4

Nastavení databáze pro použití Flashbacku

Ve verzi Oracle 9i byla představena nová vlastnost databáze, a to **flashback**. K jejímu razantnímu zdokonalení došlo s verzí 10g a ve verzi 11g se již hojně využívá. Pojmenování plně odpovídá účelu, pro který byla vytvořena. Z praxe, ale i z příkladů, které jste si zde zkoušeli, jste jistě pochopili, jak náročné na čas je dělat obnovu do času v minulosti, a to i v případě, že se jedná o banální smazání tabulky. Důvod je jednoduchý. Při klasické obnově se vždy obnoví databáze ze zálohy a teprve potom se aplikují veškeré změny provedené v databázi až po nějak určený konečný bod. Konečný čas může být časová značka, SCN číslo, číslo transakce atd. Při obnově se využívají archivní logy a pak redo logy. Postup je to časově náročný, protože musíte obnovit celou databázi a poté aplikovat všechny změny z archivního logu, online redo lodů a UNDO segmentů. Při smazání tabulky například musíte udělat PITR celého tabulkového prostoru někde mimo a poté třeba exportem a importem dostat tabulku zpět.

Naproti tomu technologie flashback používá své flashback logy, undo segmentů a odpadkového koše. Jednoduše ji lze přirovnat k tlačítku Undo ve vašem textovém editoru. Namísto toho, abyste nastalý problém řešili a snažili se ho odstranit, použijete akci „krok zpět".

Veškeré akce a změny provedené v databázi jsou označeny svým číslem SCN (system change number). V případě, že budete chtít změnu vrátit, udělat tzv. **rollback**, musíte říci, po které SCN číslo se chcete vrátit. Flashback můžete použít na veškeré tabulkové prostory vyjma systémového.

Flashback je rozdělen do různých oblastí podle akce, kterou chcete provádět. Zde je seznam a v příkladech si vyzkoušíte některé z nich.

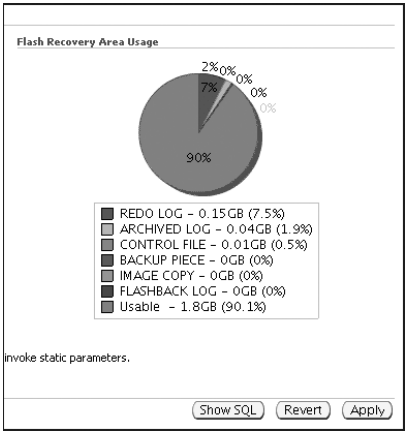
- **Oracle Flashback Query**
- **Oracle Flashback Version Query**
- **Flashback Transaction Query**
- **Flashback Transaction**
- **Flashback Data Archive (Oracle Total Recall)**
- **Oracle Flashback Table**
- **Oracle Flashback Drop**
- **Oracle Flashback Database**

Abyste mohli flashback využívat, musíte ho nejdříve zapnout. Aktivovat flashback můžete rovnou při vytváření databáze pomoci **dbca** nebo kdykoli později. Lze to udělat také později přímo v sqlplus nebo pomocí EM. Podmínkou je, aby databáze běžela v **archivním módu**. Dalším parametrem, který je potřeba nastavit, je parametr **DB_FLASHBACK_RETENTION_TARGET**, který udává v minutách, jak dlouho si má databáze pamatovat provedené změny. To znamená, jak hluboko do minulosti chcete být schopni udělat flashback. Nakonec danou vlastnost aktivujete příkazem **alter database flashback on**.

Přihlaste se do EM a v hlavní obrazovce uvidíte informaci, zdali je flashback zapnutý nebo vypnutý.

Obrázek 4.1 Aktuální stav Flashbacku

V databázi není flashback aktivován. Aktivovat jej můžete buď přímo v EM klepnutím na odkaz **Disabled**, nebo to můžete udělat v sqlplus. Záleží na vás, jaká cesta bude pro vás přijatelnější. Každopádně klepnete-li na odkaz, uvidíte i aktuální obsazení flash recovery area.



Obrázek 4.2 Obsazení flash recovery area

Poznámka: Aktuální stav můžete také získat výpisem parametru **flashback_on**.

```
SQL> select flashback_on,log_mode from
v$database;
FLASHBACK_ON       LOG_MODE
------------------ ------------
YES                ARCHIVELOG
```

Flash recovery area je oblast, kam se ukládají flashback logy.

Flashback se zapíná nebo vypíná v mount režimu, je tedy nutné databázi nejdříve zavřít a otevřít v mount režimu.

1. Nastartujte databázi v mount režimu.

```
SQL> shutdown immediate;
ORACLE instance shut down.
SQL> startup mount
Database mounted.
```

Zkontrolujte, zda je databáze v archivním módu. Pokud není, zapněte ho pomocí příkazu **alter database archivelog**.

```
SQL> archive log list;
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
```

Zkontrolujte, zda máte nastavené parametry pro recovery. V případě, že nejsou nastaveny, nastavte je pomocí příkazu **alter system**.

```
SQL> show parameter recovery
NAME                                 TYPE        VALUE
------------------------------------ ----------- ----------------
db_recovery_file_dest                string      +DATA
db_recovery_file_dest_size           big integer 2G
```

Nakonec nastavte parametr potřebný pro Flashback a aktivujte ho. Hodnota 1440 říká, že informace o prováděných změnách se budou pamatovat 1 den. Nakonec databázi otevřete.

```
SQL> alter system set DB_FLASHBACK_RETENTION_TARGET=1440;
System altered.
```

2. Zapněte flashback v databázi.

```
SQL> alter database flashback on;
Database altered.
```

3. Otevřete databázi.

```
SQL> alter database open;
Database altered.
```

V alert.logu je informace o zapnutí flashbacku a první SCN číslo, ke kterému se můžete vrátit.

```
Flashback Database Enabled
Turn database flashback on at SCN 2362030
```

Tímto máte Flashback nastavený a v příkladech si vyzkoušíte, kdy a jak ho můžete využít.

Dokumentace:

- http://download.oracle.com/docs/cd/B28359_01/appdev.111/b28424/adfns_flashback.htm#insertedID1.

Obnova smazané tabulky pomocí Flashback Table

Flashback budete často využívat při nechtěném smazání tabulky. Při obnově můžete použít EM nebo sqlplus.

1. V příkladu si vytvořte libovolnou tabulku pod svým uživatelem a naplňte ji daty.

```
SQL> create table seznam(id number(2),jmeno varchar2(30));
```

```
Table created.
SQL> insert into seznam values (1, 'Jaroslava Solarova');
...
SQL> select * from seznam;
        ID JMENO
---------- ------------------------------
         1 Jaroslava Solarova
         2 Tomas Solar
         3 Matej Solar
         4 Martin Komberec
```

2. Nyní tabulku smažte a podívejte se, co se stalo.

```
SQL> drop table seznam;
Table dropped.
```

Při zapnutém flashbacku nedochází k fyzickému smazání objektu, nýbrž objekt je jen přejmenován a uložen do koše (recycle bin). Jeho jméno je nahrazeno jiným začínajícím **BIN$** a pokračuje hashem. Můžete se na tabulku podívat. Zároveň uvidíte i přesný čas, kdy byla smazaná.

```
SQL> select object_name, original_name, droptime from recyclebin;
OBJECT_NAME                    ORIGINAL_NAME   DROPTIME
------------------------------------------------------------------------------
BIN$eH7dNKyOJJ7gQEAKWwFH7A==$0      SEZNAM          2009-11-16:16:42:59
```

Tabulku také můžete prohlížet.

```
SQL> select * from "BIN$eH7dNKyOJJ7gQEAKWwFH7A==$0";

        ID JMENO
---------- ------------------------------
         1 Jaroslava Solarova
         2 Tomas Solar
         3 Matej Solar
         4 Martin Komberec
```

Poznámka: Nezapomeňte na dvojité uvozovky, protože ve jménu tabulky jsou znaky jako rovná se.

K vytažení tabulky z koše slouží příkaz **flashback table <jméno tabulky> to before drop**.

3. Obnovte tabulku z koše.

```
SQL> flashback table seznam to before drop;
Flashback complete.
```

Podíváte-li se nyní do koše, tabulku tam nenajdete. Naopak je plně funkční a plná dat a prohlížet si ji můžete opět pod původním jménem.

```
SQL> select * from seznam;
        ID JMENO
---------- ------------------------------
         1 Jaroslava Solarova
```

```
2 Tomas Solar
3 Matej Solar
4 Martin Komberec
```

Jednoduchý příklad, který plně ilustroval použití Flashbacku v praxi.

Metalink:

- 265254.1 Flashback Table Feature in Oracle Database 10g.

Obnova pomocí Flashback Table – rozšíření

V předchozím příkladě jste si vyzkoušeli flashback tabulky. Co ale v případě, že jste danou tabulku vytvořili a smazali vícekrát. Například ve Windows máte různé verze smazaných souborů. V Oracle tomu není jinak a v koši se uchovávají všechny verze smazaných objektů. Zkuste si nyní vytvořit tabulku a opět smazat dvakrát a podívejte se, co se uchovává v koši.

Poznámka: Koš si můžete vysypat pomocí příkazu **purge recyclebin**.

```
SQL> purge recyclebin;
Recyclebin purged.
```

Vytvořte si tabulku, ve které budete mít různé hodnoty v každé verzi. Já jsem zvolil například tabulku **poradi**, kde v první verzi bude hodnota **prvni** a v druhé instanci bude hodnota **druha**.

```
SQL> create table poradi(poradi varchar2(30));
SQL> insert into poradi values ('prvni');
SQL> drop table poradi;
SQL> create table poradi(poradi varchar2(30));
SQL> insert into poradi values ('druha');
SQL> drop table poradi;
```

Při pohledu do koše zjistíte, že se v něm uložily dvě verze jedné tabulky: stejné původní jméno, jen jiný čas smazání.

```
SQL>  select object_name, original_name, droptime from recyclebin;
OBJECT_NAME                          ORIGINAL_NAME   DROPTIME
-------------------------------------------------------------------------------
BIN$eH7dNKyRJJ7gQEAKWwFH7A==$0       PORADI          2009-11-16:17:06:44
BIN$eH7dNKyQJJ7gQEAKWwFH7A==$0       PORADI          2009-11-16:17:06:18
```

V případě, že byste provedli flashback tabulky jako v předchozím případě, Oracle automaticky bere tu poslední verzi, která byla smazaná.

```
SQL> flashback table poradi to before drop;
Flashback complete.

SQL> select * from poradi;
PORADI
------------------------------
Druha
```

Opět tabulku smažu. V případě, že chcete získat verzi, která byla smazána dříve, musíte ji specifikovat jménem.

```
SQL> flashback table "BIN$eH7dNKyQJJ7gQEAKWwFH7A==$0" to before drop;
Flashback complete.

SQL> select * from poradi;
PORADI
------------------------------
Prvni
```

V případě, že na tabulku jsou navázané nějaké závislé objekty, jako třeba index, je potřeba být obezřetný. Do koše se uloží i závislé objekty a při obnově se také obnoví, ale zůstane jim jméno, které jim bylo přiřazeno v koši, proto je potřeba obnovený index přejmenovat.

```
SQL> create index seznam_idx on seznam(id);
SQL> drop table seznam;
SQL> select object_name, original_name,droptime from recyclebin;
OBJECT_NAME                    ORIGINAL_NAME   DROPTIME
-------------------------------------------------------------------
BIN$eH7dNKyTJJ7gQEAKWwFH7A==$0   SEZNAM_IDX    2009-11-16:17:19:55
BIN$eH7dNKyUJJ7gQEAKWwFH7A==$0   SEZNAM        2009-11-16:17:19:55
```

Zkuste tabulku obnovit a zkontrolujte jména, která byla jí a indexu přiřazena.

```
SQL> flashback table seznam to before drop;
Flashback complete.

SQL> select index_name from user_indexes where table_name='SEZNAM';
INDEX_NAME
------------------------------
BIN$eH7dNKyTJJ7gQEAKWwFH7A==$0
```

Nezbývá vám tedy než index přejmenovat. Naštěstí to je jen jeden příkaz, ale musíte na to pamatovat.

```
SQL> alter index "BIN$eH7dNKyTJJ7gQEAKWwFH7A==$0" rename to seznam_idx;
Index altered.

SQL> select index_name from user_indexes where table_name='SEZNAM';
INDEX_NAME
------------------------------
SEZNAM_IDX
```

Poznámka: Samozřejmě můžete použít flashback i na tabulku, která existuje a změnila se v ní data. Musíte však znát SCN číslo, kdy v tabulce byla správná data. Pak použijete příkaz **flashback table <jméno tabulky> to scn <SCN číslo>.**

Obnova celé databáze pomocí Flashback databáze

Předchozí příklady byly zaměřené na tabulku a její závislé objekty. Dovedete si představit situaci v produkčním prostředí, kdy smažete více nezávislých objektů nebo celé schéma. Zde pak přichází na řadu flashback celé databáze. U velkých databází, kde by obnova ze zálohy trvala

dlouho, je to jediné rozumné řešení, jak obnovit data s minimálním úsilím a maximálním užitkem. **Flashback database** je alternativa k nekompletní obnově tzv. Point-in-time recovery. V následujícím řešení si zkusíte udělat flashback po smazání tabulkového prostoru. Flashback tabulky provádí buď uživatel sys, nebo častěji přímo vlastník daného objektu. U flashbacku databáze není možné, aby ho prováděl někdo jiný než uživatelé s oprávněním SYSDBA, protože je zde nutné databázi zavřít, flashback provést v **mount** režimu a opět otevřít.

Představte si, že máte uživatele a nechtěně jste ho smazali společně se všemi jeho objekty. Jedno řešení je udělat obnovu databáze nebo použít flashback, který je daleko rychlejší.

1. Smažte uživatele tsolar.

```
SQL> drop user tsolar cascade;
User dropped.

SQL> select USERNAME from dba_users where username='TSOLAR';
no rows selected
```

   Jak jsem podotkl dříve, flashback databáze se provádí v mount režimu. Zastavte databázi a otevřete ji v mount režimu, abyste měli jistotu, že jste k databázi připojeni pouze vy, a přidejte klausuli **exclusive**.

2. Nastartujte databázi v režimu mount exclusive.

```
SQL> shutdown immediate;
SQL> startup mount exclusive
...
Database mounted.
```

3. Proveďte flashaback databáze 10 minut zpět, protože víte, že před deseti minutami uživatel ještě existoval. Nakonec databázi otevřete s parametrem **resetlogs**.

```
SQL> flashback database to timestamp sysdate-(1/24/6);
Flashback complete.

SQL> alter database open resetlogs;
Database altered.
```

Neznáte-li čas, můžete použít další alternativy pro specifikaci minulosti.

- **flashback database to TIMESTAMP <datum>;**
- **flashback database to BEFORE TIMESTAMP <datum>;**
- **flashback database to SCN <scn cislo>;**
- **flashback database to BEFORE SCN <scn cislo>;**

Veškerá data jsou opět dostupná a v čase nesrovnatelně kratším než při běžné obnově ze záloh.

```
SQL> connect tsolar/oracle
Connected.
SQL> select * from seznam;
        ID JMENO
---------- ------------------------------
         1 Jaroslava Solarova
         2 Tomas Solar
```

```
         3 Matej Solar
         4 Martin Komberec
```

Poznámka: Aktuální SCN číslo můžete získat z dotazu do pohledu v$database.

```
SQL> select current_scn, TO_CHAR(SYSTIMESTAMP, 'YYYY-MM-DD HH24:MI:SS') from
v$database;

CURRENT_SCN TO_CHAR(SYSTIMESTAM
----------- -------------------
    2376090 2009-11-16 18:52:03
```

Metalink:

- 356337.1 Flashback Database For Drop User.
- 834824.1 Recovering Dropped User using Flashback Database.

Hledání dat z minulosti pomocí Flashback query

Flashback query je další způsob, jak získat data z minulosti. Tento způsob vám umožní vytvořit dotaz, kde specifikujete čas nebo SCN číslo v minulosti. Podívejte se na následující řešení. Představte si tabulku, kde vkládáte postupně různá data, ale najednou je něco špatně a vy potřebujete zjistit, jaká data tam byla před nějakým časem.

Budu pracovat s tabulkou seznam, kterou jsem si připravil dříve. Tato tabulka obsahuje nějaká data.

```
SQL> select * from seznam;
        ID JMENO
---------- ------------------------------
         1 Jaroslava Solarova
         2 Tomas Solar
```

Tato data jsou platná teď. Současný stav můžete definovat buď časem, nebo hodnotou SCN čísla.

```
SQL> select current_scn, TO_CHAR(SYSTIMESTAMP, 'YYYY-MM-DD HH24:MI:SS') from
v$database;

CURRENT_SCN TO_CHAR(SYSTIMESTAM
----------- -------------------
    2409819 2009-11-17 17:28:59
```

Když provedete změnu v tabulce, například smažete jeden záznam, automaticky se změní hodnota SCN čísla a samozřejmě i čas.

```
SQL> delete seznam where id=2;
1 row deleted.

SQL> commit;
Commit complete.
```

Aktuální data jsou nyní následující.

```
SQL> select * from seznam;
```

```
        ID JMENO
---------- ------------------------------
         1 Jaroslava Solarova
```

Když víte čas, kdy byla data platná, můžete jednoduše udělat dotaz na data v konkrétním čase. Pomocí flashback definujete minulost časem nebo SCN číslem, které uvedete za klauzuli **AS OF**.

```
SQL> select * from seznam AS OF TIMESTAMP TO_TIMESTAMP('2009-11-17 17:28:59',
'YYYY-MM-DD HH24:MI:SS');
        ID JMENO
---------- ------------------------------
         1 Jaroslava Solarova
         2 Tomas Solar
```

Můžete také specifikovat SCN číslo.

```
SQL> select * from seznam AS OF SCN 2409819;
        ID JMENO
---------- ------------------------------
         1 Jaroslava Solarova
         2 Tomas Solar
```

Metalink:

- 317499.1 10G Oracle Flashback Transaction Query – Introduction and usage.

Procházení historických dat pomocí Flashback version query

Flashback version query je doplnění funkcionality obyčejného flashback query tím, že máte možnost sledovat průběh změn v čase pomocí klauzule **VERSIONS BETWEEN**.

Vyzkoušejte si to na příkladu. Použijeme tabulku seznam, kde máte nyní jeden řádek.

```
SQL> select * from seznam;
        ID JMENO
---------- ------------------------------
         1 Jaroslava Solarova
```

Tyto informace jsou platné pro toto SCN a čas.

Soubor 03_01.txt

```
SQL> select current_scn, TO_CHAR(SYSTIMESTAMP, 'YYYY-MM-DD HH24:MI:SS') from
v$database;
CURRENT_SCN TO_CHAR(SYSTIMESTAM
----------- -------------------
    2410841 2009-11-17 17:42:29
```

Dejme tomu, že se paní Solařová rozvedla a změnila si jméno a opět se provdala; budete proto měnit hodnotu jména.

```
SQL> update seznam set jmeno = 'Jaroslava Boraltova' where id=1;
1 row updated.

SQL> commit;
```

```
Commit complete.

SQL> update seznam set jmeno = 'Jaroslava Vankova' where id=1;
1 row updated.

SQL> commit;
Commit complete.

SQL> select current_scn, TO_CHAR(SYSTIMESTAMP, 'YYYY-MM-DD HH24:MI:SS') from
v$database;
CURRENT_SCN TO_CHAR(SYSTIMESTAM
----------- -------------------
    2410904 2009-11-17 17:43:47
```

Pro výpis historie změny záznamů pak použijte dotaz, kde stanovíte rozmezí, které vás zajímá, ať časově nebo specifikací SCN čísla.

Příklad použití času:

Soubor 03_02.txt

```
select versions_startscn, versions_starttime,
       versions_endscn, versions_endtime,
       versions_xid, versions_operation,jmeno
from   seznam
       VERSIONS BETWEEN TIMESTAMP TO_TIMESTAMP('2009-11-17 17:42:29', 'YYYY-MM-
DD HH24:MI:SS')
       AND TO_TIMESTAMP('2009-11-17 17:43:47', 'YYYY-MM-DD HH24:MI:SS')
where  id = 1;
```

Příklad použití SCN čísla:

Soubor 03_03.txt

```
select versions_startscn, versions_starttime,
       versions_endscn, versions_endtime,
       versions_xid, versions_operation,jmeno
from seznam VERSIONS BETWEEN SCN 2410841 AND 2410904 where id=1;
```

Jelikož jsou oba výpisy hodně široké, protože obsahují časové značky, kdy byl který záznam změněn, nebudu je zde uvádět celé, ale důležité jsou poslední dva sloupce, ve kterých jsou jednoznačně určené transakce, které změnily daný řádek. Tyto identifikátory pak můžete využít u další vlastnosti flashbacku, tzv. **Flashback transaction query**.

```
SQL> select  versions_xid, versions_operation,jmeno
from seznam VERSIONS BETWEEN SCN 2410841 AND 2410904 where id=1;
VERSIONS_XID     V JMENO
---------------- - --------------------
0A001A0006070000 U Jaroslava Vankova
060015000D070000 U Jaroslava Boraltova
                   Jaroslava Solarova
```

Sledování transakce pomocí Flashback transaction query

Flashback transaction query se používá, potřebujete-li zjistit konkrétní informace o konkrétní transakci. Z předchozího příkladu jste získali čísla dvou transakcí, které můžete podrobně zkoumat. Stačí k tomu jen dotaz do tabulky **flashback_transaction_query**, která má následující definici:

```
SQL> desc flashback_transaction_query
 Name                                      Null?    Type
 ----------------------------------------- -------- -------------
 XID                                                RAW(8)
 START_SCN                                          NUMBER
 START_TIMESTAMP                                    DATE
 COMMIT_SCN                                         NUMBER
 COMMIT_TIMESTAMP                                   DATE
 LOGON_USER                                         VARCHAR2(30)
 UNDO_CHANGE#                                       NUMBER
 OPERATION                                          VARCHAR2(32)
 TABLE_NAME                                         VARCHAR2(256)
 TABLE_OWNER                                        VARCHAR2(32)
 ROW_ID                                             VARCHAR2(19)
 UNDO_SQL                                           VARCHAR2(4000)
```

Kompletní informace o transakci pak získáte následujícím dotazem:

Soubor 03_04.txt

```
SQL> select xid, operation, start_scn,commit_scn, logon_user, undo_sql from
flashback_transaction_query
where  xid = HEXTORAW('0A001A0006070000');

XID              OPERATION                        START_SCN COMMIT_SCN
---------------- -------------------------------- ---------- ----------
LOGON_USER
------------------------------
UNDO_SQL
--------------------------------------------------------------------------------
0A001A0006070000 UPDATE                             2410894    2410895
TSOLAR
update "TSOLAR"."SEZNAM" set "JMENO" = 'Jaroslava Boraltova' where ROWID = 'AAAT
RuAAGAAABA/AAA';

0A001A0006070000 BEGIN                              2410894    2410895
TSOLAR

XID              OPERATION                        START_SCN COMMIT_SCN
---------------- -------------------------------- ---------- ----------
```

```
LOGON_USER
------------------------------
UNDO_SQL
------------------------------------------------------------------
```

Povšimněte si klauzule where, kde pracujete s číslem transakce získaným v předcházejícím příkladě. Zároveň zde vidíte vygenerovaný SQL příkaz, kterým, když jej spustíte, vrátíte hodnotu ve sloupci jméno na předcházející.

```
SQL> select * from seznam;
        ID JMENO
---------- ------------------------------
         1 Jaroslava Vankova

SQL> update "TSOLAR"."SEZNAM" set "JMENO" = 'Jaroslava Boraltova' where ROWID =
'AAATRuAAGAAABA/AAA';
1 row updated.

SQL> commit;
Commit complete.

SQL> select * from seznam;
        ID JMENO
---------- ------------------------------
         1 Jaroslava Boraltova
```

Metalink:

- 270270.1 Flashback Version Query & Flashback Transaction Query – Oracle 10G Enhancement.

Uchování historie změn v tabulce pomocí Flashback data archive (11g)

Ve verzi Oracle 11g byla představena nová vlastnost **Flashback data archive**. Také se používá označení **Oracle total recall**. V předchozích příkladech jste si vyzkoušeli, na jakém principu se s flashbackem pracuje, co je k tomu potřeba a jaké chyby s ním lze napravit. Jak bylo řečeno, oproti klasické obnově ze zálohy se při flashbacku používá UNDO tabulkový prostor a pomocí něho lze změny vrátit zpět. Sami si ale dovedete představit, že tímto způsobem můžete zpracovávat a opravovat data ne moc stará. Většinou se jedná o řády hodin, maximálně dnů. Flashback data archive rozšiřuje tuto možnost a umožňuje vracet se zpět k datům v řádu měsíců či roků. Celý princip je založen na vytvoření archivu, do kterého se ukládají veškeré změny pro tabulky, u nichž je nastaveno, že mají daný archiv využívat. Flashback data archive nelze zapnout na celou databázi, ale jen na úrovni tabulek. Flashback archiv vytvoříte podobně jako tabulku specifikujete mu nějaké tabulkové prostory a nastavíte mu garanční dobu, tedy čas, po který se v něm mají data uchovávat. Pak je dobré nastavit i kvótu, aby nenarostl do velkých rozměrů. Tedy flashaback data archive umožňuje přístup k historickým datům. Než si o tom dlouze povídat, pojďte si vyřešit nějaký praktický příklad.

Nejdříve si vytvořte archiv. K tomu slouží příkaz **create flashback archive**. Naopak **drop flashback archive** ho smaže.

```
SQL> create flashback archive default flash1 tablespace flashback_tbs quota 2G
retention 1 year;


Flashback archive created.
```

Poznámka: Parametr default označí daný archiv jako předdefinovaný pro všechny tabulky.

Poznámka: Tabulkový prostor flashback_tbs musí být prázdný.

Vytvořili jste archiv se jménem flash1. Ten bude používat tabulkový prostor flashback_tbs, který může být maximálně 2 GB velký a bude uchovávat data tabulek minimálně 1 rok.

Informace o nastavení flashback data archive získáte z pohledu to tabulky **dba_flashback_archive**.

```
SQL> SELECT flashback_archive_name, retention_in_days, status
FROM   dba_flashback_archive;
FLASHBACK_ARCHIVE_NAME        RETENTION_IN_DAYS       STATUS
------------------------------------------------------------------------------
FLASH1                        365                     DEFAULT
```

Poznámka: Veškeré parametry můžete měnit libovolně za běhu databáze pomocí příkazu **alter flashback archive**.

Následně pak můžete vytvořit tabulku s parametrem FLASHBACK ARCHIVE. Od té doby se budou veškeré změny zaznamenávat i do archivu pro případ, že byste k nim chtěli v budoucnu přistupovat.

```
SQL> create table dulezita_data (id number,jmeno varchar2(3),bonus varchar2(30))
FLASHBACK ARCHIVE;


Table created.
```

Zapnout archivování flashback dat na stavající tabulce můžete příkazem **alter table <jmeno_tabulky> flashback archive**.

Získání historických dat je pak možné dotazem, kde specifikujete čas v minulosti jako interval. Příklad níže vypíše hodnotu bonusu před dvěma hodinami:

```
select bonus from dulezita_data AS OF TIMESTAMP (sysdate -interval '120' minute);
```

Interval můžete definovat i v jiných jednotkách, jako jsou dny, měsíce, roky:

- sysdate - interval '50' second
- sysdate - interval '7' days
- sysdate - interval '11' months

Reportování dat je pak stejné jako v případě Flaschback query, tedy pomocí parametru **VERSIONS BETWEEN TIMESTAMP**.

Poznámka: Aktuální čas zjistíte dotazem z tabulky dual.

```
SQL> select to_timestamp(sysdate) from dual;
TO_TIMESTAMP(SYSDATE)
---------------------
17-NOV-09 12.00.00 AM
```

Poznámka: Chcete-li tabulku, nad kterou je zapnuté flashback archivování, smazat, musíte nejdřív archivování vypnout, jinak dostanete při mazání chybu, jakou vidíte níže.

```
SQL> drop table dulezita_data;
drop table dulezita_data
           *
ERROR at line 1:
ORA-55610: Invalid DDL statement on history-tracked table

SQL> alter table dulezita_data no flashback archive;
Table altered.

SQL> drop table dulezita_data;
Table dropped.
```

Metalink:

- 470199.1 11g feature: Flashback Data Archive Guide.

Obnova transakce pomocí Flashback Transaction s použitím Logmineru (11g)

Ve verzi Oracle 11g je možné použít flashback pro obnovu jednotlivých transakcí. Používá při tom velmi známou a mocnou utilitu **LogMiner**. Jistě dobře víte, že logminer se používá při čtení redo logů. V redo lozích se uchovávají informace o veškerých změnách provedených v databázi, tudíž se zde otevírá možnost pro celou řadu operací. Pomocí logmineru můžete dohledat sql příkazy, které provedly nežádoucí změnu, a na jejich základě můžete vytvořit opačné příkazy, které provedou opravu. Pomocí logmineru můžete dohledat časové značky, uživatelská jména, typy akcí, které byly v databázi provedeny, jako jsou insert, update, delete. Můžete získat čísla transakcí, jména tabulek a samozřejmě rekonstruovat sqlp dotazy. Abyste byli schopni propojit čísla objektů s aktuálními jmény objektů, je potřeba Logmineru umožnit přístup k datům v datovém slovníku. Veškeré informace, které si logminer přečte, ukládá do pohledu V**$LOGMNR_CONTENS**.

Abyste mohli LogMiner použít v plné síle, je potřeba nastavit dodatečné logování informací do redo logů, tzv. **supplemental logging**. Ten může být nastavený buď na úrovni celé databáze, nebo pro jednotlivé tabulky. Ačkoli Logminer vyžaduje minimálně zapnutý supplemetal logging, Flashback transaction potřebuje mít zapnuté logování na úrovni primárního klíče. Obojí logování se zapíná příkazem **alter database add supplementa log data**. Přihlaste se jako uživatel sys a zapněte si dodatečné logování.

```
SQL> alter database add supplemental log data;
Database altered.
```

```
SQL> alter database add supplemental log data (primary key) columns;
Database altered.
```

Využiji uživatele tsolar, který je v databázi již vytvořen, a vytvořím pod ním novou tabulku, na které si vyzkoušíte, jakým způsobem Flashback transaction pracuje.

```
SQL> create table test_tab (id NUMBER,popis VARCHAR2(50), CONSTRAINT test_tab_pk
PRIMARY KEY (id));

Table created.
```

Proveďte nějaké operace v tabulce:

```
SQL> insert into test_tab (id, popis) VALUES (1, 'Prvni akce');
1 row created.

SQL> commit;
Commit complete.

SQL> insert INTO test_tab (id, popis) VALUES (2, 'Druhy akce');
1 row created.

SQL> commit;
Commit complete.
SQL> update test_tab set popis = 'Treti akce' where  id = 1;
1 row updated.

SQL> commit;
Commit complete.

SQL> update test_tab set popis = 'Ctvrta akce' where  id = 2;
1 row updated.

SQL> commit;
Commit complete.

SQL> delete from test_tab where id = 1;
1 row deleted.

SQL> commit;
Commit complete.
```

K práci s LogMinerem můžete použít balík, který slouží k jeho obsluze (**DBMS_LOGMN**), anebo můžete využít EM, kam byl LogMiner ve verzi 11g implementován.

Přihlaste se do EM a v záložce **Availability** v sekci **Manage** je odkaz **View and manage transactions**. Klepněte na něj a dostanete se do hlavní stránky LogMineru.

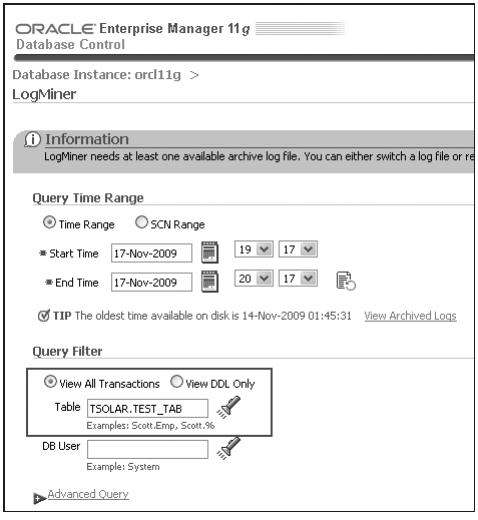
Obrázek 4.3 Odkaz na LogMiner z hlavní stránky EM

Dostali jste se do hlavní stránky LogMineru. Zde si vyberte svého uživatele a tabulku, kterou jste vytvořili a ve které jste prováděli své transakce. Nechte si zobrazit všechny transakce pro uživatele **tsolar** a tabulku **test_tab**. Jakmile máte vybraného uživatele, stiskněte tlačítko **Continue**.

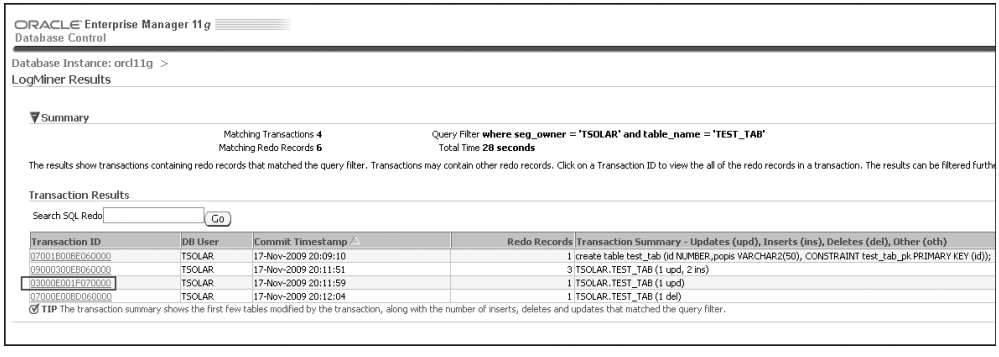
LogMiner vás nepustí dál v případě, že nemáte žádný archivní log, ze kterého by mohl číst, proto pod uživatelem SYS vygenerujte archivní log tak, že uložíte informace ze stávajícího redo logu do archivního logu. To provedete příkazem **alter system switch logfile**.

```
SQL> alter system switch logfile;
System altered.
```

Jakmile jste to provedli, stiskněte tlačítko Continue ještě jednou. Zobrazí se vám veškeré transakce, které daný uživatel provedl.

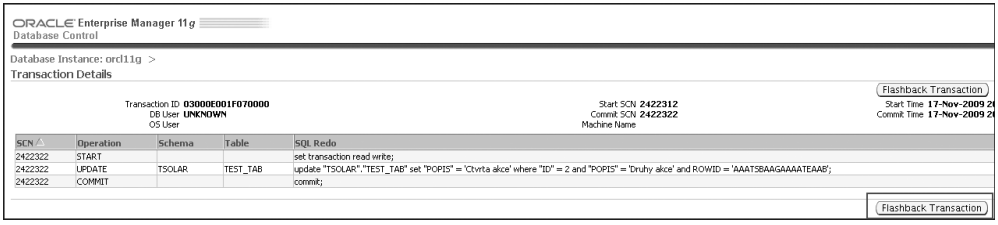


Obrázek 4.4 Výběr uživatele, pro kterého chcete zobrazit transakce



Obrázek 4.5 Seznam všech transakcí, které uživatel tsolar provedl

Vyberte si nyní jednu transakci a klepněte na její odkaz. Zobrazí se vám detailní popis dané transakce.



Obrázek 4.6 Detailní popis transakce

Všimněte si tlačítka **Flashback transaction**. Jakmile ho stisknete, provedete flashback a daná transakce se odroluje zpět. Zároveň se zkontrolují veškeré závislosti do jiných tabulek.

Graficky máte znázorněné veškeré závislé objekty a je jen na vás rozhodnout se, zda chcete provést odrolování nebo ne.

V alert.logu najdete informace o prováděné akci podobné těmto:

```
LOGMINER: Begin mining logfile for session -2147482623 thread 1 sequence 3,
/u01/app/oracle/product/11.1.0/db_1/dbs/arch1_3_703104102.dbf
LOGMINER: End mining logfile for session -2147481343 thread 1 sequence 3, /u01/
app/oracle/product/11.1.0/db_1/dbs/arch1_3_703104102.dbf
```

Metalink:

- 737332.1 Flashback transactions using dbms_flashback.transaction_backout procedure.
- 565535.1 Flashback Database Best Practices & Performance.