software

Alen Prodan

Edition Based Redefinition – Visoka dostupnost kod nadogradnje aplikacija

software

Agenda

- Upravljanje edicijama
- Upravljanje editioning viewovima
- Upravljanje crossedition triggerima
- Primjer nadogradnje aplikacijske sheme
- Pitanja i odgovori

software

Upravljanje edicijama

Što su edicije ?

- Edicije su objekti unutar baze podataka koji ne pripadaju niti jednoj pojedinoj shemi
- Kreirane su u jedinstvenom prostoru imena (namespace)
- Unutar baze podataka može postojati istovremeno više edicija
- Svaka baza podataka mora imati najmanje jednu ediciju (ORA$BASE)

software

Upravljanje edicijama

Koji se objekti mogu pridružiti edicijama ?

- Razlikujemo edicijske objekte (editioned objects) od ne-edicijskih objekata (noneditioned objects)
- Potencijalni objekt za verzioniranje predstavlja edicijski objekt koji se nalazi u shemi kojoj nije omogućeno upravljanje edicijama
- Edicija ne može imati svoju vlastitu kopiju ne-edicijskog objekta – oni su identični i na podjednak način vidljivi u svim edicijama

software

Upravljanje edicijama

Koji se objekti mogu pridružiti edicijama ?

- Tipovi objekata koji podržavaju edicije:
  - **Funkcije (FUNCTION)**
  - **Biblioteke (LIBRARY)**
  - **Paketi (PACKAGE i PACKAGE BODY)**
  - **Procedure (PROCEDURE)**
  - **Sinonimi (SYNONYMS) – ali ne i javni sinonimi (Public Synonyms) !!!**
  - **Okidači (TRIGGER)**
  - **Tipovi (TYPE i TYPE BODY)**
  - **Pregled (VIEW)**

- Tipovi objekata koji su vezani uz Javu, kao i fizički segmenti na disku (tablice, indeksi, materijalizirani viewovi) NE podržavaju edicije.

software

Novosti u generiranju SQL trace dijagnostičkih informacija

Korištenje edicija unutar aplikacijske sheme

- Prvi korak u korištenju edition-based mehanizma je aktiviranje editioning mehanizma (operacija nije reverzibilna – ne postoji DISABLE EDITIONS naredba !!!)

```
SQL> alter user test enable editions;

User altered.

SQL> select username, editions_enabled
  2  from dba_users
  3  where username = 'TEST';

USERNAME                       E
------------------------------ -
TEST                           Y
```

software

Novosti u generiranju SQL trace dijagnostičkih informacija

Kreiranje edicija, nasljeđivanje i aktualizacija objekata

- Svaka baza podataka mora imati barem jednu ediciju (ORA$BASE)
- Edicije su organizirane u roditelj-dijete hijerarhiji, uz ograničenje da svaka edicija može imati samo jedno dijete

```
SQL> create edition e1 as child of ora$base;
SQL> create edition e2 as child of e1;
```

- Korisnik koji je kreirao ediciju posjeduje USE ON EDITION privilegiju sa GRANT opcijom. I drugim korisnicima je moguće omogućiti korištenje edicija pojedinačno ili pak grantanjem PUBLIC shemi:

```
SQL> grant use on edition e1 to test;
SQL> grant use on edition e2 to test;

ili

SQL> grant use on edition e1 to public;
```

Novosti u generiranju SQL trace dijagnostičkih informacija

Kreiranje edicija, nasljeđivanje i aktualizacija objekata

- Prilikom kreiranja, child edicija nasljeđuje od svoje roditeljske edicije sve edicijske objekte koji su vidljivi parent ediciji
- Nasljeđeni objekti su samo vidljivi, ali NE pripadaju child ediciji, sve do trenutka AKTUALIZACIJE
- Aktualizacija se događa u trenutku kada korisnik u child ediciji koristeći DDL naredbu izmjeni ili kompilira nasljeđeni objekt

software

Novosti u generiranju SQL trace dijagnostičkih informacija

Kreiranje edicija, nasljeđivanje i aktualizacija objekata

- Primjer aktualizacije, kreiramo proceduru TEST u ediciji E1:

```
SQL> create edition e1 as child of ora$base;   ← kreiramo ediciju E1
Edition created.

SQL> create edition e2 as child of e1;   ← kreiramo ediciju E2, child od E1
Edition created.

SQL> alter session set edition = e1;   ← postavimo E1 kao tekuću ediciju

SQL> create or replace procedure test ← kreiramo objekt (proceduru) TEST
2 is
3 begin
4 dbms_output.put_line('Edicija E1');
5 end;
Procedure created.

SQL> set serveroutput on size 100000 ← izvršimo proceduru test
SQL> exec test;
Edicija E1   ← output je očekivano string “Edicija E1”

PL/SQL procedure successfully completed.
```

Novosti u generiranju SQL trace dijagnostičkih informacija

Kreiranje edicija, nasljeđivanje i aktualizacija objekata

- Primjer aktualizacije: postavimo E2 kao tekuću ediciju

```
SQL> alter session set edition = e2;          ← E2 postaje tekuća edicija
Session altered.

SQL> exec test;                               ← output je string “Edicija E1”
Edicija E1                                    ← E2 trenutno vidi kopiju originala iz E1
PL/SQL procedure successfully completed.
```

software

Novosti u generiranju SQL trace dijagnostičkih informacija

Kreiranje edicija, nasljeđivanje i aktualizacija objekata

- Primjer aktualizacije: rekreiranjem objekta vršimo aktualizaciju

```
SQL> create or replace procedure test   ← ovo je trenutak aktualizacije
2 is
3 begin
4 dbms_output.put_line('Edicija E2');
5 end;
6 /
Procedure created.

SQL> exec test;
Edicija E2   ← očekivano, output se mijenja iz “Edicija E1” u “Edicija E2”

PL/SQL procedure successfully completed.
```

software

Novosti u generiranju SQL trace dijagnostičkih informacija

Kreiranje edicija, nasljeđivanje i aktualizacija objekata

- Provjera da zaista imamo vlastitu kopiju TEST procedure u E1 i E2 ediciji:

```
SQL> select object_name, object_type, edition_name
2 from user_objects_ae
3 /

OBJECT_NAME                    OBJECT_TYPE         EDITION_NAME
------------------------------ ------------------- --------------
TEST                           NON-EXISTENT(???)   ORA$BASE
TEST                           PROCEDURE           E1
TEST                           PROCEDURE           E2
```

software

Novosti u generiranju SQL trace dijagnostičkih informacija

Kreiranje edicija, nasljeđivanje i aktualizacija objekata

- Svaka sesija može istovremeno koristiti samo jednu ediciju. Ediciju koju sesija koristi nazivamo tekućom edicijom (current edition), a ujedno predstavlja i ediciju sesije (session edition):

```
SQL> select sys_context('USERENV', 'SESSION_EDITION_NAME') from dual;

SYS_CONTEXT('USERENV','SESSION_EDITION_NAME')
--------------------------------------------------------------------------------
E1

SQL> select sys_context('USERENV', 'CURRENT_EDITION_NAME') from dual;

SYS_CONTEXT('USERENV','CURRENT_EDITION_NAME')
--------------------------------------------------------------------------------
E1

ili
SQL> show edition

EDITION
------------------------------
ORA$BASE
```

software

Novosti u generiranju SQL trace dijagnostičkih informacija

Kreiranje edicija, nasljeđivanje i aktualizacija objekata

- Na razini sesije, ukoliko nije posebno izabrana, koristi se osnovna database edicija (default database edition) ORA$BASE, može se promijeniti uz pomoć ALTER DATABASE naredbe:

```
SQL> alter database default edition = e2;

SQL> select property_name, property_value
2 from database_properties
3 where property_name = 'DEFAULT_EDITION';

PROPERTY_NAME                  PROPERTY_VALUE
------------------------------ --------------------
DEFAULT_EDITION                E2
```

software

Novosti u generiranju SQL trace dijagnostičkih informacija

Upravljanje editioning viewovima

- Tablice spadaju u skupinu ne-edicijskih objekata i stoga ih nije moguće direktno redefinirati
- Umjesto toga, za svaku je tablicu moguće kreirati editioning view
- U aplikaciji zamjenjujemo direktno referenciranje tablica sa referencama prema editioning viewovima
- Editioning viewovi podržavaju sve vrste triggera osim INSTEAD OF i CROSSEDITION TRIGGERA, za razliku od običnih viewova
- Nije moguće kreirati direktno constrainte i indekse nad njima već nad pripadajućim tablicama
- Editioning viewovima eksponiramo podskup kolona iz pripadajućih tablica, te definiramo zamjenske nazive kolona (alias) – stvaramo privid da su i same fizičke tablice redefinirane.

Novosti u generiranju SQL trace dijagnostičkih informacija

Upravljanje editioning viewovima

- Primjer definiranja editioning viewa sa WITH READ ONLY klauzulom:

```
create editioning view test_ev
as
select id, naziv from tab
with read only;
```

- Za vrijeme nadogranje aplikacije, editioning viewovi mogu dozvoljavati samo čitanje ili pak i ažuriranje podataka
- Jednostavnija je varijanta ako je pristup isključivo read-only
- Crossedition triggere koristimo kada je za vrijeme nadogranje aplikacije potrebno omogućiti istovremene izmjene podataka

software

Novosti u generiranju SQL trace dijagnostičkih informacija

Upravljanje crossedition triggerima

- Crossedition triggere koristimo u slučaju kada su ispunjena dva uvjeta:
  - Struktura tablica mora biti redefinirana za vrijeme nadogranje aplikacije
  - Nije moguće privremeno onemogućiti ažuriranje podataka (read only)
- Forward Crossedition triggerom se transformira redak iz stare strukture retka u novu strukturu
- Reverse Crossedition triggerom se transformira redak iz nove u staru strukturu retka
- Rezultat: korisnici mogu istovremeno ažurirati podatke za vrijeme trajanja nadogradnje aplikacije koristeći obje edicije aplikacije
- Transformaciju već pohranjenih podataka u tablici možemo na najlakši način izvesti izvršavanjem crossedition triggera i to za svaki redak, koristeći DBMS_SQL.PARSE funkciju sa APPLY_CROSSEDITION_TRIGGER parametrom

software

Primjer nadogradnje aplikacijske sheme

- Tablica sadrži telefonski imenik sa ukupno 5 slogova (E1):

```
create table imenik
(
id number,
naziv varchar2(20),
telefon varchar2(15)
);

SQL> select * from imenik;

ID          NAZIV                TELEFON
---------- -------------------- ---------------
1           ivan ivić            051/111-2222
2           pero perić           051/222-3333
3           jurica jurić         051/333-4444
4           mate matić           051/444-5555
5           luka lukić           051/555-6666
```

software

Primjer nadogradnje aplikacijske sheme

- Tablicu želimo redefinirati na način da odvojimo predbroj od telefonskog broja korisnika (E1):

```
←Prvi korak, moramo preimenovati tablicu:
alter table imenik rename to imenik_tab;

← Drugi korak, moramo kreirati editioning view:
create editioning view imenik
as
select id, naziv as ime_prezime, telefon
from imenik_tab;

← Kreiranje nove edicije E2
create edition e2 as child of e1;

← Postavljanje edicije E2 kao tekuće edicije u sesiji:
alter session set edition = e2;

← Redefinicija tablice
alter table imenik_tab add (predbroj varchar2(3), tel_broj varchar2(9));
```

software

Primjer nadogradnje aplikacijske sheme

- Tablicu želimo redefinirati na način da odvojimo predbroj od telefonskog broja korisnika (E2):

← Nakon izmjene fizičkih struktura IMENIK_TAB tablice, u novoj ediciji E2 moramo redefinirati IMENIK editioning view:

```
create or replace editioning view imenik
as
select id, naziv as ime_prezime, predbroj, tel_broj
from imenik_tab;
```

← Izvršimo li select iz IMENIK viewa u novoj ediciji vidimo da kolone predbroj i tel_broj imaju vrijednost NULL u sebi:

```
SQL> select * from imenik;

ID         IME_PREZIME          PRE TEL_BROJ
---------- -------------------- --- ---------
1          ivan ivić
2          pero perić
3          jurica jurić
4          mate matić
5          luka lukić
```

← kolone su prazne jer još uvijek nisu definirana pravila transformacije

software

Primjer nadogradnje aplikacijske sheme

- Tablicu želimo redefinirati na način da odvojimo predbroj od telefonskog broja korisnika:

← definiranje pravila transformacije putem forward crossedition triggera:

```
create trigger imenik_fw_xed_trig
before insert or update on imenik_tab
for each row
FORWARD CROSSEDITION
disable
BEGIN
 :new.predbroj := substr(:new.telefon, 1, 3);
 :new.tel_broj := substr(:new.telefon, 5);
END;
```

← Trigger je inicijalno kreiran u DISABLE statusu, kako ne bi utjecao na dostupnost aplikacije:

```
alter trigger imenik_fw_xed_trig enable;
```

software

Primjer nadogradnje aplikacijske sheme

- Tablicu želimo redefinirati na način da odvojimo predbroj od telefonskog broja korisnika:

←Transformacija strukture svih postojećih redaka u tablici. Prvo slijedi provjera da li ima otvorenih transakcija nad našom tablicom:

```
declare
 r boolean;
 scn number;
begin
 r := dbms_utility.wait_on_pending_dml(
  tables=>'IMENIK_TAB',
  timeout=>null,
  scn=>scn);
 if (r) then
  dbms_output.put_line('Ok, sve transakcije potvrdjene.');
 else
  dbms_output.put_line('Postoje otvorene transakcije.');
 end if;
end;
/
```

software

Primjer nadogradnje aplikacijske sheme

- Tablicu želimo redefinirati na način da odvojimo predbroj od telefonskog broja korisnika:

← Pozivom na dbms_sql.parse funkciju vršimo samu transformaciju, navodeći naziv crossedition triggera u apply_crossedition_trigger parametru:

```
declare
 cur number;
 nrows number;
begin
 cur := dbms_sql.open_cursor;
 dbms_sql.parse(
  c => cur,
  statement => 'UPDATE imenik SET id = id',
  language_flag => dbms_sql.native,
  apply_crossedition_trigger=> 'imenik_fw_xed_trig',
  fire_apply_trigger => true);
 nrows := dbms_sql.execute(cur);
 dbms_sql.close_cursor(cur);
 commit;
end;
```

software

Primjer nadogradnje aplikacijske sheme

- Tablicu želimo redefinirati na način da odvojimo predbroj od telefonskog broja korisnika:

← Po završetku transformacije, možemo ponoviti naš upit nad IMENIK viewom i provjeriti da li smo postigli željeni rezultat:

```
SQL> select * from imenik;

ID         IME_PREZIME          PRE TEL_BROJ
---------- -------------------- --- ---------
1          ivan ivić            051 111-2222   ← transformacija uspješno izvedena !!!
2          pero perić           051 222-3333
3          jurica jurić         051 333-4444
4          mate matić           051 444-5555
5          luka lukić           051 555-6666
```

software

Primjer nadogradnje aplikacijske sheme

- Tablicu želimo redefinirati na način da odvojimo predbroj od telefonskog broja korisnika:

← Naš test ne bi bio potpun kada ne bi testirali transformaciju u oba smjera: iz stare u novu ediciju (forward crossedition trigger), te iz nove u staru (reverse crossedition trigger):

```
create trigger imenik_rv_xed_trig
before insert or update on imenik_tab
for each row
REVERSE CROSSEDITION
Disable
BEGIN
 :new.telefon := :new.predbroj||'/'||:new.tel_broj;
END;

alter trigger imenik_rv_xed_trig enable;
```

software

Primjer nadogradnje aplikacijske sheme

- Tablicu želimo redefinirati na način da odvojimo predbroj od telefonskog broja korisnika:

← Prvo ćemo izvršiti insert u novoj ediciji, kako bi testirali prethodno kreirani reverse crossedition trigger:

```
insert into imenik (id, ime_prezime, predbroj, tel_broj)
            values (100, 'testni korisnik', '051', '123-4567');

commit;
```

← U drugoj sesiji, vraćamo se u staru ediciju E1, te provjeravamo da li je korektno prikazan netom insertirani redak:

```
SQL> select * from imenik;

ID         IME_PREZIME          TELEFON
---------- -------------------- ---------------
1          ivan ivić            051/111-2222
2          pero perić           051/222-3333
3          jurica jurić         051/333-4444
4          mate matić           051/444-5555
5          luka lukić           051/555-6666
100        testni korisnik      051/123-4567  ← redak je ispravno transformiran
```

software

Primjer nadogradnje aplikacijske sheme

- Tablicu želimo redefinirati na način da odvojimo predbroj od telefonskog broja korisnika:

← Unutar te iste sesije sa tekućom edicijom E1, insertirati ćemo još jedan redak, kako bi testirali forward crossedition trigger:

```
insert into imenik (id, ime_prezime, telefon)
            values (101, 'testni korisnik2', '051/765-4321');

commit;
```

← U drugoj sesiji, sa tekućom edicijom E2, provjeravamo da li je netom inserirani redak transformiran putem forward crossedition triggera:

```
alter session set edition = e2;

SQL> select * from imenik where id = 101;

ID         IME_PREZIME          PRE TEL_BROJ
---------- -------------------- --- ---------
101        testni korisnik2     051 765-4321        ← redak je ispravno transformiran
```

software

Zaključak

- U Oracle Database 11g Release 2, postoje dva ozbiljna ograničenja koja se odnose na hijerahije edicija:
  - **Prvo ograničenje odnosi se na činjenicu da svaka edicija može imati samo jednog nasljednika**
  - **Drugo ograničenje odnosi se na činjenicu da baza podataka može imati samo jednu izvornu (root) ediciju**
- Zbog mogućnosti konsolidacije više različitih aplikacija unutar iste baze podataka, bilo bi poželjno kada bi svaka aplikacija mogla imatu svoju vlastitu hijerarhiju verzija

software

Pitanja i Odgovori

software