés:**
  - **Auto szűrő:** Bekapcsolásával minden oszlop fejlécében (soros tárolás esetén a sor fejlécében) megjelenik egy nyíl, melyre kattintva a legördülő menüből választható ki a szűrés módja:
    - Növekvő/csökkenő rendezés az adott mező (oszlop) szerint
    - Összes megjelenítése (mind)
    - Egy adatsor kiválasztása a listából; helyezettek szűrése (Helyezés…)
    - Első/utolsó x tétel/százalék megjelenítés.
  - **Irányított szűrő:** Szűrés a tartomány megadásával.

7 Excel adatkezelés VB makróban, Form programozása, adattábla kezelése (!-)

A Microsoft Excel programban rendszeresen végzett feladatokat makróval automatikussá tehetjük. A makró olyan parancsok és függvények sorozata, melyeket Microsoft Visual Basic modulban tárolunk, és az adott feladat végrehajtásához bármikor futtathatunk.

8 Excel Form programok: adat felvitele, adat módosítás, lekérdezés

```
1  Dim tbl1 as Range
2  Dim t1 as String
3  t1 = "A4:d8"
4  Set tbl1 = Range(t1)
5
6  Dim mezo as Range
7  mezo = Range("J4").Value
8  MsgBox "mezo"
```

9 Excel Form programok: adatok integritás vizsgálata

```
1  Dim val as Integer
2  If val < 1 Then
3     MsgBox "Nem pozitív szám"
```

10 Információ vetületei (*)

- **Statisztikai:** a statisztikai oldal az információt hordozó jelek előfordulási gyakoriságait vizsgálja
- **Szintaktikai:** a szintaktikai oldal a jelsorozatok formális azonosságait vizsgálja
- **Szemantikai:** jelsorozat mögött húzódó jelentést, lényeget hangsúlyozza.
- **Pragmatikai:** a jelentés gyakorlati hasznosságát emeli ki
- **Apobetikai:** az információ mögött rejlő szándékot tartalmazza

11 DB fogalma, jellemzése

Az adatbázis egy olyan integrált adatszerkezet, mely több különböző objektum előfordulási adatait adatmodell szerint szervezetten, perzisztens módon tárolja olyan segédinformációkkal – metaadatokkal – együtt, melyek a hatékonyság, integritásőrzés, adatvédelem biztosítását szolgálják.

12 DBMS, DBS fogalma

- **DBMS:** Az adatbáziskezelő rendszer az a programrendszer, melynek feladata az adatbázishoz történő hozzáférések biztosítása és az adatbázis belső karbantartási funkcióinak végrehajtása.
- **DBS:** Az adatbázis rendszernek az adatbázis, az adatbáziskezelő rendszer, valamint az alkalmazói és segédprogramok összességét nevezzük.

13 Index fogalma és szerepe, működése

Az állomány rekordjainak kulcsértékét és a rekordpozíciót tároló szerkezet – melyben a bejegyzések kulcsérték szerinti sorrendben helyezkednek el – indexnek nevezzük. Célja a gyors kereshetőség lehetővé tétele.

14 B-fa jellemzése, keresés algoritmusa

A B-fák olyan kiegyensúlyozott keresőfák, amelyeket úgy terveztek, hogy hatékonyan lehessen alkalmazni őket mágneslemezeken, vagy más közvetlen hozzáférésű másodlagos tároló berendezéseken. A csúcsokban sok gyerekük lehet. Egy n csúcsú B-fa magassága O(lg n). Minden csúcsnak 3 eleme van: a tárolt kulcsok darabszáma, a kulcsok, és hogy levél-e. A csúcsokban meghatározott számú kulcs tárolható.

A B-fában történő kereséskor a gyökértől indulunk ki, és úgy haladunk lefelé a fában, hogy a keresett kulcsot összehasonlítjuk az érintett csúcsokban tárolt kulcsokkal. A keresési művelet végrehajtási ideje O(log(t)*n).

15 Elembeszúrás algoritmusa B-fánál

A B-fa bővítés algoritmusában a beszúrandó elem helyét a B-fa gyökeréből kiindulva keressük meg, kulcsát összehasonlítva az érintett csúcsokban tárolt kulcsokkal. Induláskor (üres B-fa) a gyökér csomópont üres, ide szúrjuk be az elemeket rendezett sorrendben. Ha egy csomópont megtelik, az új beszúrandó elemet "gondolatban" beillesztjük, az így kialakult sorból a középső elemet kiemeljük és feltesszük a szülő csomópontba. Ez a csomópont pedig ketté osztódik, az elemek két új csomópontba kerülnek szétosztásra egyenlő arányban oly módon, hogy a kiemelt elemtől balra lévő csomópontban lesznek a nála kisebb elemek, és a tőle jobbra lévő csomópontban pedig a nála nagyobb elemek. A szülő csomópont pointerei pedig aktualizálódnak.

16 Adatok törlése B-fánál (*)

Először megkeressük a törölni kívánt rekord kulcsát a fában, majd kitöröljük a rekordot az adatfájlból. Ha a Bfa tulajdonsága nem romlik, akkor készen vagyunk, különben két eset lehet:

1. Ha az N csúcs egyik szomszédos testvére több kulcsot és mutatót tartalmaz, mint amennyi minimum szükséges, akkor egy kulcsmutató párt áttehetünk az N csúcsba, miközben a kulcsok sorrendjét érintetlenül hagyjuk. Lehetséges, hogy az N csúcs szülőjében levő kulcsokat az új helyzethez kell igazítanunk. Ha például az N csúcs jobb oldali testvére M, és M biztosít egy fölösleges kulcsot és mutatót, akkor a legkisebb kulcsot áttesszük N-be. Az M és N szülőjében van egy olyan kulcs, amely az M csúcson keresztül elérhető legkisebb kulcsot jelöli; ilyenkor ezt föl kell emelni.
2. Ha egyik szomszédos testvért sem használjuk arra, hogy biztosítson egy fölösleges kulcsot az N számára, akkor van két olyan egymás melletti testvér, N és M, amelyik közül az egyik a minimum szükséges számú kulccsal rendelkezik, a másik eggyel kevesebbel. Így ezt a kettőt összeolvasztjuk úgy, hogy az egyiket töröljük. A szülőben levő kulcsokat az új helyzethez kell igazítanunk, és ezután kell törölnünk egy kulcsot a szülőből. Ha a szülő így is eléggé telítve van, akkor készen vagyunk, ha nem, akkor rekurzívan alkalmazzuk a szülőre a törlési algoritmust.

17 Hashing algoritmusok jellemzése, összevetés az indexszel

A hash elérési módszerek a rekord pozícióját közvetlenül a rekord kulcsértékéből határozzák meg, tehát csak egy blokkolvasásra van szükség a rekord eléréséhez. Sajnos azonban nem mindig igaz, hogy egy blokkolvasás elegendő, ez csak ideális esetben valósul meg. A hash elérési módszer alapelve, hogy a kulcs értékéből valamilyen egyszerűbb eljárással, egy h() hash függvényt alkalmazva meghatároznak egy pozíciót. Rendszerint nem alkalmas intervallum keresésre.

A jó hash függvény egyenletesen teríti a rekordokat a hash táblában, pl. h(x)=x mod M

Tulcsordulási problémák lehetséges kezelése:

- túlcsordulási bucket láncolása
- hash tábla és hash függvény átalakítás

18 Virtuális hash működése (*)

Többszintű hash réteg, melynek előnye, hogy minimális lesz az átrendezés költése. A bejegyzések csak pointerek a fizikai blokkokra. Ami eddig egy helyen volt és nem fér el azt átteszi 2 hash táblába úgy, hogy az elsőt az eredeti sorszámú helyre a másik részét meg annyival eltolva a következőbe amennyi a hash mérete volt.

19 DBMS belső struktúrája,komponensei

Hálózati, kapcsolati réteg

Szintaktikai elemzés
Szemantikai ellenőrzés
Védelem
optimalizálás
tranzakció kezelés
Végrehajtó I/O

20 Tranzakció kezelő modul feladatai (*)

A vezérlő utasításokhoz szokás sorolni a műveletvégrehajtást szabályozó, úgynevezett tranzakció kezelő utasításokat is. Az SQL szabványban két, a tranzakció végét jelző utasítást szokás definiálni:

- **COMMIT:** A tranzakció sikeres befejezésének utasítása. Ennek hatására a korábban végrehajtott tevékenységek véglegesítődnek, megőrződnek az adatbázisban.
- **ROLLBACK:** A korábbi tevékenységek érvénytelenítődnek, mintha ki sem adtuk volna őket. Ezzel, valamilyen hibával félbeszakadt műveletsort lehet törölni, visszavonni.

21 Adatbázis létrehozás lépései

1. Követelmény analízis, feladat specifikáció
2. Adatbázis-kezelő rendszer (DBMS) kiválasztása
3. Adatmodellezés:
   - Szemantikai adatmodell elkészítése
   - A szemantikai adatmodell konverziója a megfelelő adatbázis adatmodellre
   - Normalizálás
   - Az adatbázisban tárolt adatok típusának, integritási feltételeinek meghatározása
4. Kódolás, implementálás

22 ANSI SPARC modell (*)

Külső szint	felhasználó forgalom köre (SDM)
Logikai szint (adatbázis modellek)	informatika, tervező (Relációs)
Fizikai szint	implementáció (DBMS belső)

Függetlenségi szintek:

- **Logikai:** a külső szint és a globális szint között
- **Fizikai:** a globális szint és a fizikai szint között
- **Hálózati:** fizikai szinten belül

23 Adatmodell fogalma,komponensei

Adatmodell: Azon formalizmus, melynek segítségével megadhatók a vizsgált problémakör adateleminek szerkezete, viselkedése.

Elemei:

- Struktúra leíró: építő elemek definiálás
- Művelet leíró: elvégezhető operációk
- Integritási feltételek: statikus vagy aktív szabályok

24 SE piramis (*)

SE piramis a következő lépéseket foglalja magába:

- **Követelmény analízis:** A vizsgált problématerület elemzése, a megoldandó feladatok, a követelmények meghatározása
- **Rendszerelemzés:** A problématerület modellezése, a belső struktúrák és működés feltárása több különböző szemszögből és különböző részletességgel
- **Rendszertervezés:** Az elkészítendő szoftver belső struktúrájának, működésének több különböző szemszögből történő feltárása, különböző részletesség mellett

- **Kódolás:** Az elkészült modell leírás átkonvertálása a számítógép által érthető formára (valamely programozási nyelvet felhasználva)
- **Tesztelés:** Az elkészült kód hibamentességének ellenőrzése
- **Karbantartás:** Folyamatos ellenőrzés, módosítások, hibakijavítások sorozata

25 Adatmodellek típusai

- **Szemantikai:** ER, EER, IFO, UML
- **Adatbázis adatmodell:** Hierarchikus, hálós, relációs (objektumrelációs, objektumorientált, XML-alapú …)

26 Szemantikai adatmodellek jellemzői

ER:

- Normál – gyenge egyed közötti különbség
- Összetett kulcs (több egyértékű tulajdonság együtt azonosítja az egyedet)
- Kapcsolathoz rendelhetők tulajdonságok
- A kapcsolatot lehet egyedként is ábrázolni
- Nem érdemes zárt modellt készíteni

EER: Extended ER:
Az EER modell két új elemet tartalmaz az ER modellel összevetve, mindkettő az egyedek közötti kapcsolatokra vonatkozik. Az egyik új elem a tartalmazási (HAS_A) kapcsolat, a másik pedig a specializációs kapcsolat. (IS_A)

IFO (funkcionális adatmodell):

- **Objektumok:** Elemi vagy összetett: absztrakt vagy származtatott
- **Kapcsolat:** Asszociáció (hozzárendelés), specializáció, általánosítás
- **Összetett struktúrák:** Aggregáció, csoportképzés

UML

- Osztály, attribútumok, metódusok
- Kapcsolat: asszociáció (számosság jelölésével) általánosítás
- Aggregáció (komponensek), kompozíció (szorosabb strukturális kapcsolat)

27 Az ER modell elemei és szerepe

- **Egyed:** Egy objektum típus, egy a külvilág többi részétől egyértelműen megkülönböztetett, önálló léttel bíró dolog, amiről az információkat tárolni kívánjuk. Jelölése: téglalap
- **Tulajdonság:** Az egyedeket, kapcsolatokat jellemző mennyiség, a letárolandó információelemeket tartalmazza. Jelölése: ellipszis
- **Kapcsolat:** Az egyedek között fennálló ideiglenes vagy tartós asszociáció, ahol csak az elsődleges kapcsolatokat adjuk meg. Jelölése: rombusz

28 ER egyedtípusok

- Normál egyed (önmagában azonosítható) például: dolgozó, autó
- Gyenge egyed (más egyedhez való kapcsolatán keresztül azonosított) például: dolgozó felesége, autó motorja

29 ER tulajdonságtípusok

- **Normál:** Egyértékű
- **Kulcs:** Azonosító szerepű (rendszám)
- **Összetett:** Több tagból áll (lakcím)
- **Többértékű:** Több értéke is lehet (e-mail cím)
- **Származtatott:** Értéke kiszámítható (életkor)

30 ER kapcsolattípusok

Kötelező jelleg szerinti kapcsolattípusok:

- **Opcionális:** Létezhet olyan egyedelőfordulás, melyhez nem kapcsolódik egyedelőfordulás a kapcsolatban. Jelölése: egyvonalas nyíl. (Nem minden könyvet kölcsönöznek ki a könyvtárból)
- **Kötelező:** Minden egyedelőforduláshoz kell kapcsolódnia egyedelőfordulásnak a kapcsolatban. Jelölése: kétvonalas nyíl. (Minden könyvnek van kiadója)

Számosság szerinti kapcsolattípusok:

- **1:1 (egy – egy)** Egy egyedelőforduláshoz maximum egy egyed társul a kapcsolatban, mindkét viszonylatban.
- **1:N (egy – több):** Egy egyedelőforduláshoz több egyed társulhat, de a másik irányban csak egy kapcsolódó egyedelőfordulás létezhet.
- **N:M (több – több):** Mindkét irányban több kapcsolódó előfordulás létezhet.

31 ER modell fejlesztés alapelvei (!)

32 ER modell korlátai (*)

A rugalmasság ellenére számos esetben nem lehet egzaktul megoldani az adatrendszer leírását. Problémát jelent a specializációk, általánosítások, tartalmazási relációk ábrázolása, mivel az ER csak az asszociációt ismeri. Nem lehet továbbá integritási szabályokat megadni benne.

33 EER modell jellemzése, új elemei

Az ER modell kibővítése a specializáció és a tartalmazás kapcsolat elemekkel. Ezek jelölése: a nyílra rá kell írni a kapcsolat jellegét (IS_A, HAS_A).

Vegyük az alábbi kapcsolatokat:

- Az autónak van tulajdonosa: hozzárendelés, ideiglenes, szimmetrikus, laza kapcsolat
- Az autónak van motorja: tartalmazás jellegű, állandósult, nem szimmetrikus kapcsolat
- Az autó a járművek csoportjához tartozik: specializáció, állandósult, nem szimmetrikus kapcsolat

34 N-es kapcsolatok ábrázolása ER-ben (*)

N-ed fokú kapcsolat: n egyed vesz részt a kapcsolatban. Ábrázolásában a binér kapcsolathoz képest az a különbség, hogy a rombuszból több nyíl fut ki.

35 ER modell elkészítése fogalmi leírásból (-)

gyakorlati

36 Hálós adatmodell jellemzése

A hálós adatmodell legfontosabb jellemzői a következő pontokban foglalhatók össze:

- Hálós kapcsolati struktúra
- A háló SET-ekbol épül fel
- Gazdag kapcsolati viszony
- Hatékony kapcsolattartás
- Összetett mezőszerkezetek támogatása
- Pointer alapú kapcsolattartás
- A kialakult struktúra viszonylag körülményesen módosítható
- A lekérdezés beágyazott környezetben, algoritmikus elemekkel történik
- Támogatja az összetett integritási feltételeket

37 Hálós struktúra (mező, rekord, set, pcr) jellemzése

- MEZŐ jellemzője:
  - lehet összetett
    - vektor: többértékű struktúra
    - csoport: összetett egy vagy többértékű struktúra
  - lehet normál vagy kulcs
  - elnevezés, típus jellemzi
- REKORD
  - rögzített mezősorrend
  - elnevezés, szerkezet jellemzi
- SET jellemzője:
  - egyszintű fa struktúra
  - az azonos rekordból kiinduló PCR elemeket fogja össze
  - a gyökér rekordtípus a SET tulajdonosa

    - a gyermek rekordtípusok a tagok
    - egy rekordtípus több SET-ben is szerepelhet
    - van azonosító neve
- PCR kapcsolat jellemzője:
    - egy szülő és egy gyerek rekord alkotja
    - a szülő rekord minden előfordulásához több gyerek rekord előfordulás tartozhat
    - egy gyerek előforduláshoz egy szülő rekord előfordulás tartozik
    - nincs külön elnevezés

38 Kapcsolattartás megvalósítási szintjei (*) (!)

39 ER modell leképzése hálós modellre

Elemi tulajdonság	→	Mező
Kulcs tulajdonság	→	Kulcs mező
Egyed	→	Rekord
Összetett tulajdonság	→	Összetett mező
Többértékű tulajdonság	→	Többértékű mező
1:1 kapcsolat	→	PCR
1:N kapcsolat	→	PCR

40 N:M kapcsolat megvalósítása hálós modellben

N:M kapcsolat → kapcsoló rekord + 2 PCR

41 EER modell leképzése hálósra (*)

EER Is_A kapcsolat

a) Csak a specializált egyed konvertálása rekorddá, amely tartalmazza az absztrakt egyed tulajdonságait is
b) Az absztrakt és a specializált egyedet egyaránt konvertáljuk rekorddá, közöttük PCR kapcsolat, ahol a szülő az absztrakt egyedet megvalósító rekord lesz

42 PCR fizikai megvalósítása a hálós modellben

A rekordelőfordulások közötti kapcsolatok nem fizikai tárolási pozícióval, hanem pointerekkel kerülnek letárolásra. Ennek megfelelően pointer láncok alakulnak ki köztük. Mivel egy tetszőleges rekordtípus több SET -ben is lehet tagrekord vagy tulajdonos, pointer háló jöhet létre. Ezek a láncok és hálók adják a gyors és hatékony elérés alapját, mivel mindig rögzített számú pointernek kell helyet foglalni.

43 Szinguláris SET fogalma, megvalósítása

A szinguláris SET olyan SET, amelyben nem létezik explicit tulajdonos, hanem a rendszert tekintik implicit tulajdonosnak, és rendszerint csak egy tagrekordja van. E SET célja, hogy egy helyen érjük el a rekordtípus összes előfordulását mindennemű kapcsolatok figyelembe vétele nélkül.

44 Séma és példány fogalmai, kapcsolata

Az adatbázis sémával azonosíthatjuk az adatbázisainkat. Tetszőleges számú SET -típust lehet benne definiálni. A SET -előfordulás konkrét kapcsolódó rekordelőfordulásokat tartalmaz. Minden tulajdonos rekordelőforduláshoz pontosan egy SET -előfordulás rendelődik, melyben a tagrekordoknak tetszőleges számú előfordulása szerepel a SET sémában meghatározott sorrendben.

45 CODASYL szabályok (*)

- Az adatbázis tetszőleges számú SET -típust tartalmazhat
- Minden SET -nek van neve és egy tulajdonosa
- Minden SET -ben van egy vagy több tag rekordtípus
- Minden SET -hez tartozik egy tagrekord tárolási sorrend
- Bármely egyedtípus megadható egy vagy több SET tagjaként
- Bármely egyedtípus csak egy SET -ben lehet tulajdonos
- Egy tulajdonos rekord előfordulás létrehoz egy SET -előfordulást

- Egy SET -ben egy rekordtípus bármely előfordulása legfeljebb csak egyszer szerepelhet
- Egy SET -előfordulásban a tagrekordnak tetszőleges sok előfordulása szerepelhet

46 DBMS kezelő API logikai struktúrája hálós modellnél (!)

47 hálós modell műveleti algebra elemei, navigációs modell jellemzése

Az utasítások gazdanyelvi környezetben működnek. Léteznek kapcsoló változók, amik kapcsolatot képeznek a DBMS és az API között. A DB -ben rekord pointerek jelzik, hogy mely rekordok állnak feldolgozás alatt.

Ezek szintjei:

- Rekord szintű: megadja, hogy mely rekord-előfordulást érintették utoljára a rekordtípus rekordjaiból
- SET szintű: megadja, hogy mely rekord előfordulást érintették utoljára a SET típus rekordjaiból
- DB szintű: megadja, hogy mely rekord-előfordulást érintették utoljára az adatbázis összes rekordja közül.

A lekérdezés navigációs jellegű.

48 A p, m, o műveletek működés, paraméterezése

$p^{1}_{feltétel}$ (rekord): a megadott rekordtípuson belül az első olyan rekord előfordulásra áll rá, mely teljesíti a paraméterként megadott feltételt.

$p^{n}_{feltétel}$ (rekord): a megadott rekordtípuson belül a következő olyan rekord előfordulásra áll rá, mely teljesíti a paraméterként megadott feltételt.

o (SET, rekord): a megadott SET -típuson belül megkeresi a megadott rekordtípus kijelölt előfordulását, majd elmegy ezen előforduláshoz tartozó tulajdonos rekord előforduláshoz.

$m^{1}_{feltétel}$ (SET, rekord): a megadott rekordtípuson belül az első olyan rekord előfordulásra áll rá, mely teljesíti a paraméterként megadott feltételt és benne van a SET tagrekord előfordulásai között,valamint gyereke a SET tulajdonos rekord előfordulásának is.

$M^{n}_{feltétel}$ (SET, rekord): a megadott rekordtípuson belül a következő olyan rekord előfordulásra áll rá, mely teljesíti a paraméterként megadott feltételt és benne van a SET tagrekord előfordulásai között, valamint gyereke a SET tulajdonos rekord előfordulásának is.

49 Keresési műveletek hálós API-ban

- FIND FIRST | NEXT rekordtípus USING feltétel:
  Keresés egy rekordtípus rekord előfordulásai között a minta szelekciós feltétel alapján. A FIRST kulcsszó az első előfordulást, NEXT a következő rekord előfordulást adja vissza. A parancs a p1 feltétel(rekord) ill. pn feltétel(rekord) műveleteket valósítja meg.
- FIND FIRST | NEXT rekordtípus IN CURRENT SETnév SET USING feltétel:
  Keresés egy rekordtípus rekord előfordulásaira a megadott SET -előfordulás tagrekordjai között. A FIRST kulcsszó az első rekord előfordulást, a NEXT a következő rekord előfordulást adja vissza. A parancs az m^{1} feltétel(SET, rekord) és m^{n} feltétel(SET, rekord) műveletet valósítja meg. Az a SET -előfordulás fog kiválasztódni, amely a pointerrel kijelölt tulajdonos rekord előforduláshoz tartozik. A feltétellel szűkíthető az érintett rekordok köre.
- FIND OWNER OF rekordtípus IN CURRENT SETnév SET:
  Keresés a megadott típusú rekord előfordulás tulajdonos előfordulását adja vissza a megadott SET -hez tartozóan. A parancs az o(SET, rekord) műveletet valósítja meg.

50 módosítás végrehajtása hálós API-ban

DML utasítások végrehajtási menete a következő:

1. navigálással ráállunk arra a rekord előfordulásra, melyet módosítani kell;
2. beállítjuk a kapcsoló változó értékét a kívánt új értékre;
3. meghívjuk a megfelelő DML utasítást.

- **STORE rekordtípus:** új rekord felvitele.
- **ERASE rekordtípus:** rekord törlése.
- **MODIFY rekordtípus:** rekord módosítása.

51 hálós modell hátrányai

Hálós adatmodell korlátai:

- Merev struktúra
- komplex, erőforrás igényes
- Bonyolult, algoritmussal leírandó lekérdezés, adatkezelés
- Az N:M kapcsolat közvetlenül nem adható meg, át kell konvertálni 1:N –re. Az igényelt pointerek száma nem csökken, csak más struktúrában lesz. Létrejön egy kapcsolati egyed és 2 db 1:N.

52 Relációs modell főbb eltérései a hálós modellel összevetve

- A relációs adatmodell nem épít be a modellbe fix, rögzített kapcsolattípusokat, mint a PCR vagy a VPCR volt.
- A relációs adatábrázolás nem használja a pointeres hivatkozásokat, ezért nem alkothatunk összetett vagy többértékű mezőket, mint ahogy azt a hálós modellben megtehettük.
- A rekordok közötti kapcsolatok ábrázolása a rekordok értékegyezőségén alapszik.
- Az adatok lekérdezésekor minden reláció egyenértékű, nincs alá- vagy fölérendelt rekord.
- N:M kapcsolatnál kapcsoló táblát kell létrehozni.

53 Reláció struktúra (mező, rekord reláció, kulcs)

- **Mező:** Az adatbázis struktúra azon egysége, melyből a rekordok felépülnek; a mező rendszerint a legkisebb DB struktúra egység (egyértékű, atomi). A mezők megadásánál meg kell adni a domain-t (típust) és az integritási feltételeket.
- **Rekord:** Adatbázisstruktúra elem, mely a logikailag összetartozó, és egységként kezelhető elemi adatértékek együttesét jelöli. A mezősorrend rögzített (séma), köthetők hozzá integritási feltételek.
- **Rekordkulcs:** A rekord előfordulást azonosító mezőcsoport, a kulcs értéke nem lehet azonos két különböző rekordban.
- **Kulcs:** A keresés, az azonosítás szempontjából meghatározó mező vagy mezőcsoport, mely a rekord azonosítására szolgál azaz értéke nem ismétlődik és egyetlen egy rekordban sem üres az értéke.
  Fontosabb típusai: elsődleges kulcs, jelölt kulcs, idegen kulcs, szuper kulcs, index kulcs.
- **Kulcs tulajdonság:** Olyan tulajdonság vagy tulajdonság csoport, melynek értéke egyértelműen meghatározza az egyed előfordulását.
- **Index:** Az állomány rekordjainak kulcsértékét és a rekord pozíciót tároló szerkezet, melyben a bejegyzések kulcsérték szerinti sorrendben helyezkednek el, gyors keresést lehetővé téve.
- **Reláció:** Az azonos szerkezetű rekordelőfordulások névvel ellátott halmaza, tárolási egység a relációs adatbázisban.

54 Kapcsolattartás jellemzése (idegen kulcs)

- **Idegen kulcsmező:** Olyan mezőcsoport a relációsémában, melynek célja egy megadott másik reláció valamely rekord előfordulásának az egyértelmű kijelölése.
- **Kapcsoló kulcs:** Az idegen kulcs egy másik elnevezése, utalva arra, hogy az idegen kulcs kapcsoló szerepet tölt be.

55 Reláció halmazorientáltságának jelei

- A relációban ugyanaz a rekord előfordulás nem fordulhat elő többször;
- Egy halmaz egy elemét ugyanis csak egyszer tartalmazhatja, és ha két rekord minden mezője megegyezik egymással, akkor a két rekord ugyanazt az előfordulást jelenti, tehát csak egyszer fordulhat elő;
- Minden mező atomi értéket tartalmaz;
- A reláció elemei között semmiféle rendezettség, sorrend nem értelmezett (gyakorlatban nem mindig teljesül);
- A mezők között semmiféle rendezettség, sorrend nem értelmezett (gyakorlatban nem mindig teljesül).

56 Reláció formális felírása (*)

$\mathbf{U}$	univerzum
$A \in \mathbf{A} \subseteq \mathbf{U}$	attribútumok
$\mathbf{V} \subseteq \mathbf{U}$	értékek halmaza
$D \in 2^{\mathbf{V}}$	domain
$dom : \mathbf{A} \Rightarrow 2^{\mathbf{V}}$	attribútumok hozzárendelése domain-ekhez
$\mathbf{R} \subseteq \mathbf{U}$	reláció séma

r(**R**)	reláció R felett
t	egy tuple, ahol r(**R**) = { $t : \mathbf{R} \Rightarrow \mathbf{V} \mid \forall A \in \mathbf{R} : t(A) \in dom(A)$}
R = {**R**}	reláció sémák halmaza
REL(**R**) = {r | r(**R**)}	R feletti relációk
B = {b}	lokális integritási feltételek, ahol $b : \mathbf{REL}(\mathbf{R}) \Rightarrow (0,1)$
R = (**R**, **B**)	kiterjesztett relációs séma és reláció r(***R***) = {r | r (**R**) $\wedge$ $\forall b \in \mathbf{B}$: b(r) = 1 }
D $\subseteq$ {**R**}	adatbázis séma
d(**D**) = {r(**R**) | **R** $\in$ **D**}	adatbázis
DAT(**D**) = {d | d(**D**)}	D feletti adatbázisok
B' = {b'}	globális integritási feltételek, ahol $b' : \mathbf{DAT}(\mathbf{D}) \Rightarrow (0,1)$

- **Egyed integritási szabály**: Minden relációs sémában létezzen kulcs (a kulcs nem üres, egyedi és azonosító)
  $b_K(r(\mathbf{R})) = 1$, ha $K \subseteq \mathbf{R} \wedge \forall t_1,t_2 \in r(\mathbf{R}): t_1 \neq t_2 \Rightarrow t_1(K) \neq t_2(K)$
  0, különben
- **Hivatkozási integritási szabály**: Az idegen kulcs vagy üres vagy létező kulcs értékre mutat
  $b'_{X,Y}(r1(\mathbf{R1}),r2(\mathbf{R2})) =$ 1, ha $X \subseteq \mathbf{R1}$, $Y \subseteq \mathbf{R2} \wedge b_Y(r(\mathbf{R2})) \wedge \{t(X) \mid t \in r(\mathbf{R1})\} \subseteq \{t(Y) \mid t \in r(\mathbf{R2})\}$
  0, különben
- **Reláció elnevezése**: $r(\mathbf{R}) \subseteq dom(A_1) \times dom(A_2) \times \ldots \times dom(A_n)$ ahol $\mathbf{R} = \{A_1,A_2,\ldots,A_n\}$ a mezők értéke lehet kitöltetlen, ennek jele: NULL $\in dom(A_i)$

57 ER modell konverziója relációsra

ER modell konverziója relációs modellre

- egyed → reláció
- egyértékű tulajdonság → mező
- összetett tulajdonság → több mező a tagokhoz
- többértékű tulajdonság → új leíró reláció
- 1:1 kapcsolat → idegen kulcs
- 1:N kapcsolat → idegen kulcs
- N:M kapcsolat → új kapcsoló reláció

58 Kapcsolat konvertálása relációsra

- 1:1 kapcsolat → idegen kulcs
- 1:N kapcsolat → idegen kulcs
- N:M kapcsolat → új kapcsolóreláció

59 Tulajdonságok konvertálása relációsra

- egyértékű tulajdonság → mező
- összetett tulajdonság → több mező a tagokhoz
- többértékű tulajdonság → új leíró reláció

60 Egyedintegritási szabály és hivatkozás épség szabályai (*)

- **Egyed-integritás (entity-integrity):** Minden relációnál legyen kulcs mezőcsoport és a kulcs mezőcsoport nem lehet még részlegesen sem üres, azaz NULL érték. A részlegesen üres kulcs akkor lehetséges, ha a kulcs nem elemi, hanem több mező együtteséből áll. Azaz minden alaptáblában tárolt egyednek legyen egyedi azonosító kulcsa, mely egyetlen egy esetben sem lehet üres.
- **Hivatkozási integritás (referential integrity):** Léteznie kell olyan rekordnak a másik relációban, melynek kulcsértéke megegyezik a kapcsoló kulcs értékével. A kapcsoló kulcs értéke tehát vagy üres, vagy a hivatkozott tábla kulcsmezői között szerepel, azaz a kapcsoló kulcsnak létező egyedre kell mutatnia.

61 Integritási elemek a relációs modellnél, szabályok jellemzése

- **Domain (Mező) szintű:** Egy mezőre vonatkozó érték előfordulások körét lehet megadni (pl. rendszám első három karaktere nem lehet szám)
  - CHECK feltétel: Értékellenőrzés
  - NOT NULL: Kötelező kitölteni, nem maradhat üres
- **Rekord szintű:** Egy teljes rekord elfogadhatóságát döntjük el, célja az egy rekordon belül egymáshoz kapcsolódó mezők értékeinek vizsgálata.
  - CHECK feltétel: Több mezőt érintő értékellenőrzés
- **Reláció szintű:** A teljes relációt, azaz több rekordelőfordulást is át kell vizsgálni (pl. egy mezőérték csak egyszer fordulhat elő)
  - PRIMARY KEY: Elsődleges kulcs
  - UNIQUE Egyediség
- **Adatbázis szintű:** A feltétel több relációban szétszórtan elhelyezkedő mezőkre vonatkozik (pl az autó típuskódjának szerepelnie kell a típusok reláció valamely rekordjának kód mezőjében)
  - FOREIGN KEY Idegen kulcs
  - ASSERTION feltétel Összetett, több tábla mezőjét érintő értékellenőrzés

62 EER modell konverziója relációs modellre (*)

Nincs egyértelmű megoldás az IS_A kapcsolat relációssá konvertálásában, ezért sima kapcsolatként konvertáljuk a speciális elemeket.

63 Integritási szabályok formális felírása (*)

- **Egyed integritási szabály**: minden relációs sémában létezzen kulcs (a kulcs nem üres, egyedi és azonosító)
  $b_K(r(\mathbf{R})) =$ 1, ha $K \subseteq \mathbf{R} \wedge \forall t_1,t_2 \in r(\mathbf{R}): t_1 \neq t_2 \Rightarrow t_1(K) \neq t_2(K)$
  0, különben
- **Hivatkozási integritási szabály**: az idegen kulcs vagy üres vagy létező kulcs értékre mutat
  $b'_{X,Y}(r1(\mathbf{R1}),r2(\mathbf{R2})) =$ 1, ha $X \subseteq \mathbf{R1}$, $Y \subseteq \mathbf{R2} \wedge b_Y(r(\mathbf{R2})) \wedge \{t(X)|t \in r(\mathbf{R1})\} \subseteq \{t(Y)|t \in r(\mathbf{R2})\}$
  0, különben

64 Relációs algebra jellemzése

Relációs algebra: az operandusok és az eredmények relációk, azaz a relációs algebra műveletei zártak a relációk halmazára.

Műveletei:

- Ismert halmazműveletek (Unió, különbség, metszet)
- Hányados: A minden kifejezésére
- Szelekció (s): Sorok kiválasztása
- Projekció (p): Oszlopok kiválasztása
- Aggregáció: Sum, Count, avg
- Összekapcsolások: INNER, OUTER LEFT/RIGHT/FULL, NATURAL, és Descartes-szorzat
- Csoportképzés
- Kiterjesztés

65 Szelekció, projekció jellemzése

A szelekciós művelet a relációban szereplő rekord előfordulások egy részhalmazának az előállítására szolgál. A művelet értelmezése: a szelekció eredményhalmazába csak azok a rekord előfordulások kerülnek bele, melyek kielégítik a megadott szelekciós feltételt. A szelekció a tábla bizonyos sorait adja eredményként. Jele: σ (szigma) feltétel (reláció)

A projekció azt a műveletsort jelenti, amikor a relációban a rekord előfordulásokat leíró mezőkből csak bizonyos mezők értékeit kérdezzük le eredményként. Az eredményreláció csak a kijelölt mezőkre vonatkozó adatokat tartalmazza, viszont minden rekord előfordulás szerepel az eredményrelációban. A projekció műveletének értelmezése: a projekció eredmény halmazába csak a megadott mezőértékek kerülnek át az alapreláció minden egyes rekord előfordulásából. Jele: Π (pí) mezőlista (reláció)

66 Join típusainak jellemzése

- Alap join (Descartes-join, minden lehetséges rekordpárost előállít):
  reláció1 ►◄ reláció2
- Szelekciós join (csak a feltételt kielégítő rekordpárosokat adja vissza):
  reláció1 ►◄ feltétel reláció2
- Natural join (az azonos elnevezésű mezők értékegyezőségén alapszik):
  reláció1 ►◄ = reláció2
- Inner join (a feltételnek megfelelően csak az illeszkedő rekordpárosok):
  reláció1 ►◄ feltétel reláció2
- Szemi-join (a feltételt kielégítő rekordpárosokból csak az egyik oldal jelenik meg):
  - reláció1 ► feltétel reláció2 (csak a join bal oldalán szereplő relációból kapjuk vissza az illeszkedő rekordokat)
  - reláció1 ◄ feltétel reláció2 (csak a join jobb oldalán szereplő relációból kapjuk vissza az illeszkedő rekordokat)
- Outer join (a feltétel szerint illeszkedő rekordpárosok + a pár nélküli rekordok a megadott oldalról)
  - Right outer join: reláció1 ►◄ + reláció2
  - Left outer join: reláció1 + ►◄ reláció2
  - Full outer join: reláció1 + ►◄ + reláció2

67 Csoportképzés, aggregáció jellemzése

Bizonyos esetekben nem magára a konkrét rekord előfordulásra vagyunk kíváncsiak, hanem a rekord előfordulások valamilyen összesítő adataira. A relációs műveletek egységessége értelmében az ilyen kérdésekre adott válaszoknak is relációban kell tárolódnia. Az eredményreláció viszont nem részhalmaza az induló relációnak, ugyanis az eredményreláció minden egyes rekordja összesítő adatokat tartalmaz az induló relációk megadott rekord előfordulásainak egy-egy csoportjára.

Aggregáció jele: $\Gamma^{\text{aggregációs kifejezés}}\text{reláció}()$
A csoportképzés és aggregáció értelmezése: a csoportképzés és aggregáció művelete során előbb csoportokba válogatjuk szét a rekord előfordulásokat. A szétválogatás során azon rekordok kerülnek egy csoportba, melyekre egy megadott kifejezés megegyező értékű. Minden csoportra egy rekordot fog tartalmazni az eredményreláció. Ez a rekord a csoportbeli rekord előfordulásokból számított aggregációs értékeket tartalmaz.

Csoportképzés jele: $\Gamma^{\text{csoportképző kifejezés, aggregációs függvény}}_{\text{csoportképző kifejezés}}\textit{reláció}()$

A csoportképzés művelete tehát három bemenő paramétert is igényel.

1. Annak a relációnak az azonosító neve, amelyre a csoportképzés vonatkozik.
2. A csoportképzés alapjául szolgáló kifejezés. A csoportképzés értelmezése alapján a rendszer minden alaptáblabeli rekordra kiértékeli a megadott kifejezést, és azokat a rekordokat, melyekre a kifejezés ugyanazon értéket szolgáltatja, egyazon csoportba osztja be. Így tehát annyi különböző csoport jön létre, ahány különböző értéket ad a kifejezés a tábla rekordjainál.
3. Az egyes csoportokra kiszámítandó kifejezéseket határozza meg.

Ezeknek az értékeknek is egyértelműeknek, egyértékűeknek kell lenniük. Mivel egy csoporton belül több rekord is elhelyezkedhet, az alaptábla mezőinek is több értéke lehet, így ezek a mezők csak akkor maradhatnak meg az eredményrelációban, ha azok csoportképzés alapjául is szolgáltak, mert csak ebben az esetben biztosítható az értékek elemisége. A csoportképzés alapjául szolgáló mezők mellett e csoportokra a csoport egyes elemeiből képzett összesítő értékeket szokták még szerepeltetni az eredményrelációban. Minden egyes csoportra lehet összesített adatokat képezni aggregációs függvények segítségével.

Aggregációs függvények:

- **count():** az előfordulások darabszáma
- **sum():** az előfordulások valamely mezőjének összege
- **max():** az előfordulások valamely mezőjének maximuma
- **min():** az előfordulások valamely mezőjének minimuma
- **avg():** az előfordulások valamely mezőjének átlaga.

Az eredménytábla állhat egy sorból és lehet üres is, attól függően, hogy hány csoport képződött a rekord előfordulásokból.

68 Al- lekérdezések használata, alias nevek

Az al- lekérdezést mindig zárójelben kell megadni, hogy elemei elkülönüljenek, elválaszthatók legyenek a fő lekérdezés opcióitól. Minden lekérdezés eredményrelációjának lehet alias nevet adni az AS kulcsszóval, mellyel a későbbiekben hivatkozhatunk rá.

69 Halmaz műveletek, kiterjesztés jellemzése

Az R1(A1,...,An) és R2(B1,...,Bm) relációsémák **kompatibilisek**, ha n = m és dom(Ai) = dom(Bi) minden i-re. Két **táblát kompatibilisnek** nevezünk, ha sémáik kompatibilisek.

- **Unió:** Legyenek a T1 és T2 kompatibilis táblák. Ezek halmazelméleti egyesítése a T = T1 ∪ T2 tábla lesz, amelynek sémája szintén kompatibilis T1 ill. T2 sémájával.
- **Metszet:** Két kompatibilis tábla halmazelméleti metszete: T=T1 ∩ T2
- **Különbség:** Két kompatibilis tábla halmazelméleti különbsége: T = T1 \ T2
- **Kiterjesztés:** A lekérdezésben nem egyes konkrét mezőértékeket, hanem az azokból matematikai műveletekkel képzett eredményeket jelenítünk meg. Az eredmény relációban újabb mezőt képezünk a kiinduló reláció mezőivel végzett műveletek eredményének megjelenítésére. Jele: ε kifejezés (reláció)

70 Osztás jellemzése (*)

Az R1 és R2 relációk hányadosa az a reláció, amelybe R1 mindazon rekordjainak projekciói beletartoznak, amelyeknek az R2-vel való Descartes-szorzata a legnagyobb részhalmazát alkotja az R1-nek. Jele: reláció1 / reláció2

71 Lekérdezések végrehajtása algebrában (-)

GYAKORLATI

72 Relációs algebra korlátai (*) (!)

73 Algebrai parancsok konvertálása hálós modellre (*)

- JOIN → ciklus
- szelekció → ciklus és feltétel
- projekció → ciklus és csak az igényelt elem kell

74 SQL nyelv jellemzése, története

Az SQL, (strukturált lekérdezőnyelv) a relációs adatbázis-kezelők lekérdezési nyelve. A nyelvben az utasításokat a pontosvessző választja el egymástól.

Az SQL nyelvi elemeket 4 részre lehet bontani:

- Adatdefiníciós (Data Definition Language, DDL)
- Adatkezelési (Data Manipulation Language, DML)
- Lekérdező (QUERY)
- Adatvezérlő (Data Control Language, DCL)

Az SQL alapjait az IBM-nél fektették le, még az 1970-es években. Elvi alapot a relációs adatmodell szolgáltatott, amit Edgar F. Codd híres 12 szabályával írt le először, 1970-ben.

Az IBM, az Oracle és más gyártók is érdekeltek voltak egy szabványos lekérdező nyelv kifejlesztésében, amivel a relációs adatbázisok programozhatók. Az iparági összefogással létrejött ANSI NCITS H2 csoport lerakta az SQL alapjait.

A szabványt az ANSI 1986-ban, az ISO 1987-ben jegyezte be. Az első változatot SQL86 néven is szokták emlegetni. Az SQL-t folyamatosan továbbfejlesztették, és hat jelentős kiadást különböztetünk meg: SQL86, SQL89, SQL92, SQL99 (SQL3), SQL:2006, SQL:2008

75 SQL parancsok csoportjai

- DDL (Data Definition Language) – adatdefiníciós nyelv
- DML (Data Manipulation Language) – adatkezelő nyelv
- DQL (Data Query Language) – adatlekérdező nyelv
- DCL (Data Control Language) – adatvezérlő nyelv

76 CREATE TABLE parancs szintaktikája

CREATE TABLE tnev (m1 t1 [lok.integ.felt.], m2 t2 [lok.integ.felt], …, [glob.integ.felt.]);

77 INSERT parancs szintaktikája

INSERT INTO tnev VALUES (ertek1, ertek2, ertek3, …);
A mezők sorrendje meg kell, hogy egyezzen a sémadefinícióban megadott sorrenddel, minden mezőhöz kell értéket rendelni (üres érték = NULL), szigorú típusellenőrzés van.

INSERT INTO tnev VALUES (mezo1=ertek1, mezo8=ertek8, mezo4=ertek4, …);

78 UPDATE parancs szintaktikája

UPDATE tnev SET mezo1 = e1, mezo2 = e2 … [WHERE feltetel];

Feltételnek megfelelő rekordok megadott mezőértékeit módosítja (a feltétel megadása opcionális, elhagyásával a tábla összes rekordjában módosul a megadott mező(k) értéke)

79 DELETE parancs szintaktikája

DELETE FROM tnev WHERE feltetel;

Csak a feltételnek megfelelő rekordokat töröljük. A feltétel elhagyható, akkor az összes rekord törlésre kerül a táblából.

80 SELECT parancs szintaktikája,komponensei

SELECT [DISTINCT] projekciós rész FROM alapreláció [

WHERE szelekció

GROUP BY csoportképző kif.

HAVING csoport szelekció

ORDER BY mezo1 [ASC/DESC], mezo2 [ASC/DESC] …

];

A DISTINCT kulcsszó jelentése: Az eredménytáblában ne legyen ismétlődés a mezőértékek között.

81 Projekció, szelekció megvalósítása SELECT -ben

- Projekció: SELECT projekciós rész FROM reláció;
- Szelekció: SELECT projekciós rész FROM reláció WHERE feltétel;

82 JOIN típusok megvalósítása SELECT -ben

SELECT projekciós rész FROM alapreláció [INNER | NATURAL | LEFT | RIGHT | FULL | CROSS] JOIN reláció [ON kapcsolási feltétel] …

A CROSS JOIN Descartes szorzatot képez, azaz minden elem párosítva lesz minden elemmel.

83 Csoportképzés, aggregáció megvalósítása SELECT -ben

A csoportképzésre a GROUP BY függvény szolgál, az aggregációra pedig a különböző aggregációs függvény: AVG(), SUM(), MIN(), MAX(), COUNT().

84 Al- SELECT használata, rendezés, aliasok SELECT -ben

Az al- lekérdezést mindig zárójelben kell megadni, hogy elemi elkülönüljenek, elválaszthatók legyenek a fő lekérdezés opcióitól. Az al- SELECT SELECTen belüli SELECT, tehát allekérdezés. Példa: SELECT mnev FROM tnev WHERE (SELECT COUNT(mnev2) FROM tnev) > tnev.mnev2;

Az eredményrelációt egy mezője szerint az ORDER BY kulcsszóval rendezhetünk növekvő (INC – elhagyható, ez az alapértelmezett), vagy csökkenő (DESC) sorrendben is.

Példa: SELECT nev, fizetes FROM dolgozok WHERE dolgozok.tipus = ”Alkalmazott” ORDER BY fizetes DESC;

SQL-ben lehetőségünk van a relációknak, vagy számított mezőknek nevet adni az AS kulcsszó segítségével.

Példa: SELECT tipus, SUM(fizetes) AS ”Fizetendő összeg” FROM dolgozok GROUP BY tipus;

85 Darabszám korlát, ismétlődés tiltás, sztring hasonlítás (LIKE, SIMILAR) SELECT-ben (*)

Egyes adatbázis-kezelő rendszerekben meg lehet adni darabszám korlátot a LIMIT N kifejezéssel, ahol N a darabszámot jelenti.

Az ismétlődő elemeket kiszűrni a DISTINCT kulcsszóval lehet.

Például: SELECT DISTINCT gyarto FROM autok; – Listázzuk ki az összes gyártót, de mindegyiket csak egyszer!

Az SQL-ben két -féle lehetőség is adódik a sztringek összehasonlítására:

- **LIKE:** A LIKE kulcsszóval egy egyszerű mintaillesztést lehet megadni
  Használata: kifejezés LIKE minta, ahol a mintában szerepelhet a '_' (tetszőleges egy karaktert helyettesít) és a '%' (tetszőleges szövegrészt helyettesít) speciális karakter
- **SIMILAR:** A SIMILAR kulcsszó segítségével reguláris kifejezést adhatunk meg, ahol a REGEXP mintára illeszkedő szöveg egyezőségét fogadja el.
  Használata: kifejezés SIMILAR TO RegExp.

86 Reguláris kifejezések (!)

87 CASE szerkezet SELECT-ben (*)

SELECT mnev CASE kif WHEN kif1 THEN kif2 WHEN… ELSE kif0 END FROM tnev;

88 NULL érték kezelése, operátorai

A NULL egy állapotot fejez ki, ami azt jelenti, hogy nincs értéke az adott mezőnek.

Használata:

- UPDATE ... SET m=NULL;
- SELECT .... WHERE m IS NULL;
- SELET m FROM

89 3VL működése, hatása (*)

SQL-ben 3 logikai érték van: TRUE, FALSE, UNKNOWN, ezeken végzett műveletek igazságtáblái pedig a következők:

p	q	p OR q	p AND q	NOT p
TRUE	TRUE	TRUE	TRUE	FALSE
TRUE	FALSE	TRUE	FALSE	FALSE
TRUE	UNKNOWN	TRUE	UNKNOWN	FALSE
FALSE	TRUE	TRUE	FALSE	TRUE
FALSE	FALSE	FALSE	FALSE	TRUE
FALSE	UNKNOWN	UNKNOWN	FALSE	TRUE
UNKNOWN	TRUE	TRUE	UNKNOWN	UNKNOWN
UNKNOWN	FALSE	UNKNOWN	FALSE	UNKNOWN
UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN

90 Hierarchikus SELECT működése, parancsai (*)

Mivel a relációs algebra nem tartalmaz rekurzív elemet, az RDBMS fejlesztők kibővítették a SELECT utasítás funkciókörét úgy, hogy alkalmas legyen hierarchia bejárásra is. A neve hierarchikus SELECT. Emögött tehát egy rekurzív fa bejárási algoritmus van. Nem része az SQL szabványnak, sem a relációs algebrának. Oracle szintaktika:

SELECT mezőlista FROM táblanév WHERE feltétel1 START WITH feltétel2

CONNECT BY PRIOR mező1=mező2 ... ;

Ahol:

- feltétel2: a bejárás induló elemét kijelölő feltétel

- CONNECT BY ... : a szülő-gyermek kapcsolatot kijelölő kapcsolódási feltétel, a PRIOR mögötti a szülő
- feltétel1: a bejárásba bevont rekordok szűkítése; itt lehet korlátozni a mélységet is, a LEVEL opcióval.

91 Indexkezelés SQL-ben, automatikus index generálás

Indexet akkor használunk, amikor egy adott mező alapján gyakran keresünk. Az elsődleges kulcsnak a DBMS automatikusan létrehoz egy indexet (implicit létrehozás), de mi is létrehozhatunk tetszőlegesen bármelyik egyéb mezőre is indexet (explicit létrehozás):

CREATE INDEX inev ON tábla (mezőkif)

Sokan ajánlják, hogy minden idegenkulcsra hozzunk létre indexet, ugyanakkor a sok index a teljesítményre káros hatással van beszúrásnál, és törlésnél, ugyanis akkor sokat kell módosítani az indexekben.

92 Származtatott táblák szerepe, működése

A származtatott táblák olyan táblák, amelyek más táblákból egy lekérdezés útján származtatva lettek. Származtatott táblákat létrehozhatunk kézzel, illetve a DBMS allekérdezéseknél – az eredmény ideiglenes tárolása végett – automatikusan létrehoz.

A kézzel létrehozottak a rendszer katalógusában tárolódnak, és két fajtája van:

- **VIEW:** Létrehozáskor csupán az eredményt előállító utasítás tárolódik. Használatakor a lekérdezésnek mindig le kell futnia. Akkor célszerű használni, ha a lekérdezés eredményrelációja sűrűn módosul, vagy a lekérdezés lefutása a DBMS számára nem megterhelő. Használata:

  CREATE VIEW wnév AS SELECT … ;

- **Materialized VIEW:** Létrehozáskor az eredményt előállító utasításon kívül maga az eredményreláció is eltárolódik, magától nem frissül, így elképzelhető, hogy a tábla nem a legfrissebb adatokat tartalmazza. A frissítés történhet kézzel, vagy automatikusan. Akkor célszerű használni, ha az eredményrelációt ritkán módosítjuk, azonban a lekérdezés futási ideje nagy, és sűrűn felhasználjuk. Használata (Oracle):

  CREATE [MATERIALIZED VIEW | SNAPSHOT] MVnév REFRESH FAST START WITH SYSDATE NEXT SYSDATE + 1 AS SELECT …;

93 VIEW kezelés parancsai

- Létrehozás: CREATE VIEW wnév AS SELECT …;
- Használat: SELECT * FROM wnév;
- Törlés: DROP VIEW wnév;

94 VIEW használata védelemben és összetett aggregációban

A VIEW -et kell használni adathozzáférés finomítására is. A tábla mellett le lehet hozni mező vagy rekord szintre (nem mindet) Létrehozunk egy VIEW -ot és ahhoz adunk jogot: GRANT SELECT ON wnév

95 VIEW módosítás jellemzése (*)

Ha a VIEW -ben módosítunk valamilyen adatot, az a táblában is módosul amiről a VIEW -ot készítettük, feltéve hogy nincs benne származtatott lekérdezés, mert különben nem lenne egyértelmű.

96 VIEW CHECK opció működése (*)

A CREATE VIEW parancsnak egy része. Ha ott van a CHECK option, akkor nem lehet olyan módosításokat végrehajtani amely megsértené a VIEW szelekciós feltételét.

97 Materialized View (SNAPSHOT) kezelése, működése

Létrehozáskor az eredményt előállító utasításon kívül maga az eredményreláció is eltárolódik, magától nem frissül, így elképzelhető, hogy a tábla nem a legfrissebb adatokat tartalmazza. A frissítés történhet kézzel, vagy automatikusan. Akkor célszerű használni, ha az eredményrelációt ritkán módosítjuk, azonban a lekérdezés futási ideje nagy, és sűrűn felhasználjuk. Használata (Oracle):

CREATE [MATERIALIZED VIEW | SNAPSHOT] MVnév REFRESH FAST START WITH SYSDATE NEXT SYSDATE + 1 AS SELECT …;

98 DBMS védelmi modellje

A tranzakció sikeres befejezésének utasítása a COMMIT. Ennek hatására a korábban végrehajtott tevékenységek véglegesítődnek, megőrződnek az adatbázisban. A ROLLBACK hatására a korábbi tevékenységek érvénytelenítődnek, ez a hiba utáni visszaállítás. Ezek a parancsok nem az adatbázis adatain manipulálnak, hanem az adatcache adatain. A tranzakció lezárásának módjától függően íródnak át az adatok.

99 GRANT parancs szintaktikája

GRANT művelet ON tábla TO felhasználó [WITCH GRANT OPTION];

100 REVOKE parancs szintaktikája

REVOKE művelet ON táblázatnév FROM felhasználó;

101 Jelszó tárolás és kezelés módjai (*)

102 SQL API típusai

- Natív SQL
- E-SQL
- C-SQL
- O-SQL
- 4GL-SQL

103 Natív SQL API jellemzése

Az SQL maga a gazdanyelv része.

104 E-SQL API jellemzése

A gazdanyelv utasításai közé beszúrjuk a parancsokat, SQL szabvány szerinti szintaktikával. Az ezen az elven működő bővítések a gazdanyelvi beágyazások. Jól elkülönülnek a két nyelv utasításai.

105 CLI API jellemzése

A gazdanyelv szintaktikájának megfelelő formalizmussal lehet az SQL utasításokat végrehajtani. Ekkor a gazdanyelv utasításai közé eljárások, függvények formájában szúrjuk be az adatkezelő tevékenységeket.

Pl. INSERT Oracle-ben: osql(&cursor,"INSERT INTO auto VALUES('fad534','Opel',4);",-1);

Ezt a mechanizmust hívják CLI -nek (Call Library Interface), utalva rá, hogy a kapcsolat könyvtári függvények hívásán keresztül valósul meg.

106 OO-API jellemzése

Objektum orientált API. Objektumokon keresztül végezzük el az adatkezelést.

107 ODBC middleware működési elve (!)

108 Adat átadás, átvétel módjai

Először bekérjük az új adatot, majd kiadjuk az UPDATE utasítást. Az előfordító számára ismertté kell tenni az adatokat amikkel dolgozni akarunk:

```
BEGIN DECLARE SECTION;
        adatok;
END DECLARE SECTION;
```

EXEC SQL -el kell kezdeni minden SQL utasítást, amit a gazdanyelvben használunk. A deklarációs blokkban minden használt SQL változónak szerepelnie kell. A deklarációnak szintaktikailag és szemantikailag is kompatibilisnek kell lennie a gazdanyelvvel. Input gazdanyelvi változónak hívják, aminek az értékeit bevisszük az adatbázisba, output gazdanyelvi változónak pedig azt, ami a DB -ből kap értéket.

109 Hibakezelés elvei

A programozónak expliciten be kell építenie az SQL hibák kezelésére vonatkozó elemeket. Deklarálni kell egy megadott szerkezetű struktúrát, amihez az RDBMS is hozzáférhet, tehát egy kommunikációs területet, amiben az RDBMS elhelyezi a hibakódot. A hibakezelésre szolgáló struktúra deklarálása:

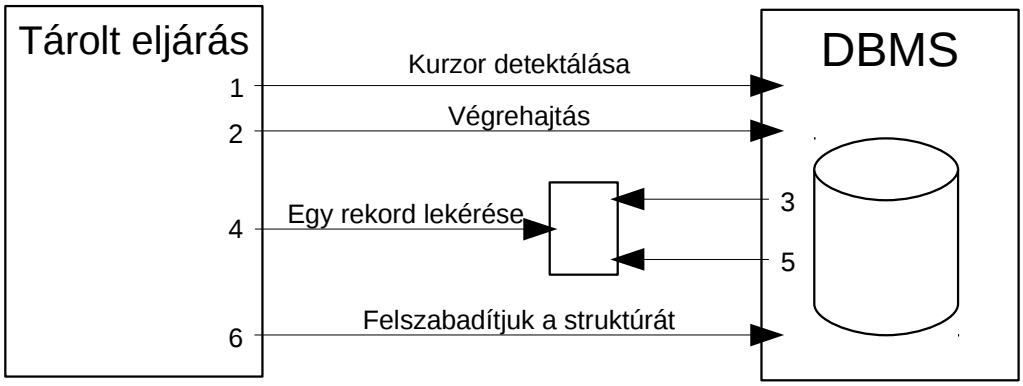
INCLUDE sqlca; ebben benne vannak a hiba információi. Közvetett hibakezelésnél ezt nézi a rendszer. Közvetlen hibakezelésnél egy automatikus hibakövetés opció folyamatosan ellenőrzi az SQLCA -t, és adott hiba esetén elvégzi az adott tevékenységet. Ennek parancsa: WHENEVER hiba tevékenység;

Hiba típusai: SQLERROR (végrehajtási hiba), SQLWARNING (kivétel), NOT FOUND (ha SELECT üres táblázatot ad vissza, vagy a FETCH a lista végére ért).

A tevékenység típusai: CONTINUE, STOP, GOTO címke, DO függvény().

A WHENEVER -nél oda kell figyelni a forrásszövegbeli pozíciójára GOTO használat esetén, illetve, hogy végtelen ciklust idézhet elő a GOTO + ROLLBACK páros.

110 CURSOR működési elve az API-nál



111 CURSOR kezelés parancsai

- Kurzor deklarálása: DECLARE kurzornév FOR SELECT ... ; Ezzel tesszük ismertté a szerkezetet
- Kurzor megnyitása: OPEN kurzornév; A SELECT végrehajtódik, létrejön az eredménytábla első rekordjára mutató pointer.
- Rekordok lekérdezése: FETCH [NEXT|PREV|FIRST|LAST] kurzornév INTO változólista; Annyi FETCH utasítást kell kiadni, amennyi rekordot az eredménytábla tartalmaz. A lekérdezés történhet bármelyik irányba.
- Kurzor lezárása: CLOSE kurzornév; Felszabadul az eredménytáblázatnak lefoglalt hely, megszűnik a pointer.

112 Módosítható CURSOR működése, parancsai(*)

Módosítható kurzor segítségével rekord-orientált módon, egyenként módosíthatóak, törölhetőek a rekordok. A kurzor deklarációjának végén a FOR UPDATE|DELETE [OF (mező1(, mező2...)]; opcióval megadjuk, hogy módosítani vagy törölni szeretnénk a kiválasztott rekordokat. Mezőlista megadása esetén csak azok, egyébként minden mező módosítható lesz. Ezután a rekordfeldolgozó cikluson belül kiadott UPDATE vagy DELETE szűkítése következik úgy, hogy a WHERE szelekciós feltételünkben a CURRENT OF kurzornév; feltételt adjuk meg.

113 PHP - DBMS működési struktúra (!)

114 PHP-ODBC API jellemzése

Függvény alapú.

1. Kötődés az adatbázis szerverhez
2. SQL parancs elküldése
3. Az elküldött eredményből egy rekordot kér vissza
4. Az áthozott rekordból egy mezőt fog visszahozni

115 odbc_connect parancs

Kapcsolódik az adatbázis szerverhez. Paraméterei: DSN, felhasználónév, jelszó. Visszaadja az erőforrás sorszámot.

116 odbc_exec parancs

SQL parancs elküldése. Paraméterei: erőforrás azonosító, SQL string. Visszaad egy eredmény halmazt.

117 odbc_fetch_row parancs

Az elküldött eredményből egy rekordot kér vissza. Egy eredményhalmaz lesz az eredménye.

118 odbc_result parancs

Az áthozott rekordokból egy mezőt fog visszahozni. Paraméterei: result set, mezőazonosító.

119 Prepared parancsok jellemzése (*)

120 PHP - böngésző adatkapcsolat jellemzése (!)

121 Minta program fejlesztése PHP-ban (-)

GYAKORLATI

122 mySQL RDBMS jellemzése (!)

123 mySQL installáció lépései,paraméterei

1. Telepítő letöltése, futtatás
2. Paraméterezés.
   Adatkezelés jellege:
   - transactional database only
   - multifunctional database
3. DBMS processzek kezelése
4. telepítés helye
   A műveletek jellege
   - OLTP tranzakció orientált rendszer - sok kicsi parancs módosítás jelleggel, pl helyjegyfoglalás
   - OLAP adatelemzés
5. Karakterkészlet kezelése: utólag nem módosítható, két dolgot kell megadni:
   1. Karakterkészletet
   2. Rendezési sorrendet
6. Szerver elérési címe
7. DBA jogosultsági beállítása

124 OLAP, OLTP fogalmai

- **OLAP:** adatelemzés, adatbányászat
- **OLTP:** a vállalat külső állapota

125 mySQL adatbázis típusok (myISAM, innoDB) jellemzése

MyISAM

- Ez volt az alapértelmezett tároló motor a MySQL -hez
- általában jó hely és időhatékonyságú
- stabil
- Tábla szintű zárolás
- full text index
- Könnyebb adminisztráció
- Relatív gyorsaság

InnoDB

- Az alapértelmezett tárolómotor MySQL -hez
- modern alternatívája a MyISAM-nak
- tranzakciókezelés (az adatbázisműveletek tranzakcióként hajthatók végre)
- Sor szintű zárolás

- Több felhasználó egyszerre használhatja a táblát
- Idegenkulcs
- Gyors összeomlás helyreállítás: automatikusan és nagyon gyorsan konzisztens állapotba kerül az adatbázis
- ACID-elvek (atomicity, consistency, isolation, durability)
- Nagyobb biztonság
- Nagyobb memóriahasználat
- Létezik statikus, dinamikus és tömörített MyISAM

126 Tárolt eljárások jellemzése, előnyei

Több SQL parancs egy eljárásban/függvényben (+vezérlési szerkezetek) Előnyei:

- egyszeri adatkapcsolat
- kicsi adatforgalom
- hatékony ismétlődő, összetett feladatoknál
- védelem

127 CREATE PROCEDURE és CREATE FUNCTION parancsok

CREATE FUNCTION függvény_név (paraméterlista) RETURNS típus

BEGIN

...

END;

A PROCEDURE ugyan ez kivéve, hogy nincs visszatérési értéke, tehát a RETURNS elmarad.

128 Változó létrehozás (DECLARE)

BEGIN DECLARE SECTION;

adatok;

END DECLARE SECTION;

129 SELECT INTO parancs

A beépített SQL esetén szükség van egy fogadó változó kijelölésére a SELECThez, erre van egy módosított SELECT.

SELECT mezőlista INTO eredménylista FROM ...; Ezt akkor lehet használni, ha egy rekord jön át.

130 Vezérlési parancsok a tárolt eljárásban (!)

131 Anomáliák fogalma és típusai;

A redundancia mellett számos egyéb műveleti nehézséget is okozhatnak a modell hiányosságai. A nem megfelelő relációs sémából eredő problémákat szokás anomáliáknak nevezni.

Az anomáliák legfontosabb típusai

- **Beszúrási:** ismétlődő adatelem újabb beszúrásakor, ha nem pontosan úgy adjuk meg az adatot, ahogy korábban (pl. elgépeljük), akkor új adatelem válik belőle.
- **Törlési:** ismétlődő adatelem törlésekor, minden előfordulási helyen törölni kell az adatot.
- **Módosítási:** ismétlődő adatelem módosításakor minden adatbázisbeli előfordulást módosítani kell.

132 Funkcionális függőség értelmezése és jelölése

A -> B : A előforduláshoz egyetlen B érték tartozik

Redundancia akkor lép fel, ha a sémában van egy A -> B funkcionális függőség és A értéke ismétlődhet.

A tervezés célja az ismétlődő értékű attribútumokból kiinduló FD-k megszüntetése;

133 Redundancia fogalma

Egyes adatelemek feleslegesen többször is le vannak tárolva.

134 Redundancia oka

Egyes mezők nem függetlenek egymástól, illetve egy mező értéke ismétlődik.

135 Normalizálás fogalma és célja;

A normalizálás olyan eljárás sorozat, melynek célja anomáliamentes relációséma létrehozása/előállítása. Az adatbázis tervezési folyamat egy eleme, mely több, egymásra épülő lépésből (normálforma) áll és minden lépéshez tartozik egy kritérium.

136 Első, második normálforma

- 1NF: egy R séma 1NF-ben van, ha minden attribútum egyértékű és létezik kulcs mező.
- 2NF: egy R séma 2NF-ben van, ha 1NF teljesül és minden attribútum a teljes kulcstól függ, nem annak egy részkulcsától (csak összetett kulcsok esetén van értelme vizsgálni, elemi kulcs esetén 2NF automatikusan teljesül).

137 BCNF normálforma

BCNF (általánosított 3NF): egy R séma akkor van BCNF -ben, ha 2NF teljesül és funkcionális függőség csak jelölt kulcsból (olyan mező, mely egyértelműen meghatározza a többi mező értékét, egyértelműen azonosítja a rekord-előfordulást) indul ki

138 Armstrong szabályok

- Ha A része B-nek akkor B -> A (az egész meghatározza a részét)
- Ha A -> B akkor AC -> BC (kibővíthetőség)
- Ha A -> B és B -> C akkor A -> C (tranzitivitás)

139 Irreducibilis FD mag és szerepe (*)

A funkcionális függőségek legtömörebb halmazát irreducibilis FD halmaznak nevezzük. Ez az a legkisebb halmaz, amely segítségével előállítható a sémában előforduló összes funkcionális függőség.

140 Dekompozíció szerepe

A dekompozíció során az induló sémát részekre bonjuk, a nem kívánt FD-ket külön relációkba emeljük ki. A normalizáláshoz van rá szükség.

141 Veszteségmentes dekompozíció

A reláció felbontása a redundancia csökkentésével, információvesztés nélkül. Az új relációk JOIN -ja az eredeti táblát adja vissza.

142 Heath tétele

Ha R(A,B,C) adott és teljesül, hogy A->B , akkor (PI)_AB és (PI)_AC veszteségmentes felbontás. R reláció, A,B,C attribútumcsoport.

143 Független dekompozíció

Ha R(A,B,C) adott, akkor (PI)_AB , (PI)_AC független, ha

- R(A,B,C) minden FD -je származtatható R(A,B) és R(A,C) FD -iből, és
- A legalább az egyik felbontásban jelölt kulcs.

144 Rissanen tétele

Cél: független dekompozíció biztosítása és R(A, B, C) → R1(A,B), R2(A,C) független, ha:

- FD(R1) és FD(R2) → FD(R)
- A→ B v A→C

b) FD (R) ={ rsz → tipus, rsz → gyarto, tipus → gyarto }
FD (R1) = {rsz → gyarto}
FD (R2) = {rsz → tipus}

c) FD (R1) = {tipus → gyarto}
FD (R2) = {rsz → tipus} - ezt le tudjuk vezetni, a tétel is ezt fogja kiválasztani

145 Atomi felbontás (*)

Nincs hozzá független felbontás.

146 Atomiság és BCNF kapcsolata (*)

Létezik atomi de nem BCNF formula. Például: R(tanar, diak, targy)

147 Normalizálási szintek kapcsolata (*)

2NFhez szükséges az 1NF

3NFhez és BCNF -hez szükséges a 2NF

Tehát: BCNF <- 2NF <- 1NF

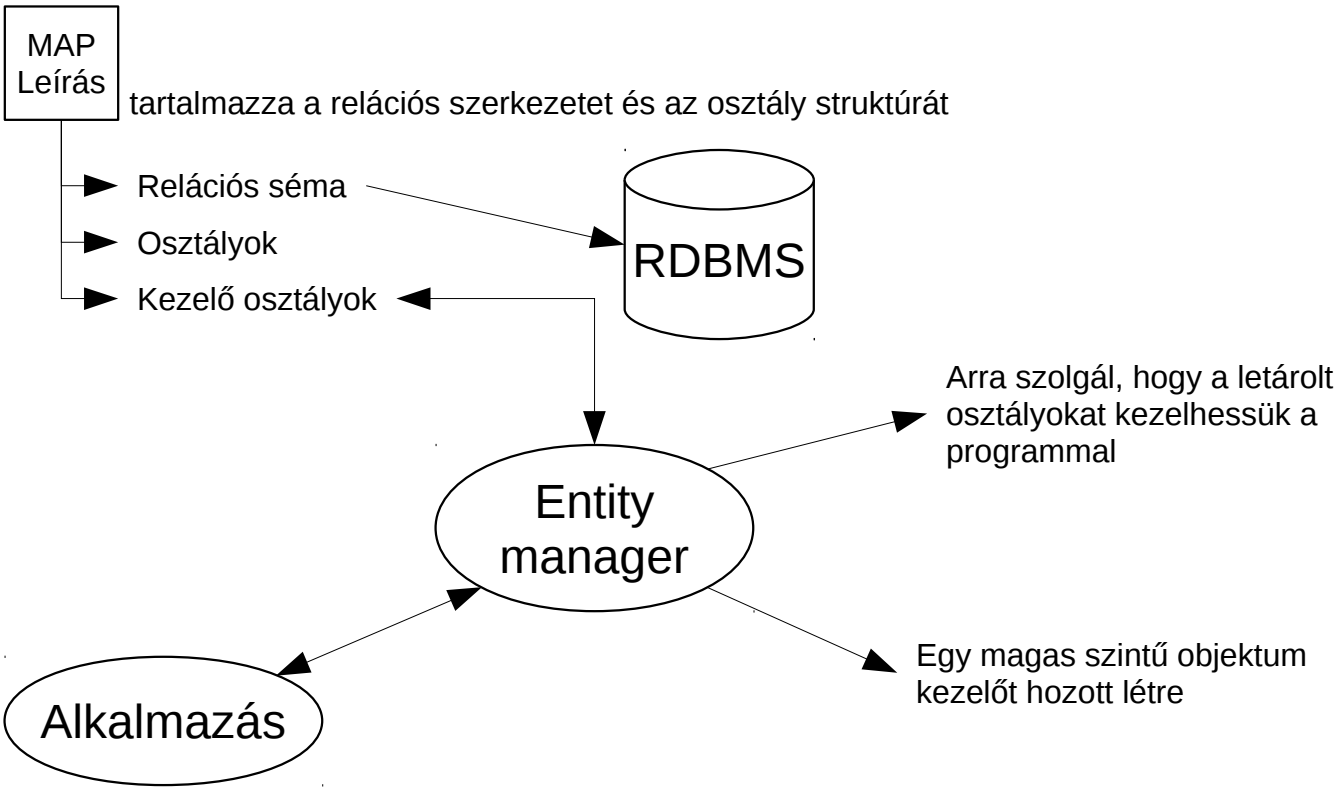
148 Séma normalizálása

GYAKORLATI

149 OOP osztályok leképzése relációs sémára(*)

Osztály	→	reláció
objektum	→	rekord
adattagok	→	mező
összetett	→	külön mező
tömb	→	külön rekord
asszociáció	→	kapcsolat

150 Hibernate struktúra elemei(*)



151 JPA struktúra elemei (*)

JPA (Java perzisztens API) A HIBERNATE-ra épül rá. Maga a JPA megközelítés egy alkalmazásból indul ki, a java forrást lehet ellátni annotációkkal. Annotációk a mappingre vonatkoznak.

152 Replikáció fogalma és működési struktúrája,módjai (!)

153 Elosztott adatbázisok (DDBMS) működési elve

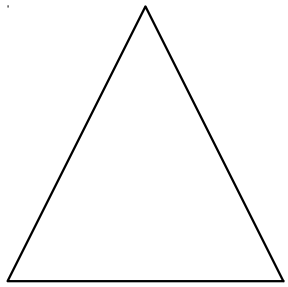
Több önálló szerver. Egyetlen önálló adatbázisként jelennek meg. Master csomópont – koordinálást, szinkronizációt oldja meg.

Több speciális probléma is fellép:

- Hibák kezelése. Nincs tökéletes hibakezelő protokoll!
- Az elosztott adatkezelésnél a CAP elv érvényesül.

154 CAP elv az elosztott adatbázisoknál

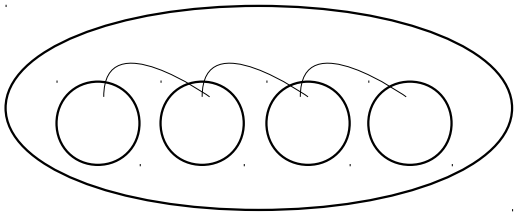
C – konzisztencia: bármely tag is dolgozza fel, ugyanazon képet látnak



A – Rendelkezésre állás

P – feldarabolhatóság: ha a hálózat megszakad mindegyik rész önállóan tud dolgozni

155 RAC architektúra jellemzése (*)



logikailag 1 adatbázis kezelő szoftver több hálózati csomópontra szétosztva működik

Felhasznált erőforrások

Forrás	Internetes hivatkozás
Eredeti jegyzetet készítette: Huzinec Erik	http://users.iit.uni-miskolc.hu/~huzynets/adatb1/KERDESEK%20VIZSGARA%20valasszal.docx
Varga Attila Károly kidolgozott tételsora (2001/2002)	
Kobakbt informatikai tudástár	http://www.kobakbt.hu/jegyzet/AdatbazisElmelet/ora3_index.html
Wikipédia SQL szócikk	http://hu.wikipedia.org/wiki/SQL
Angol nyelvű Wikipédia NULL_(SQL) szócikke	http://en.wikipedia.org/wiki/Null_(SQL)

A dokumentum készítése során kizárólag szabad szoftverek kerültek felhasználásra. Jelen dokumentum a LibreOffice 3.4.4-es verziójával készült. Az eredeti forrásállományt nyomós indokkal és egyben a karbantartói szerepkör el-/át- vállalásával a jelenlegi karbantartó elektronikus levélcímén elkérhető.

Jogi nyilatkozat

A dokumentum jelenleg Pozsgay Máté gondozásában áll. Kérdéseivel, észrevételeivel, javaslataival bátran keresse őt a matthew.linux@gmail.com elektronikus levélcímen.