ADATBÁZIS ALAPÚ RENDSZEREK

Triggerek

II.

ORACLE “AUTO INCREMENT” ELSŐDLEGES KULCS

- Az Oracle kifinomult módon **támogatja** a sorszámozások generálását
- **Szekvencia**: olyan adatbázis-objektum, amely meghatározott módon számok sorozatát generálja

```
CREATE SEQUENCE sequence_name
  MINVALUE value
  MAXVALUE value
  START WITH value
  INCREMENT BY value
  CACHE value;
```

- Ahhoz, hogy egy, a MySQL AUTO_INCREMENT megszorítású elsődleges kulcsához hasonló viselkedésű oszlopot kapjunk, felveendő egy szekvencia és egy **trigger**

ORACLE “AUTO INCREMENT” ELSŐDLEGES KULCS

- Egy egyszerű megoldás, a kulcs mindig eggyel növekszik:

```
CREATE SEQUENCE emp_seq
START WITH 1
INCREMENT BY 1
NOMAXVALUE;

CREATE OR REPLACE TRIGGER emp_auto_increment_pk
BEFORE INSERT ON emp
FOR EACH ROW
BEGIN
      :NEW.empno := emp_seq.NEXTVAL;
END;
/
```

ORACLE RENDSZERESEMÉNYEK

- Az Oracle a különböző események (pl. bejelentkezés, leállítás, stb.) bekövetkeztét jelzi
- Az események két csoportja:
  - rendszeresemények – az egész adatbázist érintő történések
    - `STARTUP`
    - `SHUTDOWN`
    - `SERVERERROR`
  - kliens események – a felhasználó által végzett műveletek (pl. belépés, DDL műveletek, stb.)
    - `LOGON, LOGOFF, BEFORE CREATE, AFTER CREATE`
    - `BEFORE ALTER, AFTER ALTER, BEFORE DROP, AFTER DROP`
- definiálhatunk triggereket ezen eseményekre

1. FELADAT

- Írjunk két triggert, amelyek egy adott felhasználó belépése és kilépése esetén aktivizálódnak. A triggerek készítsenek a belépésről illetve kilépésről egy bejegyzést egy naplózó táblába, amely definíciója a következő:

```
CREATE TABLE LOGON(
  nev VARCHAR(20),
  datum DATE
);
```

- Próbáljuk ki a triggereket.

1. FELADAT / II.

```
CREATE OR REPLACE TRIGGER On_Logon
AFTER LOGON
ON felhasználónév.Schema
BEGIN
 INSERT INTO LOGON VALUES (USER || ' belepett', SYSDATE);
END;
/

CREATE OR REPLACE TRIGGER On_Logoff
BEFORE LOGOFF
ON felhasználónév.Schema
BEGIN
 INSERT INTO LOGON VALUES (USER || ' kilepett', SYSDATE);
END;
/
```

```
select nev, to_char(datum,'DD/MM/YYYY HH24:MI:SS') as "DATUM"
from logon;
```

2. FELADAT

- Hozzunk létre egy olyan utasításszintű BEFORE triggert, amely megakadályozza amunkaidőn kívüli adatmanipulációkat az emp táblán!
- Próbáljunk meg beszúrni egy sort!

```
CREATE OR REPLACE TRIGGER NeNyuljHozza
BEFORE DELETE OR INSERT OR UPDATE ON emp
BEGIN
 IF TO_CHAR(SYSDATE, 'HH24:MI') > '14:00'
 THEN
      RAISE_APPLICATION_ERROR(-20111, 'Csak
      munkaidoben szabad az adatbazis!');
 END IF;
END;
```

```
insert into emp values (1234, 'kiss', 'clerk', 1111,
'99-MAJ-20', 1200, NULL, 10);
```

3. FELADAT

- Oldjuk meg az előző feladatot oly módon, hogy azt is kiíratjuk, milyen műveletet kíséreltek meg végrehajtani munkaidőn kívül.

```
CREATE OR REPLACE TRIGGER NeNyuljHozza
BEFORE DELETE OR INSERT OR UPDATE ON emp
BEGIN
 IF TO_CHAR(SYSDATE, 'HH24:MI') > '14:00’ THEN
  IF DELETING THEN
   RAISE_APPLICATION_ERROR(-20211, 'Csak munkaidoben szabad
   adatot torolni!');
  ELSIF INSERTING THEN
   RAISE_APPLICATION_ERROR(-20212, 'Csak munkaidoben szabad
   adatot bevinni!');
  ELSE
   RAISE_APPLICATION_ERROR(-20213, 'Csak munkaidoben szabad
   adatot modositani!');
  END IF;
 END IF;
END;
```

```
DELETE FROM emp WHERE UPPER(ename) = UPPER('Smith');
```

4. FELADAT

- Írjunk triggert, mely megakadályozza az elnökre (president) vonatkozó törlő, beszúró ésadatmódosító utasítások működését!

```
CREATE OR REPLACE TRIGGER KiveveAzElnok
BEFORE DELETE OR INSERT OR UPDATE ON emp
FOR EACH ROW
WHEN (UPPER(OLD.job) = 'PRESIDENT'
OR UPPER(NEW.job) = 'PRESIDENT')
BEGIN
 IF DELETING THEN
  RAISE_APPLICATION_ERROR(-20001, 'Az elnokot nem lehet
  torolni!');
 ELSIF INSERTING THEN
  RAISE_APPLICATION_ERROR(-20002, 'Elnokot nem lehet
  beszurni!');
 ELSIF UPDATING THEN
  RAISE_APPLICATION_ERROR(-20003, 'Elnok adatok nem
  modosithatok!');
 END IF;
END;
```

```
UPDATE emp SET sal = sal - 555 WHERE deptno = 10;
insert into emp values (5678, 'kiss', 'president', NULL,
SYSDATE, 6000, NULL, 10)
```

TOVÁBBI FELADATOK

2. Hozzuk létre a RAPID RALLY autóversenyre benevezett versenyzők tábláját. Készítsünk triggert, amely minden beillesztett versenyzőhöz megkísérel alábbi feltételeknek megfelelő partnert keresni: 1) a partnerek nemzetisége egyezzen meg, 2) a partnerek minősítése legyen 20 ponton belül, 3) a feladatkörök illeszkedjenek. Tötse fel a partner oszlopokat(!) a feltételeknek megfelelően.
3. Hozzuk létre a ALVÓ LAJHÁR Sport Egyesület által szervezett bajnokság adatbázisát. Készítsünk triggert amelyik minden új résztvevő csapathoz legenerálja a lejátszandó mérkőzéseket a mérkőzések táblában. Minden csapat egy mérkőzést játszik az adott bajnokság többi csapatával.

TOVÁBBI FELADATOK

4. Partnerek(<u>név</u>, nem, életkor, város, partner_neve)
   - Új sor beszúrása esetén keressünk automatikusan partnert:
     - Különböző neműek
     - Max. 10 év korkülönbséggel
     - Ugyanabban a városban lakjanak
   - Egy emberhez egy párt társítunk, ha több is megfelelne, akkor a legkisebb korkülönbségűt válasszuk.
   - Beszúráskor partner_neve legyen '---'.