**Tárolt eljárás**

Adatbázis-objektumként tárolt algoritmikus program, amely SQL utasításokat is tartalmazhat → az SQL algoritmikus kiterjesztése

A tárolt eljárások olyan SQL parancsok halmazát tartalmazzák, amelyeket a szerveren lehet fordítani és tárolni. Így a gyakran használt lekérdezés tárolása helyett az ügyfelek hivatkozhatnak a megfelelő tárolt eljárásra. Ez jobb teljesítményt nyújt, mivel ezt a kérést csak egyszer kell elemezni, és a kiszolgáló és az ügyfél közötti forgalom csökken.

Adatbázis rendszerenként más-más elnevezéssel bír, szintaxisa eltérő lehet. MySQL → SPL: Stored Program Language

**Tárolt eljárások előnyei:**

* Az eljárások a szerveren tárolódnak, így nem kell üzeneteket küldözgetni az SQL utasítások végrehajtásakor a kliens és a szerver között.
* Az eljárások elemzése egyszer történik meg, helyes szintaxis esetén lefordítva (futásra kész állapotban) tárolódnak.
* Az eljárásokra ugyanolyan biztonsági intézkedések vonatkoznak, mint az adatbázisban tárolt többi adatra.
* Biztonságosabb

**Kétféle tárolt rutin van a MySQL-ben:**

* Eljárás → PROCEDURE: nem ad vissza értéket
* Függvény → FUNCTION: mindig visszaad egy értéket a hívó félnek

A hívás során alkalmazhatunk paramétereket, így ezekkel vezérelhetjük a tárolt rutinok működését.

A tárolt eljárás adatbázishoz van kötve → töröljük az adatbázist, a tárolt eljárás is törlődik

**Utasítások:**

* CREATE PROCEDURE / FUNCTION: tárolt eljárások létrehozása
* ALTER PROCEDURE / FUNCTION: az eljárások módosításához vagy törléséhez szükséges
* DROP PROCEDURE / FUNCTION: törlés
* CALL: a szubrutin végrehajtása ELJÁRÁS esetén – mivel a függvényt lekérdezésbe ágyazzuk bele!!!

CREATE PROCEDURE / FUNCTION <eljárásnév>(<paraméterlista (IN, OUT, INOUT)>)

**Határoló:**

Az SQL és a tárolt eljárások esetén is a határoló a „;”. Mivel gondot okoz a fordítás során, hogy melyikhez tartozik a határoló (elsődleges az SQL), ezért erre használjuk:

DELIMITER //

tárolt eljárás

DELIMITER

**Szintaktikája:**

DELIMITER //

create procedure eljaras\_nev(paraméterlista)

begin

declare változók;

utasítások

end; //

DELIMITER ;

**Tárolt eljárás meghívása:**

call eljaras\_nev(paraméterek);

**Elágazások:**

**else-if szintaktikája:**

if feltétel then utasítás;

elseif feltétel then utasítás;

….

else utasítás;

end if;

**case szintaktikája:**

case változó

when érték then utasítás;

….

else utasítás;

end case;

Feladat:

Készítsen tárolt eljárást, mely a kerület alapján eldönti, hogy budai vagy pesti ingatlanról van szó

(budai kerületek: 1,2,3,11,12,22 a többi pesti)!

**Ciklusok:**

* LOOP – feltétel vizsgálat alapján ciklusban marad és újra végrehajt, vagy pedig kilép a ciklusból

loop\_cimke:LOOP

IF vizsgálat THEN

LEAVE loop\_cimke; elhagyja a ciklust

ENDIF;

IF vizsgálat1 THEN

utasítás1;

ITERATE loop\_cimke; ciklusban marad

END LOOP loop\_cimke;

* WHILE - amíg igaz a kifejezés, addig működik

WHILE kifejezés DO

utasítások;

END WHILE

* REPEAT - amíg igaz nem lesz a kifejezés, addig működik

REPEAT

utasítások;

UNTIL kifejezés

END REPEAT;

**Kurzorok:**

* A kurzor egy lekérdezés eredményének átmeneti tárolási lehetősége.
* Akkor használjuk, ha a lekérdezés több rekordot ad eredményül.
* Mindig egy SELECT parancs eredményeképpen jön létre
* A rekordok ciklus segítségével bejárhatók (kiírathatók, módosíthatók)
* Lépések:

1. Kurzor deklaráció (begin előtt!)

Szintaktika*: DECLARE kurzornév CURSOR FOR lekérdezés;*

1. Kurzor megnyitás

*Szintaktika: open kurzornév;*

1. Rekord kiolvasások ciklusa

*Szintaktika: fetch kurzornév into változók;*

1. Kurzor lezárás

*Szintaktika: close kurzornév;*

**Triggerek**

Olyan tárolt rutin, amely valamilyen DML művelet (insert, delete, update) bekövetkeztekor automatikusan meghívódik, és végrehajtja a benne megadott műveleteket.

A trigger egy táblához kötődik, és csak az adott táblánál bekövetkező DML műveletek aktivizálják.

**Használata:**

* ellenőrzésre
* számított adatok kiszámítására
* törlés előtt adatok mentésére
* események naplózására használjuk.

**Megkötések:**

* Egy eseményhez nem kapcsolhatunk több, ugyan arra az eseményre (INSERT, UPDATE, DELETE), ugyan abban az időben (BEFORE, AFTER) aktiválódó triggert.

Például: nem lehet két BEFORE UPDATE trigger egy táblára, de lehet egy BEFOR EUPDATE és AFTER UPDATE.

* Átmeneti táblákhoz és nézetekhez nem kapcsolhatunk triggert
* A triggerek nem módosíthatják azokat a táblákat, amikhez kapcsolva vannak.
* Nem lehet benne tárolt eljárást meghívni
* Nem lehet benne tranzakciót kezdeményezni, jóváhagyni vagy visszavonni
* Nem lehet benne SELECT parancs INTO nélkül
* Csak az aktuálisan érintett rekord mezőire hivatkozhatunk:
* INSERT esetén: NEW.oszlopnév(írható)
* DELETE esetén: OLD.oszlopnév(olvasható)
* UPDATE esetén: NEW.oszlopnév(írható), és OLD.oszlopnév(olvasható)

**Szintaktika:**

delimiter //

create trigger <triggernév> BEFORE|AFTER <dmlmuvelet>

for each row

begin

<utasítások>

end; //

delimiter ;