Oracle data Sluething with SQL

Presented by Corvin Deus

Developer at Harmons City Inc.

- Feel free to stop me with questions.
- If you have a question raise your hand, if I don't see you, feel free to yell out "I have a question".
- Try to be respectful and considerate of others in the
- In most of the examples, I changed my table names to "EMP"

- Quick survey:
  - Raise your hand if you are a DBA, Developer, Analyst.
  - Raise your hand if you have ever used flashback.

- This presentation will show real life examples of the need to know what happened to data and how it was determined who did it and when. It will go over simple techniques to test, compare and move data. Specifically, it will show how to use Oracle flashback, some psuedo columns, key underlying database tables, and examples of database links. Attendees should leave with a better understanding of data diagnostics and testing techniques along with SQL queries that they can use.
- http://www.hark.com/clips/sltdrqzfdh-flash-1

What is Oracle Flashback?

- **Flashback** provides ways to view past states of database objects, or to return database objects to a previous state, without using traditional point-in-time recovery.
- Flashback is for the oh shhh-crap moments.
- Flashback features of the database can be used to:
  - Perform queries that return past data.
  - Perform queries that return metadata showing a detailed history of changes to the database.
  - Recover tables or individual rows to a previous point in time.
- Flashback features use the *Automatic Undo Management* system to obtain metadata and historical data for transactions. Undo data is persistent and survives a database malfunction or shutdown.

Things to know about Flashback

1. Oracle introduced flashback features in Oracle 9i to address simple data recovery needs.
2. Make sure your undo retention is set to keep undo long enough for your purposes.
3. The duration available to Flashback is dependent on how the parameter DB_FLASHBACK_RETENTION_TARGET. The default (in 10G) is 1440 minutes = 24 hours.
4. Query your undo setting: Select * from v$parameter where Upper(name) = "DB_FLASHBACK_RETENTION_TARGET'
5. Be aware that truncates and other DDL will cause a 'break' and you will not be able to query before that.
6. We don't see all versions, only committed ones - A flashback version is created whenever a COMMIT statement is executed.
7. Flashback query allows a user to view the data quickly and easily the way it was at a particular time in the past, even when it is modified and committed, be it a single row or the whole table.
8. I had a hard time canceling long running flashback queries, so run them first with out the flashback.
9. The ability to use these examples depends on if the DB has been setup to use it and what privileges you have as a user.
10. **Make sure you understand it before using recovery techniques**.

Flashback query examples:

Use: AS OF TIMESTAMP or
VERSIONS BETWEEN or
AS OF SCN

- SELECT * FROM EMP AS OF TIMESTAMP to_Date('28-JAN-2014 08:50:58','DD-MON-YYYY HH24:MI:SS')
- SELECT * FROM EMP VERSIONS BETWEEN TIMESTAMP to_Date('28-JAN-2014 08:50:58','DD-MON-YYYY HH24:MI:SS') and SYSDATE
- SELECT * FROM EMP AS OF TIMESTAMP (SYSTIMESTAMP - INTERVAL '30' MINUTE) -- Not commonly used
- SELECT * FROM EMP AS OF SCN 10280403339) -- Not commonly used
- Where the current SCN is stored

  select current_scn from V$database

Flashback Restore Examples

- To return the whole table back to its original state:

  Before the drop

  - FLASHBACK TABLE EMP TO BEFORE DROP;

  To a point in time

  - FLASHBACK TABLE EMP AS OF TIMESTAMP to_Date('28-JAN-2014 08:50:58','DD-MON-YYYY HH24:MI:SS') ;

Other flashback applications

- INSERT INTO EMP (SELECT * FROM EMP AS OF TIMESTAMP to_Date('28-JAN-2014 08:50:58','DD-MON-YYYY HH24:MI:SS')

OR

- INSERT INTO EMP (SELECT * FROM EMP AS OF SCN 10280403339);

- CREATE TABLE EMP_BACKUP AS SELECT * FROM EMP AS OF TIMESTAMP to_Date('28-JAN-2014 08:50:58','DD-MON-YYYY HH24:MI:SS')

```
select *  --basic query (no flashback)
from EMP
where pin_number = '000030623'
```

INSERT_DATE	LAST_UPDATE_DATE
9/15/2013 3:00:42 AM	1/29/2014 3:00:32 AM

```
select *  -- with Flashback
from EMP VERSIONS BETWEEN TIMESTAMP to_Date('28-JAN-2014 08:50:58','DD-MON-YYYY HH24:MI:SS')  and SYSDATE
where pin_number = '000030623'
```

INSERT_DATE	LAST_UPDATE_DATE
9/15/2013 3:00:42 AM	1/29/2014 3:00:32 AM
9/15/2013 3:00:42 AM	9/15/2013 3:00:42 AM

```
select 'Current' Record_type, e.*  -- with dynamic record type column
from EMP     e
where pin_number = '000030623'
UNION ALL
select 'Flash' Record_type,  e2.insert_date,  e2.last_update_date
from EMP AS OF TIMESTAMP to_Date('28-JAN-2014 08:50:58','DD-MON-YYYY HH24:MI:SS') e2
where pin_number = '000030623'
```

RECORD_TYPE	INSERT_DATE	LAST_UPDATE_DATE
Current	9/15/2013 3:00:42 AM	1/29/2014 3:00:32 AM
Flash	9/15/2013 3:00:42 AM	9/15/2013 3:00:42 AM

Add this column to help distinquish the two records

Flashback PSEUDO columns

```
-- Find info with PSEUDO columns – (similar to sequence.CURRVAL sequence.NEXTVAL)
select i.invoice_id, i.invoice_num,
  versions_operation,
  versions_starttime,
  versions_endtime,
  versions_xid
from ap_invoices_interface
versions between TIMESTAMP to_Date('24-SEP-2013 7:20:05', 'DD-MON-YYYY HH24:MI:SS') and sysdate i
where  invoice_num like 'Oct13 010118%'
--and versions_operation = 'D'
order by versions_starttime
```

Table Alias (i) goes after the flashback timestamps

I could just look for deleted records

INVOICE_ID	INVOICE_NUM	VERSIONS_OPERATION	VERSIONS_STARTTIME	VERSIONS_ENDTIME	VERSIONS_XID
4633	Oct13 010118	I	9/24/2013 4:10:17. PM	9/26/2013 7:59:29. AM	27002100E12E0000
4629	Oct13 010118	I	9/24/2013 4:10:17. PM	9/26/2013 7:53:56. AM	27002100E12E0000
4629	Oct13 010118	D	9/26/2013 7:53:56. AM		17000B00164F0000
4633	Oct13 010118	U	9/26/2013 7:59:29. AM	9/26/2013 7:59:34. AM	0B000F002A380000
4633	Oct13 010118	U	9/26/2013 7:59:34. AM	9/26/2013 8:05:35. AM	07000D00754E0000
4633	Oct13 010118	U	9/26/2013 8:05:35. AM	9/26/2013 8:05:35. AM	1B001600C7370000
4633	Oct13 010118	D	9/26/2013 8:05:35. AM		02001E0026410000
4633	Oct13 010118	U	9/26/2013 8:05:35. AM	9/26/2013 8:05:35. AM	01001500DB430000

The query pseudo columns are valid only in Oracle Flashback Version Query (does not work with “AS OF TIMESTAMP”).

Versions_operation column

I = Insert, D = Delete, U = Update

VERSIONS_STARTSCN: Returns the SCN of the first version of the rows returned by the query.

VERSIONS_ENDSCN: Returns the SCN of the last version of the rows returned by the query.

VERSIONS_XID: For each version of each row, returns the transaction ID (a RAW number) of the transaction that created that row version.

If VERSIONS_STARTIME is NULL, then the row version was created before the lower time bound of the query BETWEEN clause.

```
SQL> Select versions_operation,
  2    versions_starttime,
  3    versions_endtime
  4  from AP_INVOICES_ALL  AS OF TIMESTAMP to_Date('29-JAN-2014 08:50:58','DD-MON-YYYY HH24:MI:SS')
  5  WHERE invoice_id = 4852072
  6  order by versions_endtime
  7  /
versions_endtime
*
ERROR at line 3:
ORA-00904: "VERSIONS_ENDTIME": invalid identifier


SQL> Select versions_operation,
  2    versions_starttime,
  3    versions_endtime
  4  from AP_INVOICES_ALL  VERSIONS BETWEEN TIMESTAMP to_Date('29-JAN-2014 08:50:58','DD-MON-YYYY HH24:MI:SS')
                                                  and to_Date('30-JAN-2014 08:50:58','DD-MON-YYYY HH24:MI:SS')
  5  WHERE invoice_id = 4852072
  6  order by versions_endtime
  7  /
```

Why doesn't the first query work and the second one does?

- Answer:

Because the PSEUDO columns are only part of the the *flashback version query*.

Must use "VERSIONS BETWEEN TIMESTAMP"

Use versions_xid column for additional info

```
Select versions_xid -- The xid to use in the flashback_transaction_query view
from AP_INVOICES_ALL VERSIONS BETWEEN TIMESTAMP to_Date('29-JAN-2014 08:50:58','DD-MON-YYYY HH24:MI:SS') and to_Date('30-JAN-2014 08:50:58','DD-MON-YYYY HH24:MI:SS')
WHERE invoice_id = 4852072


--Pass the version_xid into xid to get additional information about the txn
SELECT logon_user, operation, start_timestamp, undo_sql, start_scn, commit_scn
FROM flashback_transaction_query  -- This is a view
WHERE xid = HEXTORAW('0E00160011C40000') -- convert to take advantage of the index


--OR just use
SELECT logon_user, operation, start_timestamp, undo_sql, start_scn, commit_scn
FROM flashback_transaction_query
WHERE xid IN (Select versions_xid from AP_INVOICES_ALL VERSIONS BETWEEN TIMESTAMP
                to_Date('29-JAN-2014 08:50:58','DD-MON-YYYY HH24:MI:SS')
              and to_Date('30-JAN-2014 08:50:58','DD-MON-YYYY HH24:MI:SS'))
```

Sleuthing - Customer case:

Accounts Payable Clerk calls wondering why invoice ID = "4852072" is missing. Let's find what happened with just the invoice_id and approximate time.

```
--Start with the flashback
Select versions_operation,
  versions_starttime,
  versions_endtime
from AP_INVOICES_ALL VERSIONS BETWEEN TIMESTAMP to_Date('29-JAN-2014 08:50:58','DD-MON-YYYY HH24:MI:SS') and SYSDATE
WHERE invoice_id = 4852072
order by versions_endtime
```

VERSIONS_OPERATION	VERSIONS_STARTTIME	VERSIONS_ENDTIME
I	1/29/2014 2:35:37. PM	1/29/2014 2:35:37. PM
U	1/29/2014 2:35:37. PM	1/29/2014 2:35:46. PM
U	1/29/2014 2:35:46. PM	1/29/2014 2:35:52. PM
U	1/29/2014 2:35:52. PM	1/29/2014 2:36:37. PM
D	1/29/2014 2:36:37. PM	

```
-- Looking for a delete to AP_INVOICES_ALL
-- Using the date from the flashback, I use that for the last_active_time range
Select sql_id, sql_text, module, action, last_active_time
from v$sqlarea v
where upper(sql_text) like '%DELETE FROM AP_INVOICES_ALL%'
and command_type = '7' -- For deleted txns
and last_active_time
between to_Date('29-JAN-2014 14:36:00','DD-MON-YYYY HH24:MI:SS')
      and to_Date('29-JAN-2014 14:36:59','DD-MON-YYYY HH24:MI:SS')
```

I can see that the payables manager deleted the record with the "APXINWKB" module and the time it was deleted.

SQL_ID	SQL_TEXT	MODULE	ACTION	LAST_ACTIVE_TIME
868cm3d928xg7	DELETE FROM AP_INVOICES_ALL WHERE ROWID = :B1	e:SQLAP:frm:APXINWKB	SQLAP/PAYABLES_MANAGER	1/29/2014 2:36:39 PM

```
--get Bind info to be sure that this SQL was the right invoice_id
SELECT
   b.name,
   b.position,  -- important if more than one
   b.datatype_string,
   b.value_string,
   b.child_number, -- important if more than one
   a.sql_text
FROM
  v$sql_bind_capture b,
  v$sqlarea          a
WHERE   b.sql_id = '868cm3d928xg7'  -- paste value here from prev query
AND   b.sql_id = a.sql_id
ORDER BY child_number, name
```

N...	POSITION	DATATYPE_STRING	VALUE_STRING	CHILD_NUMBER	SQL_TEXT
:B1	1	NUMBER	4852072	0	DELETE FROM AP_INVOICE_LINES_ALL WHERE INVOICE_ID = :B1

Bind info continued

Example of 3 bind variables with different child_number's so this SQL_ID was executed multiple times with different variables.

NAME	POSITION	DATATYPE_STRING	VALUE_STRING	CHILD_NUMBER	SQL_TEXT
:1	1	VARCHAR2(32)	5997645	0	SELECT xah.doc
:2	2	VARCHAR2(32)	200	0	SELECT xah.doc
:3	3	VARCHAR2(32)	4802112	0	SELECT xah.doc
:1	1	VARCHAR2(32)	9069939	1	SELECT xah.doc
:2	2	VARCHAR2(32)	222	1	SELECT xah.doc
:3	3	VARCHAR2(32)	4069499	1	SELECT xah.doc
:1	1	VARCHAR2(32)	5983795	2	SELECT xah.doc
:2	2	VARCHAR2(32)	200	2	SELECT xah.doc
:3	3	VARCHAR2(32)	5906667	2	SELECT xah.doc

Sleuthing – combined query

```
-- Find SQL statements with BindVariable
Select v.sql_id, sql_text, module, action, last_active_time, b.value_string
from v$sqlarea                  v,
        v$sql_bind_capture   b
where upper(sql_text) like '%DELETE%FROM%AP_INVOICE_LINES%'
and command_type = '7' -- For deleted txns
and v.sql_id = b.sql_id
and b.value_string = '4852072'
```

SQL_ID	SQL_TEXT	MODULE	ACTION	LAST_ACTIVE_TIME	VALUE_STRING
7bf9a5rqyw9pb	DELETE FROM AP_INVOICE_LINES_ALL WHERE INVOICE_ID = :B1	e:SQLAP:frm:APXINWKB	SQLAP/PAYABLES_MANAGER	1/29/2014 2:36:39 PM	4852072

Another example

```
select u.last_updated_by, last_logon_date, password_date,
  versions_operation,
  versions_starttime,
  versions_endtime
from  fnd_user versions between TIMESTAMP to_Date('23-OCT-2013 10:20:05', 'DD-MON-YYYY HH24:MI:SS') and sysdate u
where user_name = '24589'
order by 5,6
```

LAST_UPDATED_BY	LAST_LOGON_DATE	PASSWORD_DATE	VERSIONS_OPERATION	VERSIONS_STARTTIME	VERSIONS_ENDTIME
-1		9/1/2003	U	10/23/2013 3:06:26. PM	10/23/2013 3:06:41. PM
-1		9/1/2003	U	10/23/2013 3:06:41. PM	10/23/2013 3:07:27. PM
-1		9/1/2003	U	10/23/2013 3:07:27. PM	10/23/2013 3:07:41. PM
-1		9/1/2003	U	10/23/2013 3:07:41. PM	10/23/2013 3:12:15. PM
-1		9/1/2003	U	10/23/2013 3:12:15. PM	10/23/2013 3:13:54. PM
2283		9/1/2003	U	10/23/2013 3:13:54. PM	10/23/2013 3:14:27. PM
2283		9/1/2003	U	10/23/2013 3:14:27. PM	10/23/2013 3:14:57. PM
2283		9/1/2003	U	10/23/2013 3:14:57. PM	10/23/2013 3:16:08. PM
2283			U	10/23/2013 3:16:08. PM	10/23/2013 3:18:00. PM
2283	10/23/2013 3:18:03 PM		U	10/23/2013 3:18:00. PM	10/23/2013 3:18:21. PM
6	10/23/2013 3:18:03 PM	10/23/2013 3:18:24 PM	U	10/23/2013 3:18:21. PM	10/23/2013 3:18:21. PM
3405	10/23/2013 3:18:03 PM	10/23/2013 3:18:24 PM	U	10/23/2013 3:18:21. PM	10/23/2013 3:18:24. PM
3405	10/23/2013 3:18:24 PM	10/23/2013 3:18:24 PM	U	10/23/2013 3:18:24. PM	
-1		9/1/2003			10/23/2013 3:06:26. PM

Another example

Needed to find a query from the forms that I only had a bind variable and it returned the query.

```
-- Find SQL statements with BindVariable
Select v.sql_id, sql_text, module, action, last_active_time,
b.value_string, v.*
from v$sqlarea                  v,
        v$sql_bind_capture   b
where 1=1
and v.sql_id = b.sql_id
and b.value_string = '1902.64'
```

SQL_ID	SQL_TEXT
57q5jchyh16zn	SELECT /*+ push_pred(AP_INVOICES_V.pv) push_pred(AP_INVOICES_V.pvs) */ CUST_REGISTRATION_NUMBER,INVOICE_TYP

Other References

- http://nyoug.org/Presentations/2010/March/Johal_Flashback.pdf - For DBA's