DB zh 2 összefoglaló

Objektumrelációs adatbázis-kezelők

Az objektumrelációs adatbázis-kezelők **legfőbb elemei**:

1. Extensible data types
2. Methods / operations
3. Extensible indexing

Jellemző használati területei:

- Geographical data / Spatial database(Maps) (a legfontosabb) - használt program: PostgreSQL
- Multimedia/complex data types eg. XML - használt program: Oracle

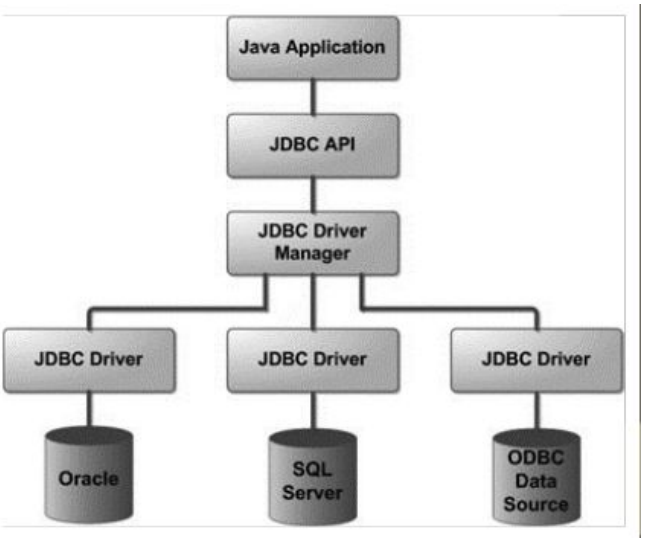
JDBC

Mi az a JDBC?

Java Database Connectivity, Java programnyelvhez API ad adatbázis hozzáférés támogatást. Tudja

- Kapcsolat létrehozása az adatbázissal
- Lekérdező / módosító parancsok küldése
- A lekérdezés eredményeinek feldolgozása

JDBC felépítése



Mi az a JDBC driver? Milyen típusai vannak? (Melyik a leggyakoribb típus, amely egyben az Oracle thin driver típusa is?)

Kliensoldali adapter, amely a Java kérései átalakítja az adatbázis-szerver által értelmezhető formára.

- Type 1 - JDBC-ODBC híd
- Type 2 - natív-api driver
- Type 3 - hálózati protokoll (middleware)
- Type 4 - native protocol driver, pure java
    - Install JVM on client side, better performance. different vendors different protocols.

Mik a kapcsolat felépítés lépései?

1. Betöltjük a drivert
2. Csatlakozunk az adatforráshoz
3. SQL utasítás végrehajtása

Milyen interfészeket tartalmaz a JDBC API?

- JDBC driver manager - betölti a drivert
- JDBC driver
- JDBC datasource - végrehajtja a lekérdezést

Mi az a Statement, PreparedStatement és CallableStatement? Mi a különbség a Statement és a PreparedStatement között?

Statement: Segítségével hajthatjuk végre az SQL utasításokat, amik egy ResultSet eredményt adnak.

- executeQuery: paraméterbe lekérdezés, majd visszaadja
- executeUpdate: paraméterben adott utasítást végrehajtja, majd a módosított sorok számát adja vissza
- execute: előző kettő metódus általánosítása

```
Statement stat1 = myCon.createStatement ();
 Stat1.executeQuery("SELECT * FROM test");
```

PreparedStatement: Paraméterezhető sql lekérdezés

```
PreparedStatement stat2 = myCon.prepareStatement ("SELECT beer, price FROM
Sells WHERE bar= ?");
```

CallableStatement: Eljárás- és függvényhíváshoz

```
CallableStatement stat3 = myCon.prepareCall ("{call JoeMenu(?, ?)}");
```

Mik a Prepared Statement tulajdonságai/előnyei?

- Cachelődik
- lekérdezési terv egyszer készül el
- SQL injectiontől véd

ResultSet

A lekérdezés eredményeit tartalmazza soronként, úgy viselkedik, mint egy kurzor/iterátor.

- next() - tel kövi sorba megy és true-t ad, ha még van elem
- getInt(i), getString(i) - információ kiszedése, i sorszám / név (tábla oszlopnév igazából)
- previous(), last(), first() - a next()-hez hasonlóan

```
ResultSet menu = stat1.executeQuery("SELECT beer, price FROM Sells WHERE bar =
'Joe''s Bar');
```

Miért használnak connection poolt?

To enhance the performance of executing commands on a db. After a connection is created, it is placed in the pool and it is used again so that a new connection doesn't have to be established.

Mi az az SQL(code) injection?

Az SQL injection egy platformfüggetlen hackelési eszköz, mely speciális db lekérdezést küld a kiszolgálónak, hogy hozzáférjen a db-hez. Erre egy koncepciós hiba kínál lehetőséget.
$query = “SELECT user FROM tbl_user WHERE name = ‘$name’”;
Hacker inputs: admin’ OR 1=1 --

JDBC megfelelője a Microsoft világban (C#, Visual Basic)?

ODBC

ORM, Hibernate

MI az Object-Relational Mapping (ORM)?

Objektum relációs leképezés: Az ORM egy programozási technika, ami objektum orientált rendszert hivatott leképezni egy nem kompatibilis rendszerbe (pl. relációs adatbázis). A cél, hogy az objektumok tulajdonságait és kapcsolatait megőrizzük.

Mi az a Hibernate?
A Hibernate egy teljesen Java alapú framework és egy ORM, amely lehetővé teszi az objektumrelációs leképezést: POJO-k relációs adatbázishoz való kapcsolását.
Nagyon hatékony leképezést hoz létre az objektumok és az adatbázis táblák közt.
A fejlesztés sokkal egyszerűbb és gyorsabb.

Hibernate kommunikációs egységek

- **SessionFactory**: Szálbiztos Session létrehozása. Ezt az interfacet egyszer implementálják az alkalmazások.
- **Session**: A kliens és adatbázis közötti kommunikációt reprezentálja. A komm. során Transaction objektumokat hoznak létre, ezek reprezentálják az elvégzett munkát. Nem szálbiztos.
- **Perzisztens objektumok**: Rövid életű objektumok, egy sessionhöz tartoznak. Ha a session véget ér, akkor elérhetővé válnak más objektumok számára is.

Hibernate felépítése:

- Java osztály
- Relációs adatbázisbeli táblák
- Descriptor (konverziós szabályok definiálása):
    - XML
    - Annotáció

Alapesetben egy osztály: tábla
Egy példány: sor

Milyen két módja van a konverziós szabályok megadásának? (descriptor)

- XML fájlok
- Annotációk használata

Milyen kardinalitású kapcsolatokat képes kezelni a Hibernate?

- @OneToOne
    - 1-1, Join Column-al történik
- @ManyToOne
    - N-1 az N oldali netitásban, Join Column (kapcsoló mező)
- @OneToMany
    - 1-N az 1 oldali entitásban, Join Table, de érdemesebb az N oldali entitás táblájába illesztett kapcsoló mezővel megvalósítani
- @ManyToMany
    - N-M, Join Table, de lehet érdemes köztes entitást használni

MIlyen két lehetőség van a kapcsolódó objektumok betöltésére?

- **Lusta(Lazy)**: kollekció tartalma akkor inicializálódik, amikor először hivatkozunk rá. / a kapcsolatot csak akkor töltjük be, ha azt külön kérjük
- **Mohó (Eager)**: Kollekció betöltése után rögtön van hivatkozás rá. / az objektum betöltése során betölti a hozzá tartozó kapcsolatot

Használna ORM-et (Hibernate-et) egy tranzakciós alkalmazás esetén? Egy adattárházban? (Gondolkodós feladat, próbálja megindokolni)
Hibernate főleg kisebb adatok 1-1 kapcsolat esetén hasznos, viszont nagyobb adattárházak esetén a nagy adatállomány miatt nem jó.
Ilyenkor jobb a JDBC.

Igaz-hamis (Ebből sok nem lesz)

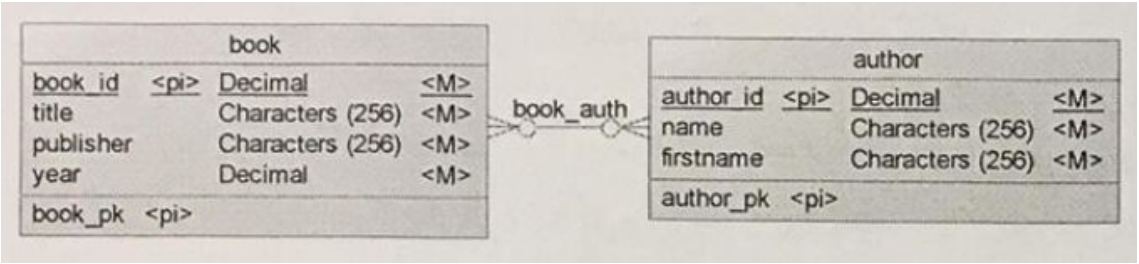
A. táblázatban jelölje, igaz vagy hamis az állítás! (12 pont)

1. A PL/SQL az SQL olyan procedurális kiterjesztése, amely jól szabványosított.
2. A PL/SQL programkód az adatbáziskezelő szerveren fut le, azzal jól integrált (ebből következően pl. tranzakciókat és a jogosultságkezelést is támogatja)
3. Az objektumorientált adatbáziskezelő rendszerek mára rendkívül elterjetté váltak.
4. Sok relációs adatbáziskezelőként ismert rendszer (pl. Oracle, PostgreSQL) tartalmaznak objektumrelációs kiterjesztéseket, és így valójában objektumrelációs adatbáziskezelők.
5. A B+ fa alapú indexek jól használhatóak földrajzi adatokhoz.
6. Az objektumrelációs adatbáziskezelés jól szabványosított, valamennyi gyártó megvalósítása egységes.
7. Célszerű a objektumrelációs lehetőségeket akár egyszerű üzleti adatokhoz is használni.
8. A Google által használt földrajzi vonatkoztatási rendszer, a World Geodetic System (SRID=4326) közvetlenül és egyszerűen használható távolságmérésre.
9. Egy adattárház tulajdonképpen tranzakciós adatok másolata, kifejezetten lekérdezési, elemzési célra.
10. A B+ fa alapú index jól támogatja az adattárházak átfogó lekérdezéseit.
11. Egy materializált nézet lényegében bármilyen SQL kifejezést tartalmazhat.
12. Az in-memory adatbáziskezelőkben feleslegesség vált az SQL procedurális kiterjesztése, az SAP HANA rendszere is csak (deklaratív) SQL-t támogat.

	Igaz	Hamis
1	X	
2	X	
3		X
4	X	
5		X
6		X
7		X
8	X	
9	X	
10		X
11		X
12		X

Ez nem tudom melyik téma

Készítse el az alábbi ER diagram alapján a szükséges relációs táblákat (relációs séma diagramot adjon meg). Különösen figyeljen az elsődleges és a külső kulcsokra!



- Book_auth(book_id, author_id)
- Book(Book_id, title, publisher, year)
- Author(Author_id, name, first_name)

Az elméletben tanult Cehn-notációs ER diagram mely elemeivel fordul elő, hogy a gyakorlatban használt tervező eszközök, notációk nem támogatják?

1. derived attribute
2. multivalued attribute
3. discriminator

Nevezze meg a jelentéssel bíró kulcsok (“natural key”) ill. a szintetikus kulcsok (“surrogate key”) összesen legalább három előnyét a másikkal szemben!
Natural key előnyei:

- (+) readable by humans
- (+) attribute already exists
- (-) maintenance
- (+) unique

Surrogate key előnye:

- (+) uniform
- (+) performance
- (+) compatible
- (+) maintenance

PL/SQL (Ez nem lesz)

A **cursor** segítségével végre lehet hajtani olyan SQL lekérdezéseket, melyek több sort adnak vissza, és az eredményüket PL/SQL-ben szeretnénk feldolgozni. Ennek jellegzetes műveletei a felhasználás szokásos logikai sorrendjében:

1. Open
2. Fetch
3. Check if found
4. Close

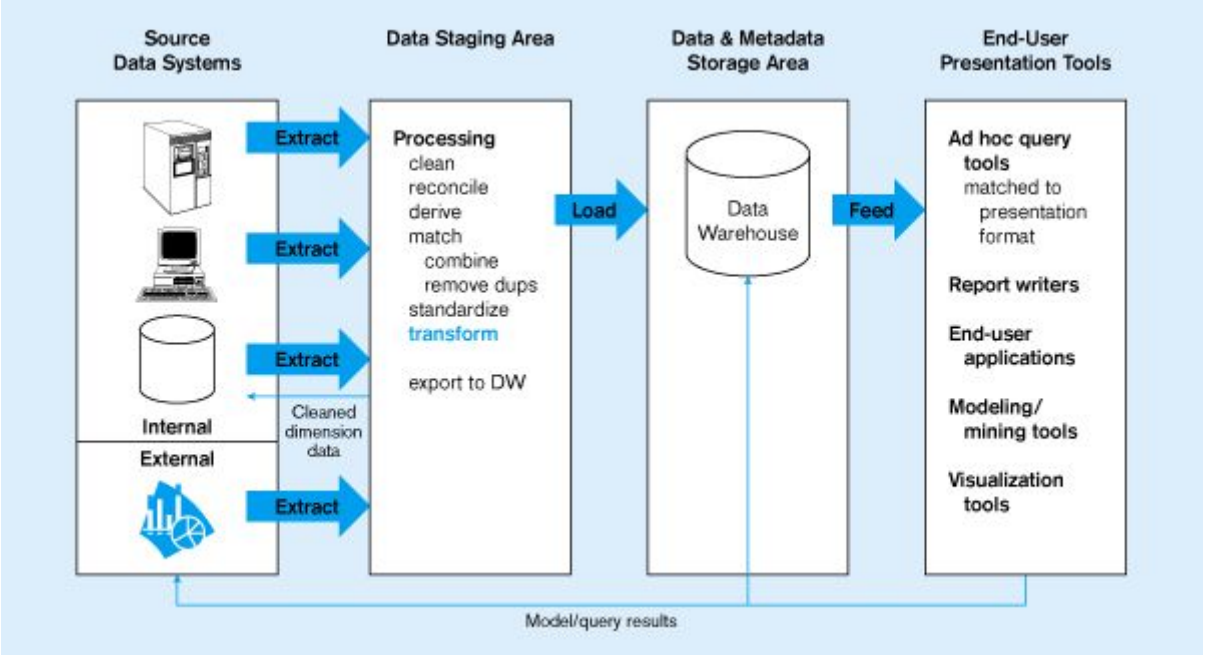
A **trigger** egy olyan PL/SQL eljárás, mely automatikusan a háttérben végrehajtódik egy bizonyos adatbázis művelet esetén.

Nevezze meg legalább két módját annak, hogyan tud SQL kifejezést beágyazni PL/SQL kódba (mikor használhatóak ezek a bágyazások?):

1. INTO - csak ha a lekérdezés eredménye egy sor
2. CURSOR - ha a lekérdezés több sort ad vissza

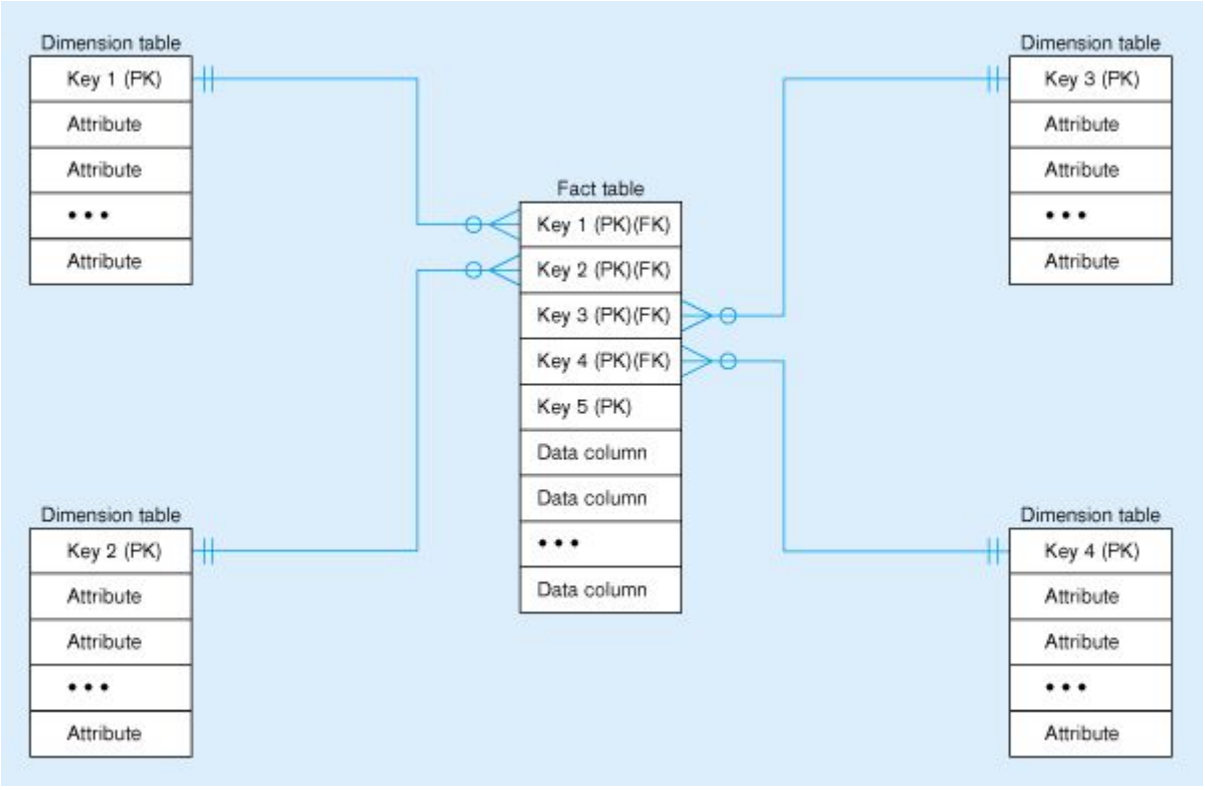
Adattárházak (Ehelyett idén Tableau volt, szal nem biztos, hogy lesz [bár az ehhez kapcsolódik szóval…])

Rövid definíció: A copy of transaction data, specifically arranged, grouped from operational data sources, to support **business decisions**, reporting etc.
Szokásos architektúra:



Az ETL betűszó jelentése: Extract, Transform, Load

Az adattárházak szokásos sémamegoldása az ún. **csillagséma**, mely **fact** táblákból és **dimenzió** táblákból áll. Ez az adatbázis séma jól támogatja az ad-hoc, interaktív lekérdezéseket. Ezt a sémamegoldást ill. ezeket a lekérdezéseket támogatják jól a **bitmap-indexek**.



A BI eszközök interaktív adatkockájának 3 fő művelete:

- Slice
- Drill down
- Roll up

B+ fa index korlátai

Nevezzen meg két olyan alkalmazási területet, amelyet a szokásos B+ fa alapú index nem támogat kellőképpen!

- Földrajzi lekérdezések > az adatok nem rendezhetőek
- DWM (?) sok rekordot érintő (nem szelektív) lekérdezések

Milyen más megoldás van ezekre a feladatokra?

- R-fa például kifejezetten földrajzi lekérdezéshez jó
- Bitmap-index, max. view