Adatbázisrendszerek
11. előadás: Objektum-orientáltság adatbázisokban
Objektumorientált és objektum-relációs adatbázisok

2023. május 12.

DEBRECENI EGYETEM

Előadás vázlata

11. előadás: Objektum-orientáltság adatbázisok-ban

Bevezetés

OO adatbázisok

OO alapfogalmak

Objektum-relációs adatbázisok

- Bevezetés
- Objektumorientált adatbázisok
- Az ODMG (Object Database Management Group) szerepe
- OO alapfogalmak
- Az SQL objektumorientált kiterjesztései: objektum-relációs adatbázisok

11. előadás: Objektum-orientáltság adatbázisok-ban

Bevezetés

OO adatbázisok

OO alapfogalmak

Objektum-relációs adatbázisok

Hagyományos adatmodellek melyek sikeresek voltak a tradicionális alkalmazásoknál:

- Hierarchikus
- Hálózati (a 60's évek közepe óta)
- Relációs (1970 óta illetve 1982 óta)

Objektumorientált (OO) adatmodellek 90's évek közepétől. Az objektumorientált adatbázisok létrehozásának okai:

- Növekvő igény a komplex adatbázis alkalmazások iránt, pl. CAD/CAM, CIM, tudományos, telekom, GIS, multimédia
- Komplexebb struktúrájú objektumok: összetett tranzakciók, új adattípusok (képek, filmek, nagy szöveges állományok), nem-standard alkalmazás-generált műveletek
- Igény arra, hogy kezelni tudjunk tetszőleges adattípust a hagyományos lekérdező nyelvek megtartásával
- OO programozási nyelvek elterjedése melyekkel a hagyományos adatbázis rendszerek nehézkesen működtek

11. előadás: Objektum-orientáltság adatbázisokban

Bevezetés

OO adatbázisok

OO alapfogalmak

Objektum-relációs adatbázisok

Fő jellemző és előny: a tervező specifikálni tudja mind a sokszorosan összetett objektumok szerkezetét, mind az ezeken az objektumokon alkalmazható műveleteket.

- Object Databases (O(O)DB): Objektum-orientált adatbázisok
- Object Database Management Systems (OD(B)MS): Objektum-orientált adatbázis-kezelő rendszerek
- Objektum-relációs rendszerek (ORDBMS): a tradicionális RDBMS-ek OO szemlélettel való bővítései (Oracle), SQL 2008 szabvány (először mint SQL/Object majd az SQL/Foundation része)
- Szabványosítás az ODMG (Object Database Management Group) konzorcium keretében: ODMG 3.0 szabvány
- Tisztán OO rendszerek. Prototípusok: ORION, OpenOODB, IRIS (HP), ODE. Üzleti rendszerek: GEMSTONE/S Object server, ONTOS DB, Objectivity/DB, Versant Object Database, ObjectStore, Ardent

OO fogalmak áttekintése

- Objektum-orientált (OO) szó eredete az OO programozási nyelvekre vezethető vissza
- Objektum azonosító (OID) fogalma
- Az osztály fogalmának bevezetése, egy objektum belső adatstruktúrájának leírása egy osztály deklarációban
- Absztrakt adattípus (absztrakt adatobjektum) fogalmának bevezetése: objektum szerkezete és típus konstruktorok
- A műveletek egységbezárása (encapsulation), a belső adatstruktúra elrejtése, az alkalmazható külső műveletek, metódusok meghatározása
- Perzisztencia (állandóság) és tranziencia (ideiglenesség)
- Típus vagy osztály hierarchia és öröklődés
- Hibrid OO programozási nyelvek: C++

Objektum

- Két komponense van: állapot (tulajdonság, érték) és viselkedés (művelet)
- „Hasonló” a program változójához kivéve, hogy összetett adatstruktúra is lehet és a programozó speciális műveleteket definiálhat rajta.
- Az objektumok az OO programozási nyelvekben tranziensek (átmenetiek), a futás befejeztével törlődnek.
- Az objektumok az O(R)DMS-ekben perzisztensek (állandóak), eltárolódnak, később kinyerhetőek és megoszthatóak más programokkal.
- Egy O(R)DMS általában több OO programnyelvhez kapcsolódik állandó és osztott objektumokat szolgáltatva számukra.

Objektumok jellemzői

- Direkt kapcsolat a valós világ és az adatbázis objektumai között az állandó (megváltozhatatlan) objektum azonosító (OID) révén.
- Tetszőlegesen összetett szerkezetű objektum megengedett (minden az objektummal kapcsolatos információ tárolható, míg a relációs adatbázisokban az információ sok relációban szóródik szét).
- Példányváltozók (instance variables): azon értékeket hordozzák, melyek az objektum belső állapotát írják le. Hasonló a relációs modell attribútumához kivéve, hogy a külső felhasználók számára nem láthatóak.
- Teljes egységbezárás: minden a felhasználók által használható műveletnek előre definiáltnak kell lenni. (Túl szigorú, pl. a lekérdezésekhez tudni kell az attribútum neveket, az ad-hoc lekérdezéseket nehéz specifikálni.)

Objektumok jellemzői

- Az egységbezárás támogatására a műveletek két részből állnak: interfész (szignatúra) és metódus (törzs). Az első a művelet nevét és paramétereit írja le, a második a művelet implementációját specifikálja.
- Típusok és osztályok hierarchiája és öröklődése: típusok és osztályok inkrementális létrehozása, újrahasznosítás
- Reláció-kezelés: az egységbezárás miatt ez elrejtésre került a külső felhasználók számára (korai rendszerek problémája), az ODMG szabvány ezt megoldotta a bináris kapcsolatok explicit megjelenítésével, inverz hivatkozási párral
- Verzió-kezelés: egy objektumnak időbeni története van, ennek a nyomonkövetése. Sémaevolúció (ld. ALTER TABLE)

11. előadás: Objektum-orientáltság adatbázisok-ban

Bevezetés

OO adatbázisok

OO alapfogalmak

Objektum-relációs adatbázisok

- Művelet túlterhelés (operator overloading): ugyanaz a művelet többféle típusú objektumon is végrehajtható, a művelet nevéhez több különböző implementáció tartozik attól függően, hogy milyen objektumra kívánjuk alkalmazni. (operátor polimorfizmus – polymorphism)
- Késői kötés: a futás során kapcsolódik össze a művelet neve és a megfelelő implementáció.

Objektumok konstrukciója

- Az O(R)DMS-ekben az objektumok és a literálok (OID nélküli értékek) tetszőleges összetettségű típus szerkezettel bírhatnak, amely az összes szükséges információt tartalmazza.
- A hagyományos adatbázis rendszerekben az összetett objektumokkal kapcsolatos információk sok relációra illetve rekordra szóródnak szét elvesztve így sokszor a direkt kapcsolatot a valós világ objektumai és azok adatbázisbeli reprezentációi között.
- OO databázisokban egy összetett objektum állapota más objektumok alapján határozódik meg ún. típus konstruktorok segítségével.
- Alapvető típus konstruktorok: atom, rekord (struct, tuple), kollekció (több-értékű).
- További konstruktorok a kollekción belül: halmaz (set), lista (list), zsák (bag), tömb (array), szótár (dictionary).

Műveletek egységbezárása

11. előadás: Objektum-orientáltság adatbázisok-ban

Bevezetés

OO adatbázisok

OO alapfogalmak

Objektum-relációs adatbázisok

- Cél: absztrakt adattípusok kezelése, információ-elrejtés (szemben a tradicionális relációs modellel, ahol minden reláció összes attribútumával együtt látható a felhasználók és külső programok számára).
- Egy objektumtípust viselkedésének leírásával definiálunk, melyet az objektumtípushoz külső hozzáférést biztosító műveleteken alapulnak.
- Műveletek: objektumok létrehozása, törlése, állapotának módosítása, egyes részeihez való hozzáférés biztosítása ill. ezek kombinációi.
- Műveletek felépítése: interfész (szignatúra) mely a műveletek nevét és paramétereit definiálja (látható), implementáció (metódus) az objektum belső adatszerkezetét és az eléréséhez szükséges műveletek leírását tartalmazza (rejtett).
- Teljes egységbezárás: túl erős követelmény, általában az objektumok szerkezetét két részre, látható és rejtett attribútumokra osztjuk.

Objektumok állandóságának biztosítása

- Általában nem minden objektumtípus lesz állandó (perzisztens) egy adatbázisban, ezt külön biztosítani kell.
- Az állandóság biztosítása: elnevezési mechanizmussal vagy elérhetőséggel történik.
- Elnevezési mechanizmus: egyértelmű és állandó nevet ad egy objektumnak, mellyel hivatkozhatunk rá külső programokból (belépési pont).
- Elérhetőségi mechanizmus: segitségével perzisztens objektumokból érhetünk el további objektumokat (így elég kevés számú perzisztens objektumot definiálnunk).
- Egy B objektum elérhető az A objektumból ha az objektum gráfban hivatkozások (élek) egy sorozata vezet A-ból B-be.
- Az OSSZES_HALLGATO objektum az HALLGATO osztály extentje, amely minden HALLGATO típusú objektumot perzisztenssé tesz.

Típus (osztály) hierarchia és öröklődés

- Számos objektumot szeretnénk kezelni lehetőleg minél egyszerűbben.
- Ennek módja új típusok létrehozása korábbiakból.
- Típus: (név, attribútumok, műveletek)
- Függvények: attribútumok (példányváltozók), műveletek.
  Attribútum: argumentum nélküli függvény
- TYPE_NAME: function, . . . , function
  Példa: SZEMÉLY: Név, Cím, Szül_dátum, Kor, Szsz
  ahol a Kor művelet, amellyel az életkort számolhatjuk ki, a többi pedig attribútum.
- Szubtípus (altípus): egy olyan új típus, amely egy már definiált típus összes függvényét tartamazza.
- Szupertípus (szülőtípus): az a típus, amelyből a szubtípust származtattuk.

Példa öröklődésre

SZEMÉLY: Név, Cím, Szül_dátum, Kor, Szsz (definiált típus)
Tegyük fel, hogy az alábbi új típusokat szeretnénk definiálni:

DOLGOZÓ: Név, Cím, Szül_dátum, Kor, Szsz, Fizetés, Alk_kezdete, Beosztás
HALLGATÓ: Név, Cím, Szül_dátum, Kor, Szsz, Szak, Tan_átlag

Mivel mindkettő tartalmazza a SZEMÉLY típus összes függvényét, így annak altípusaként definiálhatóak:

DOLGOZÓ subtype-of SZEMÉLY: Fizetés, Alk_kezdete, Beosztás
HALLGATÓ subtype-of SZEMÉLY: Szak, Tan_átlag

A DOLGOZÓ altípus tárolt attribútumokként tartalmazza a Fizetés, Alk_kezdete és Beosztás attribútumokat. A HALLGATÓ altípus tárolt attribútuma a Szak, míg a Tan_átlag műveletként implementálható.

Típus (osztály) hierarchia

11. előadás: Objektum-orientáltság adatbázisok-ban

Bevezetés

OO adatbázisok

OO alapfogalmak

Objektum-relációs adatbázisok

- Típus hierarchia: az összes szuper/szub típus kapcsolat rendszere
- Átnevezés: hierarchia révén származtatott függvények átnevezése
- Permanens és tranziens kollekciók
- Többszörös öröklődés: akkor beszélünk róla ha egy altípus kettő vagy több típus altípusa és így értelemszerűen örökli mindkettő vagy az összes függvényét (attribútumait és metódusait).
- Szelektív öröklődés: amikor egy altípus csak egy típus bizonyos függvényeit örökli, amelyeket nem azokat az EXPECT klózzal jelezzük

Extentek

- Ugyanolyan típusú objektumok egy kollekciója
- A cél az, hogy az objektumainkat állandóvá tegyük
- Általában az összes azonos típusú objektum szerepel a típushoz tartozó extentben
- Object: minden objektumot tartalmazó extent
- Megszorítás az extenteken, hogy legyenek kompatibilisek a típus hierarchiával, azaz az altípus extentje legyen része a szubtípus extentjének.

define class HALLGATO_CSOPORT **type set**(HALLGATO);
operations add_hallgato(d: HALLGATO): **boolean**; (új hallgató)
remove_hallgato(d: HALLGATO): **boolean**; (hallgató törlése)
create_hallgato_csoport: HALLGATO_CSOPORT; (új hallgató csoport létrehozása)
destroy_hallgato_csoport: **boolean**; (hallgató csoport megszüntetése)
end HALLGATO_CSOPORT;
persistent name OSSZES_HALLGATO: HALLGATO_CSOPORT;
d:=create_hallgato; (új hallgató létrehozása a d változóban)
b:=OSSZES_HALLGATO.add_hallgato(d); (d perzisztenssé tétele)

Objektum-relációs adatbázisok jellemzői

SQL:2008 szabvány: a tradicionális RDBMS-ek OO szemlélettel való bővítése (pl. Oracle, MS SQL Server).
A következő objektumorientált jellemzőkkel bővült az SQL:

- Típus konstruktorok melyekkel összetett objektumokat hozhatunk létre. Például:
  - rekord (*row type*), mely megfelel a rekord (tuple, struct) konstruktornak
  - tömb (*array type*), mellyel kollekciókat hozhatunk létre
  - a további kollekció típusokkal, ld. halmaz, lista, zsák, később bővült a szabvány
- Objektumok azonosítását biztosító mechanizmus a *reference type* segítségével.
- Műveletek egységbezárása a felhasználó által defniált típusokon (UDT). A felhasználó által definiált eljárás (UDR) szintén megjelenik.
- Öröklődési mechanizmus az UNDER kulcsszó segítségével.

UDT: felhasználó által definiált típus

Az SQL által nyújtott felhasználó által definiált típus (UDT: user-defined type) célja:

- összetett szerkezetű (a relációs modell rekordjainál bonyolultabb) objektumok létrehozása
- egy típus deklarációjának elválasztása a tábla (reláció) létrehozásától
- rekord típusú konstruktor a **ROW** kulcsszóval rekord típusú attribútumok létrehozására
- 4-féle kollekció típus: **ARRAY** (a kezdeti specifikációban csak ez volt), **MULTISET**, **LIST** és **SET**

Egy UDT létrehozásának módja:

CREATE TYPE típus_neve AS (komponensek deklarációja)

UDT: felhasználó által definiált típus

- Objektumok azonosítása egyértelmű, rendszer által generált OID-vel referncia típus útján:
  REF IS SYSTEM GENERATED
  Emellett használható a relációs modell hagyományos kulcsa is.
- A példányosítható (**INSTANTIABLE** kulcsszó) UDT-khez táblákat (relációkat) is létrehozhatunk.
- Az UDT-khez műveleteket (metódusokat) is definiálhatunk:
  CREATE TYPE típusnév (
  <attribútumok listája a típusaikkal>
  <metódusok (függvények) deklarációi>);
- Attribútumok és műveletek három fajtája:
  - PUBLIC - látható az UDT interfészen
  - PRIVATE - nem látható az UDT interfészen
  - PROTECTED - csak az altípusok számára látható

Példa UDT-ra


```
CREATE TYPE SZEMELY_TIPUS AS (
    NEV VARCHAR (35),
    NEM CHAR,
    SZUL_DATUM DATE,
    LAKCIM CIM_TIPUS
INSTANTIABLE
NOT FINAL
REF IS SYSTEM GENERATED
INSTANCE METHOD AGE() RETURNS INTEGER;
CREATE INSTANCE METHOD AGE() RETURNS
INTEGER
    FOR SZEMELY_TIPUS
    BEGIN
        RETURN /* programkód egy személy életkorának
kiszámolására a mai dátumból és a SZUL_DATUM-ból */
    END; );
```

Példa UDT-ra

Lakcím típusát a következőképpen definiáljuk:

```
CREATE TYPE CIM_TIPUS AS (
    UTCA_CIM ROW (KOZTERULET VARCHAR (10),
                  NEV VARCHAR (25),
                  HAZSZAM NUMBER (4),
                  EMELET NUMBER (2) ),
                  AJTO NUMBER (2) ),
    VAROS VARCHAR (25),
    IR_SZAM NUMBER (4)
);
```

A kódban szereplő **ROW** kulcsszóval egy összetett rekordot tudunk megadni, amely az utcai címet több komponensből állítja elő. Vegyük észre, hogy ez a típus nem példányosítható hiszen szerepe csupán az, hogy a SZEMELY_TIPUS lakcímét deklarálni tudjuk.

Példa UDT-ra: szubtípus létrehozása


```
CREATE TYPE KURZUS_TIPUS AS (
        KURZUS_KOD CHAR (8),
        EV CHAR (4),
        JEGY CHAR
);
CREATE TYPE HALLGATO_TIPUS UNDER
SZEMELY_TIPUS AS (
        SZAK_KOD CHAR (10),
        NEPTUN_KOD CHAR (6),
        SZEMESZTER NUMBER (2),
        KURZUSOK KURZUS_TIPUS ARRAY [100]
INSTANTIABLE
);
```

A HALLGATO altípusa a SZEMELY-nek, viszont már nem lehet neki további altípusa. Egy hallgató által felvett kurzusokat egy 100 elemű tömbben tároljuk. Mindkét típus példányosítható.

Táblák létrehozása

Az alább létrehozzuk az egyetemi dolgozók, oktatók és hallgatók tábláját:

```
CREATE TABLE DOLGOZO OF SZEMELY_TIPUS
        REF IS DOLGOZO_ID SYSTEM GENERATED;
CREATE TABLE HALLGATO OF HALLGATO_TIPUS
        UNDER DOLGOZO;
CREATE TABLE OKTATO OF OKTATO_TIPUS
        UNDER DOLGOZO;
```

A DOLGOZO tábla egyetemi dolgozókból, mint rekordok, fog állni, amelyek mindegyikének típusa SZEMELY_TIPUS.
Az SQL egy további képessége a táblaöröklődés a szuper/szubtábla kapcsolat útján az UNDER kulcsszó révén. Minden beszúrás (INSERT) a HALLGATO vagy az OKTATO táblákba egyben egy beszúrást jelent a DOLGOZO szupertáblába is. Hasonló igaz a többi DML műveletre is (DELETE, UPDATE).

11. előadás: Objektum-orientáltság adatbázisok-ban

Bevezetés

OO adatbázisok

OO alapfogalmak

Objektum-relációs adatbázisok

Az SQL szabályai típus öröklődésre az UNDER kulcsszó alatt:

- A **NOT FINAL** kulcsszót kell használni ha egy UDT-nek további altípusát szeretnénk deklarálni.
- Minden attribútum öröklődik.
- A szupertípusok sorrendje az UNDER kulcsszó után határozza meg az öröklődési sorrendet.
- Egy altípus példánya minden olyan kontextusban használható ahol szupertípusának példánya használható.
- Egy altípus minden a szupertípuson definiált függvényt újradefiniálhat feltéve hogy a szignatúra nem változhat.
- Egy függvény hívásakor a legjobban illeszkedő implementáció kerül alkalmazásra az összes argumentum típusát figyelembe véve.
- Dinamikus kötéskor a paraméterek futáskori típusait veszi figyelembe.