Adatbázis rendszerek I
2011.11.10

Mai témák:
SQL kiterjesztések
- tárolt eljárások és függvények (mySQL)
- SQL-API alapok

Tárolt eljárások:
egy üzleti feladat, funkció ->több SQL parancs

GÁZ - szolgáltató mérés adat gyűjtő

Mérőműszer leírása több táblával irható le (normalizálható)

Egy mérő felszerelés mögött értékegyezőségek, több SQL parancs állhat.

Egy logikai művelet -> több SQL

kliens ? DB-szerver

Megvalósítható:

Kliens oldalt bontjuk fel elemi SQL-re	Szerver oldalt bontjuk fel
Problémák:	Egyszeri adatkapcsolat
- nagy adatforgalom	Tárolt eljárás / függvény
- több különálló SQL	(lesz mögötte vezérlési funkció)

Tárolási eljárások
- adatforgalom
- erőforrás
- séma követés (karbantartás)
- védelem
- hatékonyság

Hátránya: A tárolt eljárások SQL specifikusak, tehát NEM szabványos

TÁROLT eljárást miért célszerű használni?
- ismétlődő
- összetett
- hatékony

Használata

Létrehozás
CREATE FUNCTION *függvényNév* (*paraméterlista*) **RETURNS** *tipus*
BEGIN
...
END;

Törzs
- SQL parancsok
- változók
DECLARE *név tipus*
- értékadás **SET** *nev=kif*
- vazérlési elem
IF *felt* **THEN**
ELSE
END IF

- Ciklus
LOOP
...
END LOOP - végtelen ciklus

WHILE *felt* **LOOP**
...
END LOOP
RETURN *kif*

Kapcsolat a lokális változók és SQL parancsok között.
A változók beépíthetők az SQL parancsba.

PÉLDÁK

Készítsünk függvényt ami két szám összegét adja vissza

```
delimiter //
CREATE FUNCTION oa (a int, b int) returns int
BEGIN
        return a+b;
END //
        delimiter;
```

Megjegyzés: A delimiter az egészet hajtja végre, nem részenként.

A tárolt függvényeket fel kell tudjuk használni
SELECT oa (1, 2);

Két számból a nagyobbat:

```
delimiter//
CREATE FUNCTION oa2 (a int, b int) returns int
BEGIN
        DECLARE e int;
        if a>b THEN
                SET e=a;
        ELSE
                e=b;
        END if;
        return e;
END//
delimiter;
```

N-faktoriális kiszámítása

```
delimiter//
CREATE FUNCION oa3 (a int) returns int
BEGIN
        DECLARE e int;
        DECLARE i int;

        SET e=1;
        SET i=1;

        WHILE i<=a do
                SET e=e*i;
                SET i=i+1;
        END WHILE
        return e;
        END//
```

delimiter;

Adatkapcsolat a TÁROLT eljárás és a DBMS között?

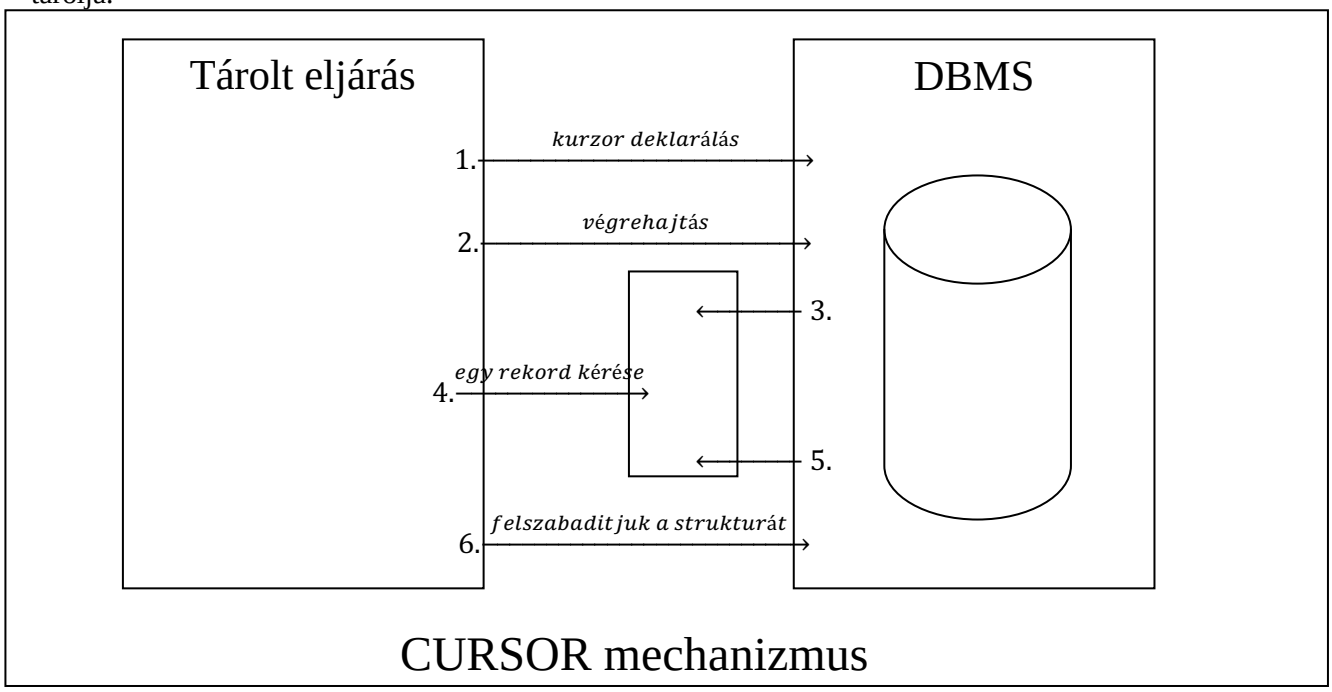
TÁROLT ELJÁRÁS -> DBMS

DECLARE a INT
SET a=3,
UPDATE dolgozo SET fiz=a
WHERE ...,

Megjegyzés:MYSQL-ben ügyelni kell a név átfedésre

TÁROLT ELJÁRÁS <- DBMS

Probléma: Eredmény tárolása. Megoldás: Nem a teljes adathalmazt veszi át a kliens, hanem rekordokba veszi át. Viszont a működéséhez kell egy köztes tároló mechanizmus amely ideiglenesen tárolja.



CURSOR mechanizmus

Különböző CURSOR modellek vannak
A ciklus leállásához különböző mechanizmusok vannak, leggyakrabban a hibáknál áll le.

Általános modell (a PHP-ban is használják)
Megjegyzés:
Az újabb API-knál a CURSOR fölé épül 1 kiszolgáló réteg ami elvégzi helyettünk ezeket a lépéseket.

Megjegyzés:
Ha az eredmény 1 db rekord akkor kikerülhető a CURSOR mechanizmus.

SELECT ... **INTO** ... **FROM END IF**

Problémák az adatbázisnál:
- NULL érték kezelése
Ha Tárolt eljárásban vagyunk, akkor nem probléma mert támogatják a NULL értéket.

Megjegyzés:A Tárolt eljárásban a DBMS-operátorok is közvetlenül használhatók.
Megjegyzés: Ügyelni kell a mezőhosszakra mert a csonkolás automatikus.

Szintaktika
DECLARE név **CURSOR FOR SELECT** ...
A CURSOR-nak 2 fajtája van:

- olvasási CURSOR-ok
- módosításra nyitható CURSOR-ok

Módosítható CURSOR célja:
A CURSOR aktuális rekordon keresztül gyorsan, hatékonyan elérjük egy forrás adatbázis rekordot.

UPDATE CURSOR esetén:
A CURSOR rekordhoz bemásolódik a forrás rekord pointere

UPDATE *tábla* **SET** ... **WHERE CURRENT OF** *kurzornév*

OPEN *kurzornév*

FETCH *kurzornév* **INTO** *változó*

A direkt CURSOR-kezelés sokkal rugalmasabb ezért is kell tudni.

SQL APi alapok

API mögött egy könyvtár van.

Típusai:

Natív SQL	SQL maga a gazdanyelv része (gazdanyelv - amiben a kliens programot irjuk) Van ilyen, pl VFP A=1 IF B>2 INSERT INTO ... VALUES (A);
E-SQL	Beágyazott SQL Az SQL parancsokat illesztünk be megfelelő szeparátorral A=1 IF B>2 EXEC SQL INSERT INTO ... VALUES (A) Arra szolgál, hogy a hagyományos SQL nyelvet lehet használni
C-SQL	Hívás alakú SQL Függvényhívásokból áll az API A=1 IF (B>2) EXEC_SQL (celdb, "INSERT INTO...", ...);
O-SQL	Objektum orientált API Objektumokon keresztül végezzük el az adatkezelést a=1 IF(b>2) Strut.exec ("INSERT INTO ... ");
4GL-SQL	 Paraméterezés Kód írása nélkül kellene készíteni a programokat nem adunk ki SQL parancsot csak az SQL parancs paramétereit adjuk át

O-SQL
C-SQL
E-SQS
} funkciók:

- adatbázis kapcsolat
- parancs (STATEMENT, exec, stb)
- eredmény fogadása (CURSOR)
- lezárás