Projektovanje baza podataka - Okidači nad pogledima

Pogledi se obično koriste za razdvajanje logičke sheme od fizičke sheme. Ovakvo razdvajanje često ima za posledicu nemogućnosti izvršavanja DELETE i INSERT operacija, osim u slučaju jednostavnih pogleda. U tu svrhu se koriste INSTEAD OF okidači kojima se omogućavaju i ovakve operacije.

Potrebno je precizirati tri nivoa ažuriranja:

Brisanje: potrebno je izdvojiti vrstu koja se briše
Izmena: Potrebno je izdvojiti vrstu i kolonu koja se menja
Unos: Potrebno je izdvojiti novu vrstu i sve kolone koje pripadaju toj vrsti.

Definišimo sledeće tabele

```
CREATE TABLE T1(c1 INT, c2 FLOAT)$
INSERT INTO T1 VALUES (5, 6.0),
                      (6, 7.0),
                      (5, 6.0)$
CREATE TABLE T2(c1 INT, c2 FLOAT)$
INSERT INTO T2 VALUES (5, 9.0),
                      (5, 4.0),
                      (7, 5.0)$
```

Primer 1:

```
CREATE VIEW V1(c1)
AS SELECT c1 FROM T1 WHERE c2 > 0$
```

Ovaj pogled može da briše i menja podatke s obzirom na to da se može pronaći konkretna vrsta koja se ažurira.

```
DELETE FROM V1 WHERE c1 = 6$

UPDATE V1 SET c1 = c1 + 5 WHERE c1 = 5$
```

Međutim, ne mogu se unostiti podaci jer ovaj pogled ne zna za drugu kolonu u tabeli.

```
INSERT INTO V1 VALUES (8)$
```

Primer 2:

```
CREATE VIEW V2(c1, c2)
AS SELECT c1, c2 * c2 FROM T1$
```

U ovom primeru brisanje je moguće, pošto se može izdvojiti vrsta na osnovu polja c1. Menjanje nije moguće zato što se možda ne može izračunati c2.

Unos je moguć ukoliko se za ostale kolone koje nisu obuhvaćene pogledom može postaviti NULL ili

neka podrazumevana vrednost.

```
INSERT INTO V2(c1) VALUES (7)$
```

Sistem će uneti insert (7, NULL) u tabelu T1.

Primer 3:

```
CREATE VIEW V3(c1, c2, c3)
AS SELECT T1.c1, T1.c2, T2.c2
     FROM T1, T2 WHERE T1.c1 = T2.c1$
```

Ovaj pogled je dobijen spajanjem. Rezultat je sledeći:

```
SELECT * FROM V3 ORDER BY c1, c2, c3$

C1  C2    C3
--  ---  ---
 5  6.0  4.0
 5  6.0  4.0
 5  6.0  9.0
 5  6.0  9.0
```

Preko ovog pogleda se ne mogu brisati kao ni menjati vrste u tabelama T1 i T2.

Primer 4:

```
CREATE VIEW V4(c1, c2)
AS SELECT c1, c2 FROM T1
   UNION ALL
   SELECT c1, c2 FROM T2$
```

Preko ovog pogleda se mogu brisati vrste, zato što svaka vrsta potiče iz tačno jedne tabele, ali se ne mogu dodavati, pošto nije jasno u koju tabelu se dodaje vrsta.

INSTEAD OF okidači

```
CREATE VIEW V7(c1, c2)
AS SELECT DISTINCT c1, c2 FROM T1$
```

INSTEAD OF okidač može biti definisan da obriše sve vrste koje zadovoljavaju uslov, ili pak samo neku po defnisanom pravilu.

Sledeći okidač briše sve vrste koje zadovoljavaju uslov

```
CREATE TRIGGER V7_DELETE INSTEAD OF DELETE ON V7
```

```
  REFERENCING OLD_TABLE AS OLD_TAB
  OLD AS O FOR EACH ROW MODE DB2SQL
  DELETE FROM T1 WHERE o.c1 = c1 AND o.c2 = c2$
```

Okidač se izvršava tek kada se pokuša brisanje vrsta nad pogledom. Tada se doslovce ne izvršava brisanje nad pogledom već se izvršava definisani INSTEAD OF okidač.

INSTEAD OF okidači se uvek definišu samo nad pogledima, nikada nad tabelama. Takođe, uvek sa FOR EACH ROW klauzom, što znači da se izvršavaju jednom za svaku vrstu koja zadovoljava uslov u skladu sa definicijom pogleda. Ovakvim okidačima se ne može zadati vreme izvršavanja BEFORE ili AFTER.

Primeri

Primer 1:

Definišimo okidač za inverznu operaciju enkriptovanja.

```
CREATE TABLE USERS
   (user VARCHAR(20),
    system VARCHAR(30),
    login VARCHAR(20),
    password VARCHAR(40) FOR BIT DATA)$
```

Tabela USERS čuva identifikatore i enkriptovane lozinke korisnika na različitim sistemima. Sledeći pogled dekriptuje lozinku za datog korisnika.

```
CREATE VIEW MY_LOGINS(system, login, password)
AS SELECT system, login, decrypt_char(password)
     FROM USERS AS u WHERE u.user = USER$
```

Za menjanje ili brisanje preko ovog pogleda, potrebno je definisati okidač koji enkriptuje lozinke dobijene od korisnika.

```
CREATE TRIGGER INSERT_MY_LOGINS INSTEAD OF INSERT
  ON MY_LOGINS REFERENCING NEW AS n
  FOR EACH ROW MODE DB2SQL
  INSERT INTO USERS
    VALUES(USER, n.system, n.login,
           encrypt(password))$

CREATE TRIGGER UPDATE_MY_LOGINS INSTEAD OF UPDATE
  ON MY_LOGINS REFERENCING OLD AS o NEW AS n
  FOR EACH ROW MODE DB2SQL
  UPDATE USERS U
    SET system = n.system,
        login = n.login,
        password = encrypt(n.password)
    WHERE system = o.system
      AND login = o.login
      AND U.user = USER$
```

Dodajmo informacije u tabelu

```
INSERT INTO MY_LOGINS
  VALUES('AFS',   'srielau', 'mydogsname'),
        ('Linux', 'root',    'oopsIforgot'),
        ('IIUG',  'Rielau',  '123456789')$

SELECT * FROM MY_LOGINS WHERE system = 'Linux'$

SYSTEM  LOGIN     PASSWORD
------  -------   ------------
Linux   root      oopsIforgot
```

Lozinka u tabeli je enkriptovana i ne može se napraviti inverzan postupak.

```
SELECT * FROM USERS U ORDER BY U.user, system, login$

USER     SYSTEM   LOGIN     PASSWORD
-------- -------  -------   -------------
SRIELAU  AFS      srielau   0x..........
SRIELAU  IIUG     Rielau    0x..........
SRIELAU  Linux    root      0x..........
```

Moguće je izmeniti lozinku ili bilo koje drugo polje u pogledu.

```
UPDATE MY_LOGINS
  SET password = 'mycatsname'
  WHERE system = 'AFS' AND login = 'srielau'$
```

Polja iz pogleda se mogu brisati i bez korišćenja okidača

```
DELETE FROM my_logins WHERE SYSTEM = 'AFS'$
```

Primer 2:

Menadžer može videti plate svojih zaposlenih, ali ne i zaposlenih ostalih menadžera, kao ni njihove plate. Napravimo shemu i okidače koji omogućavaju menadžerima da menjaju plate zaposlenima za koje su zaduženi.

```
CREATE TABLE PROFILES
(empid INT, name VARCHAR(20), sqlid VARCHAR(18),
 mgrid INT, salary DECIMAL(9,2), ismgr CHAR(1))$
```

Želimo da vidimo podatke iz hijerarhije od četiri nivoa. Mi smo menadžer drugog nivoa sa nazivom MySelf.

```
INSERT INTO PROFILES
VALUES(0001, 'SuperBoss',  'sboss', NULL, 500000, 'Y'),
      (1001, 'BigBoss',    'bboss', 0001, 200000, 'Y'),
      (1002, 'MySelf',     USER,    0001, 250000, 'Y'),
      (2001, 'FirstLine',  'fline', 1001, 100000, 'Y'),
```

```
      (2002, 'MiddleMen',  'mmen',  1001, 110000, 'Y'),
      (2003, 'Yeti',       'yeti',  1002, 90000,  'Y'),
      (2004, 'BigFoot',    'bfoot', 1002, 80000,  'N'),
      (3001, 'TinyToon',   'ttoon', 2001, 50000,  'N'),
      (3002, 'Mouse',      'Mouse', 2001, 40000,  'N'),
      (3003, 'Whatsisname','wname', 2002, 45000,  'N'),
      (3004, 'Hasnoclue',  'hclue', 2002, 38000,  'N'),
      (3005, 'Doesallwork','dwork', 2003, 15000,  'N')$
```

Sledeći pogled nam omogućava da vidimo plate svojih zaposlenih kao i svoju.

```
CREATE VIEW my_emps(empid, level, salary)
AS WITH rec(empid, level, salary)
         AS (SELECT empid, 0, salary FROM PROFILES
               WHERE sqlid = USER
             UNION ALL
             SELECT P.empid, level-1, P.salary
               FROM PROFILES P, REC R
               WHERE level > -100
                 AND R.empid = P.mgrid)
   SELECT empid, level, salary FROM rec$
```

Sledeći pogled omogućava svima da vide informacije o zaposlenma.

```
CREATE VIEW PROFILES_V(empid, name, mgrname,
                       salary, sqlid, ismgr)
  AS SELECT P.empid, P.name,
            (SELECT name FROM PROFILES M
              WHERE M.empid = P.mgrid),
            ME.salary, P.sqlid, P.ismgr
     FROM PROFILES P LEFT OUTER JOIN my_emps ME
       ON ME.empid = P.empid$
```

Preko ovog pogleda nije moguće menjati podatke, tako da definišemo okidač

```
CREATE TRIGGER INSERT_PROFILES_V
  INSTEAD OF INSERT ON PROFILES_V REFERENCING NEW AS n
  FOR EACH ROW MODE DB2SQL
  BEGIN ATOMIC
    DECLARE mgrid INT;
    DECLARE ismgr CHAR(1);
    SET (mgrid, ismgr)
      = (SELECT empid, ismgr FROM PROFILES
          WHERE name = n.mgrname);
    IF mgrid NOT IN (SELECT empid FROM my_emps)
       OR ismgr = 'N'
    THEN
      SIGNAL SQLSTATE '70000'
         SET MESSAGE_TEXT = 'Not Authorized!';
    END IF;
    INSERT INTO PROFILES
      VALUES(n.empid, n.name, n.sqlid,
             mgrid, n.salary, n.ismgr);
  END$
```

Okidač omogućava da menadžeri mogu dodati zaposlene samo u svojoj nadležnosti.

Sledeći okidač dozvoljava menjanje podataka samo svojim zaposlenim. Ukoliko se želi promeniti menadžer oba menadžera moraju biti u nadležnosti onog koji menja podatke.

```
CREATE TRIGGER UPDATE_PROFILES_V
  INSTEAD OF UPDATE ON PROFILES_V REFERENCING NEW AS n
  OLD AS o FOR EACH ROW MODE DB2SQL
  BEGIN ATOMIC
    DECLARE oldmgrid, newmgrid INT;
    SET oldmgrid = (SELECT empid FROM PROFILES
                    WHERE name = o.mgrname),
        newmgrid = (SELECT empid FROM PROFILES
                    WHERE name = n.mgrname);
    IF oldmgrid NOT IN (SELECT empid FROM my_emps)
       OR newmgrid NOT IN (SELECT empid FROM my_emps)
       OR o.empid = (SELECT empid FROM PROFILES P
                     WHERE USER = P.sqlid)
    THEN
      SIGNAL SQLSTATE '70000'
         SET MESSAGE_TEXT = 'Not Authorized!';
    END IF;
    UPDATE PROFILES SET empid = n.empid,
                        mgrid = newmgrid,
                        salary = n.salary,
                        sqlid = n.sqlid,
                        name = n.name,
                        ismgr = n.ismgr
     WHERE empid = o.empid;
  END$
```

Sledeći okidač dozvoljava brisanje samo onih zaposlenih za koje je zadužen menadžer, ali ne i samog sebe.

```
CREATE TRIGGER DELETE_PROFILES_V
  INSTEAD OF DELETE ON PROFILES_V
  REFERENCING OLD AS o FOR EACH ROW MODE DB2SQL
  BEGIN ATOMIC
    DECLARE mgrid INT;
    SET mgrid = (SELECT empid FROM PROFILES
                 WHERE name = o.mgrname);
    IF mgrid NOT IN (SELECT empid FROM my_emps)
       OR o.empid = (SELECT empid FROM PROFILES P
                     WHERE USER = P.sqlid)
    THEN
      SIGNAL SQLSTATE '70000'
         SET MESSAGE_TEXT = 'Not Authorized!';
    END IF;
    DELETE FROM PROFILES WHERE empid = o.empid;
  END$
```

Primer korišćenja:

```
SELECT * FROM PROFILES_V ORDER BY empid$

EMPID  NAME         MGRNAME    SALARY    SQLID    ISMGR
-----  ------------ ---------- --------- -------- -----
    1  SuperBoss    -                  - sboss    Y
 1001  BigBoss      SuperBoss          - bboss    Y
```

```
 1002  MySelf       SuperBoss  250000.00 SRIELAU  Y
 2001  FirstLine    BigBoss           - fline    Y
 2002  MiddleMen    BigBoss           - mmen     Y
 2003  Yeti         MySelf      90000.00 yeti     Y
 2004  BigFoot      MySelf      80000.00 bfoot    N
 3001  TinyToon     FirstLine         - ttoon    N
 3002  Mouse        FirstLine         - Mouse    N
 3003  Whatsisname  MiddleMen         - wname    N
 3004  Hasnoclue    MiddleMen         - hclue    N
 3005  Doesallwork  Yeti        15000.00 dwork    N
```

Zaposlimo NewGuy i postavimo da radi za Yeti.

```
INSERT INTO PROFILES_V
VALUES (3006, 'NewGuy', 'Yeti', 35000, 'nguy', 'N')$

SELECT * FROM PROFILES_V WHERE empid = 3006$

EMPID  NAME         MGRNAME    SALARY    SQLID    ISMGR
-----  ------------ ---------- --------- -------- -----
 3006  NewGuy       Yeti        35000.00 nguy     N
```

Promvišimo Doesallwork za menadžera, dodelimo mu menadžera Myself i postavimo mu povišicu za 30%.

```
UPDATE PROFILES_V
  SET ismgr = 'Y',
      salary = salary * 1.30,
      mgrname = 'MySelf'
  WHERE name = 'Doesallwork'$

SELECT * FROM PROFILES_V WHERE name = 'Doesallwork'$

EMPID  NAME         MGRNAME    SALARY    SQLID    ISMGR
-----  ------------ ---------- --------- -------- -----
 3005  Doesallwork  MySelf      19500.00 dwork    Y
```

Izbrišimo Doesallwork.

```
DELETE FROM PROFILES_V WHERE name = 'Doesallwork'$
```

Primer 3:

Vertikalno particionisanje podataka.

```
CREATE TABLE PERSONS(ssn INT NOT NULL,
                     name VARCHAR(20) NOT NULL)$

CREATE TABLE EMPLOYEES(ssn INT NOT NULL,
                       company VARCHAR(20) NOT NULL,
                       salary DECIMAL(9,2))$

CREATE TABLE STUDENTS(ssn INT NOT NULL,
                       university VARCHAR(20) NOT NULL,
                       major VARCHAR(10))$
```

Spajanje ovih tabela može biti zahtevno, tako da pravimo pogled

```
CREATE VIEW PERSONS_V(ssn, name, company,
                      salary, university, major)
AS SELECT P.ssn, name, company,
          salary, university, major
     FROM PERSONS P LEFT OUTER JOIN EMPLOYEES E
                         ON P.ssn = E.ssn
                    LEFT OUTER JOIN STUDENTS S
                         ON P.ssn = S.ssn$
```

Ovaj pogled nije moguće menjati, brisati kao ni unositi vrste. Stoga pišemo okidač.

```
CREATE TRIGGER DELETE_PERSONS_V
  INSTEAD OF DELETE ON PERSONS_V
  REFERENCING OLD AS o FOR EACH ROW MODE DB2SQL
  BEGIN ATOMIC
    DELETE FROM STUDENTS WHERE ssn = o.ssn;
    DELETE FROM EMPLOYEES WHERE ssn = o.ssn;
    DELETE FROM PERSONS WHERE ssn = o.ssn;
  END$
```

Prilikom unosa studenta, moramo ga pridružiti univerzitetu kao i zaposlenima. Stoga, ako osoba menja neko od ovih pridruživanja, vrste se moraju izbisati iz ili dodati u odgovarajuću tabelu.

```
CREATE TRIGGER UPDATE_PERSONS_V
  INSTEAD OF UPDATE ON PERSONS_V
  REFERENCING OLD AS o NEW AS n
  FOR EACH ROW MODE DB2SQL
  BEGIN ATOMIC
    UPDATE PERSONS
      SET (ssn, name) = (n.ssn, n.name)
      WHERE ssn = o.ssn;
    IF n.university IS NOT NULL
       AND o.university IS NOT NULL THEN
      UPDATE STUDENTS
        SET (ssn, university, major)
          = (n.ssn, n.university, n.major)
        WHERE ssn = o.ssn;
    ELSEIF n.university IS NULL THEN
      DELETE FROM STUDENTS WHERE ssn = o.ssn;
    ELSE
      INSERT INTO STUDENTS
        VALUES(n.ssn, n.university, n.major);
    END IF;
    IF n.company IS NOT NULL
       AND o.company IS NOT NULL THEN
      UPDATE EMPLOYEES
        SET (ssn, company, salary)
          = (n.ssn, n.company, n.salary)
        WHERE ssn = o.ssn;
    ELSEIF n.company IS NULL THEN
      DELETE FROM EMPLOYEES WHERE ssn = o.ssn;
    ELSE
      INSERT INTO EMPLOYEES
        VALUES(n.ssn, n.company, n.salary);
```

```
    END IF;
  END$
```

Primer korišćenja

```
INSERT INTO PERSONS_V VALUES
  (123456, 'Smith', NULL, NULL, NULL, NULL),
  (234567, 'Jones', 'Wmart', 20000, NULL, NULL),
  (345678, 'Miller', NULL, NULL, 'Harvard', 'Math'),
  (456789, 'McNuts', 'SelfEmp', 60000, 'UCLA', 'CS')$

SELECT * FROM PERSONS_V ORDER BY SSN$

SSN     NAME    COMPANY  SALARY    UNIVERSITY  MAJOR
------  ------  -------- --------  ----------- ------
123456  Smith   -               -  -           -
234567  Jones   Wmart    20000.00  -           -
345678  Miller  -               -  Harvard     Math
456789  McNuts  SelfEmp  60000.00  UCLA        CS
```

Mr. Smith menja prezime, i zapošljava se kod Mickburgs za platu 15000. Mr. Miller završava Harvard i zapošljava se u IBM.

```
UPDATE PERSONS_V SET (name, company, salary)
                   = ('Johnson', 'Mickburgs', 15000)
  WHERE SSN = 123456$

UPDATE PERSONS_V SET (company, salary, university)
                   = ('IBM', 70000, NULL)
  WHERE SSN = 345678$

SELECT * FROM PERSONS_V WHERE SSN IN (123456, 345678)
  ORDER BY SSN$

SSN     NAME     COMPANY  SALARY    UNIVERSITY  MAJOR
------  -------  -------- --------  ----------- ------
123456  Johnson  Mickburgs   15000  -           -
345678  Miller   IBM          70000  -           -
```

Mr. Jones napušta sve.

```
DELETE FROM PERSONS_V WHERE NAME = 'Jones'$
```

Primer 4:

Prilikom unosa podataka u tabelu, BEFORE okidači i uslovi ograničenja proveravaju da li su podaci koji se unose ispravni, ali nemaju načina da ih isprave, već samo dozvoljavaju ili odbijaju unos.

Sledeći primer preusmerava loše podatke u drugu tabelu.

```
CREATE TABLE ADDRESSES(name varchar(10),
                       number INT,
                       street varchar(20),
                       country VARCHAR(10)
                               WITH DEFAULT 'CANADA')$
```

```
CREATE TABLE BAD_ADDRESSES
AS (SELECT CAST(NULL AS VARCHAR(30)) AS Reason, A.*
    FROM ADDRESSES A) DEFINITION ONLY$

CREATE VIEW ADDRESSES_V
AS SELECT * FROM ADDRESSES$
CREATE TRIGGER INSERT_ADDRESSES_V
  INSTEAD OF INSERT ON ADDRESSES_V
  REFERENCING NEW AS n FOR EACH ROW MODE DB2SQL
  BEGIN ATOMIC
    DECLARE reason VARCHAR(30);
    SET reason
      = CASE WHEN n.number IS NULL OR n.number <= 0
             THEN 'Number'
            WHEN n.name IS NULL OR LENGTH(n.name) = 0
             THEN 'Name'
            WHEN n.street IS NULL
              OR LENGTH(n.street) = 0
             THEN 'Street'
            WHEN country IS NULL
              OR country NOT IN ('CANADA', 'USA',
                                 'GERMANY', 'FRANCE')
             THEN 'Country'
            ELSE NULL END;
    IF reason IS NOT NULL THEN
      INSERT INTO BAD_ADDRESSES
        VALUES(reason, n.name, n.number,
               n.street, n.country);
    ELSE
      INSERT INTO ADDRESSES
        VALUES(n.name, n.number, n.street, n.country);
    END IF;
  END$
```

U sledećem primeru, ukoliko država nije navedena, ili je korišćena ključna reč DEFAULT, INSTEAD OF okidač će videti CANADA. Za sve ostale kolone okidač će videti NULL.

```
INSERT INTO ADDRESSES_V VALUES
 ('Jones', 510, 'Yonge St.', DEFAULT),
 ('Smith', -1, 'Nowhere', 'USA'),
 (NULL, 38, 'Am Feldweg', 'GERMANY'),
 ('Poubelle', 23, 'Rue de Jardin', 'FRANCE')$

SELECT * FROM ADDRESSES ORDER BY name$

NAME       NUMBER  STREET         COUNTRY
---------- ------  -------------- -------
Jones         510  Yonge St.      CANADA
Poubelle       23  Rue de Jardin  FRANCE

SELECT * FROM BAD_ADDRESSES ORDER BY name$

REASON  NAME    NUMBER  STREET      COUNTRY
------  ------- ------  ----------- -------
Number  Smith       -1  Nowhere     USA
Name    -           38  Am Feldweg  GERMANY
```

Okidač takođe može biti podešen da ignorše loše podatke umesto da ih smešta u tabelu.