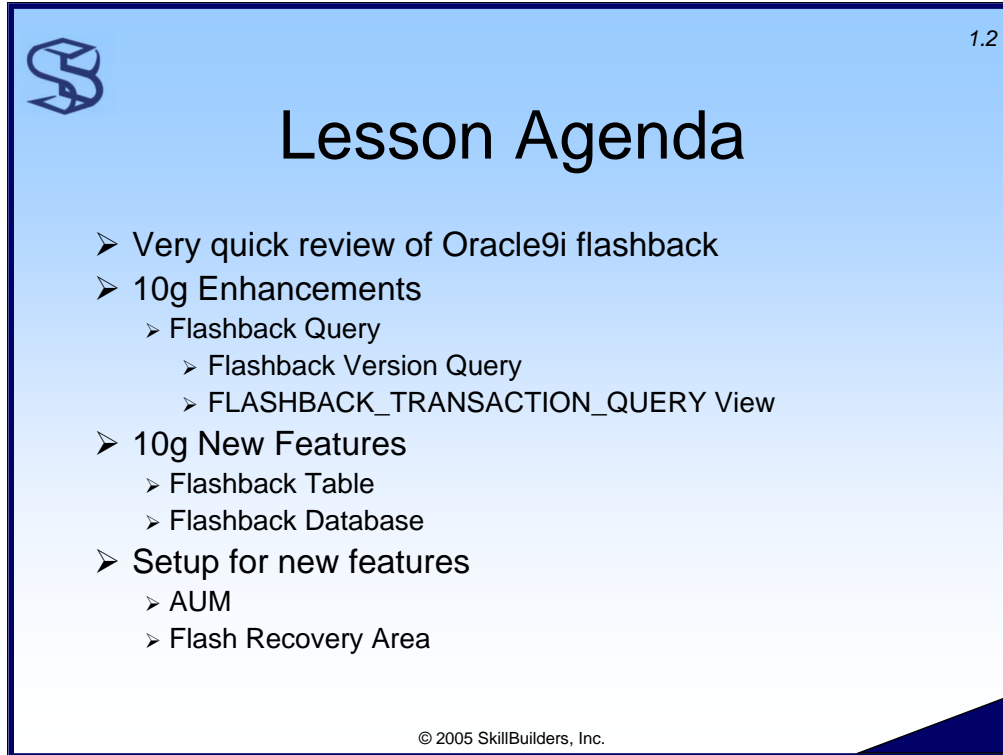


Author: Dave Anderson

Release Date: October 2004

Last Revision Date: January 2005

A blue slide with a dark blue border. In the top left corner is a stylized 'S' logo. In the top right corner is the number '1.2'. The title 'Lesson Agenda' is centered in a large, bold, black font. Below the title is a bulleted list of topics. At the bottom center, there is a small copyright notice: '© 2005 SkillBuilders, Inc.'

1.2

## Lesson Agenda

- Very quick review of Oracle9i flashback
- 10g Enhancements
  - Flashback Query
    - Flashback Version Query
    - FLASHBACK\_TRANSACTION\_QUERY View
- 10g New Features
  - Flashback Table
  - Flashback Database
- Setup for new features
  - AUM
  - Flash Recovery Area

© 2005 SkillBuilders, Inc.

In this lesson you will learn about the Oracle10g flashback-related features.

First, we will briefly review the Oracle9i flashback query feature (hopefully you have already had a chance to learn about or use 9i flashback query). We will also briefly discuss Automatic Undo Management (AUM, introduced with Oracle9i) because 9i and many 10g flashback-features require it. Then you will learn about the 10g enhancements, including flashback version query and flashback transaction query.

Then I will present new flashback-related features called “flashback table” and “flashback database”. Finally, because the flashback database feature requires it, I introduce something called the “Flash Recovery Area” (new with Oracle10g).

*Author's Note:* Technically, I consider flashback version query and flashback transaction query enhancements to Oracle9i's flashback query feature. I consider Oracle10g's flashback table and flashback database new flashback-related features.

Flashback Evolution

- 9i provided
  - Session-level flashback
    - DBMS\_FLASHBACK
  - Statement (and sub-statement) flashback
    - SQL "AS OF" clause

9i R1

9i R2

Can compare table to a previous version of *itself*

```
SQL> select a.lastname, a.total_purchase, b.total_purchase
2  from sales a , sales AS OF timestamp(sysdate - 1) b
3  where a.cust_no = b.cust_no
4  and a.total_purchase != nvl(b.total_purchase, 0);
```

LASTNAME	TOTAL_PURCHASE	TOTAL_PURCHASE
ANDERSON	Amount today	55000
DASWANT	Amount yesterday	55000

© 2005 SkillBuilders, Inc.

Oracle9i Release 1 introduced the concept of "flashback". With 9i Release 1, we have session-level flashback query ability – via the DBMS\_FLASHBACK PL/SQL package. For example, after enabling flashback at the session level, the subsequent queries show the data as it existed 1 day ago:

```
SQL> exec dbms_flashback.enable_at_time(systimestamp-1)
```

PL/SQL procedure successfully completed.

```
SQL> select count(*) from customer;
```

```
  COUNT(*)
-----
       14
```

```
SQL> select count(*) from employee;
```

```
  COUNT(*)
-----
       10
```

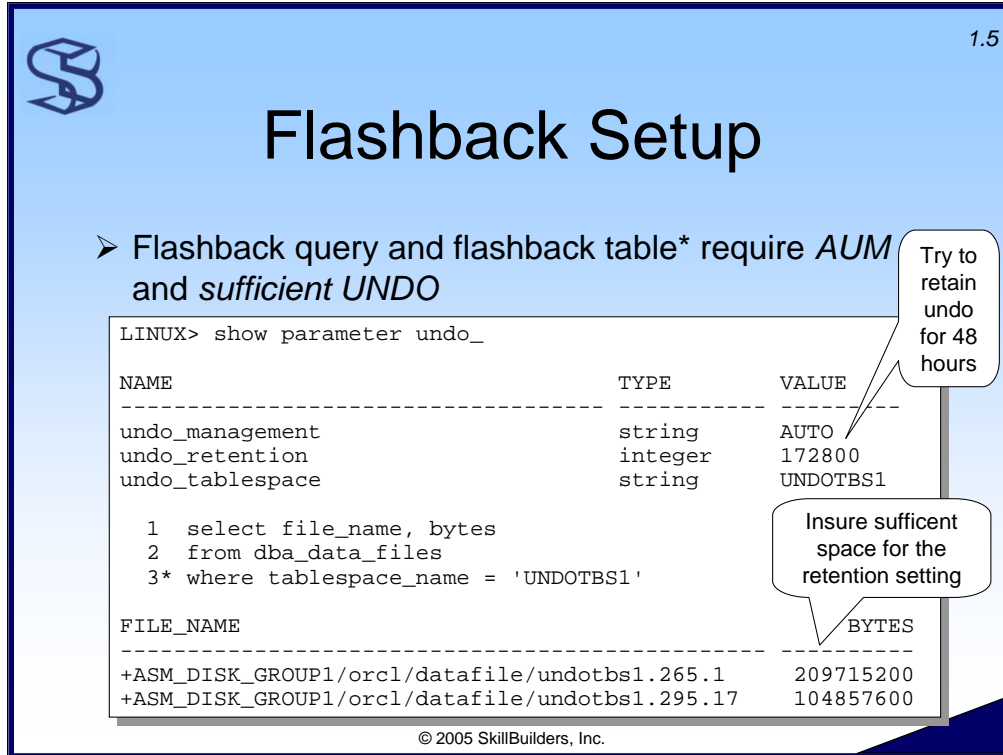
```
SQL> exec dbms_flashback.disable
```

PL/SQL procedure successfully completed.

*Notes for this slide continue on the next page...*

*Notes continued from the previous page...*

Oracle9i Release 2 went a big step further with the “FROM table AS OF” clause. This provides the ability to flashback at the statement or even sub-statement level. As the example above shows, one table in a join can be flashed back, while the other can be current; thus, we can see that sales yesterday to customers Anderson and Daswani were null, and today they are 55,000.



1.5

## Flashback Setup

➤ Flashback query and flashback table\* require *AUM* and *sufficient UNDO*

```
LINUX> show parameter undo_
```

NAME	TYPE	VALUE
undo_management	string	AUTO
undo_retention	integer	172800
undo_tablespace	string	UNDOTBS1

```

1 select file_name, bytes
2 from dba_data_files
3* where tablespace_name = 'UNDOTBS1'

```

FILE_NAME	BYTES
+ASM_DISK_GROUP1/orcl/datafile/undotbs1.265.1	209715200
+ASM_DISK_GROUP1/orcl/datafile/undotbs1.295.17	104857600

Try to retain undo for 48 hours

Insure sufficient space for the retention setting

© 2005 SkillBuilders, Inc.

With Oracle9i and Oracle10g, administrators need to implement Automatic Undo Management (AUM) to use flashback query. AUM is the Oracle9i feature that replaces manual management or rollback segments. (Actually, I have tested flashback query with *manual* undo and it works. However, you do not have explicit control over the retention time. Therefore, for that reason and several other reasons outside the scope of this lesson, you should migrate to AUM.)

\* The Oracle10g flashback table feature requires AUM too – unless you are recovering a dropped table – then it uses the “recyclebin”. You will learn more about the flashback table feature later in this lesson.

AUM is implemented with initialization parameters shown above. Set `UNDO_MANAGEMENT=AUTO` (versus `MANUAL`) and set the `UNDO_RETENTION` parameter to the desired amount of time you want to retain undo. Undo retention affects flashback query (i.e. how far back do you want to support flashback?) and long running queries that require a consistent image of data (retaining undo for the duration of the long running query can reduce or eliminate ORA-01555 “Snapshot too old” errors). In the example shown above, the database will attempt to keep undo records for a minimum of 48 hours.

*Notes for this slide continue on the next page...*

However, if undo space is required for the undo of ongoing transactions, and no free space is available in the undo tablespace, Oracle will prematurely expire undo records, i.e. overwrite records younger than the UNDO\_RETENTION value. This can cause flashback queries to fail or ORA-01555 “snapshot too old” errors.

The second query shown above reveals the name and size of the datafiles associated with my undo tablespace. (You will notice that my UNDO datafile names start with “+ASM\_DISK\_GROUP1”. This is a sure sign I am using ASM to control my datafiles. Refer to the lesson on Automatic Storage Management for more information on ASM.)

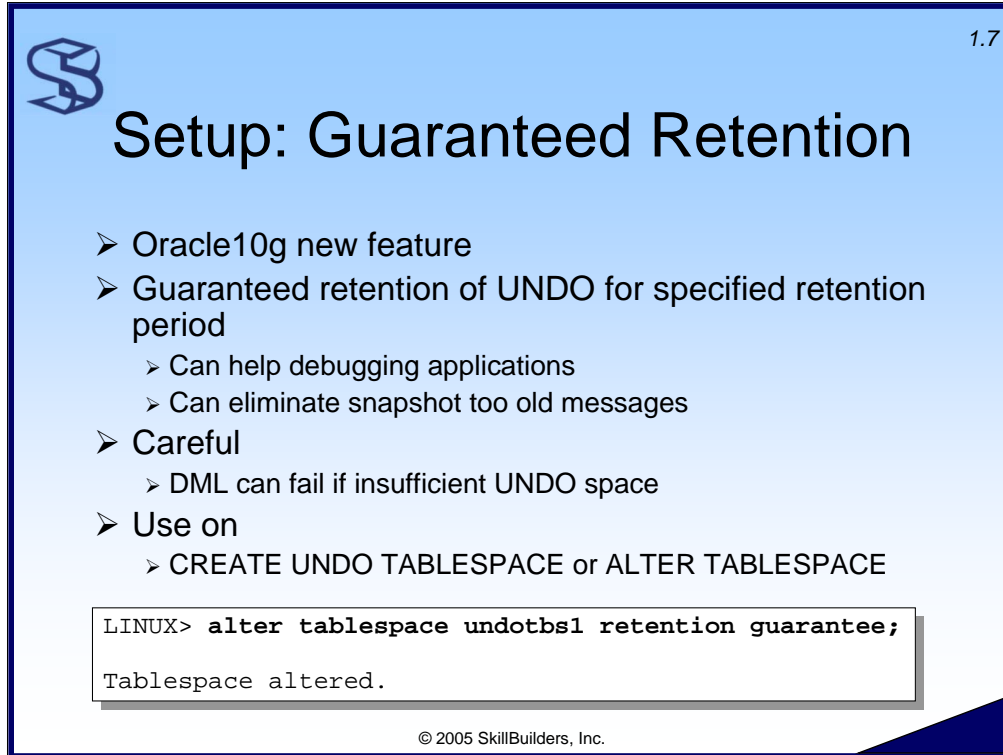
### Supplemental Notes

Insure that the undo tablespace has sufficient space to accommodate the undo generated during the specified retention time. For assistance sizing the undo tablespace, refer to the supplied script UNDO\_FILE\_SIZE.SQL, which contains a Metalink query that uses the historical undo usage on your server to estimate the required tablespace size. Oracle Enterprise Manager can also estimate the Undo tablespace size requirement.

If you are not familiar with dynamically adjusting initialization parameters, here’s an example. To setup for 48 hours of undo

```
LINUX> alter system set undo_retention=172800
  2  comment='48 hours of undo for flashback support'
  3  scope=both;
```

System altered.



**Setup: Guaranteed Retention**

- Oracle10g new feature
- Guaranteed retention of UNDO for specified retention period
  - Can help debugging applications
  - Can eliminate snapshot too old messages
- Careful
  - DML can fail if insufficient UNDO space
- Use on
  - CREATE UNDO TABLESPACE or ALTER TABLESPACE

```
LINUX> alter tablespace undotbs1 retention guarantee;  
Tablespace altered.
```

© 2005 SkillBuilders, Inc.

If you want to guarantee that undo records will be kept for your specified retention time, consider using the Oracle10g “Guaranteed Retention” feature. Enabling this feature will cause Oracle to unequivocally honor the UNDO\_RETENTION parameter – *at the expense of failing ongoing database operations if sufficient undo space is not available.*

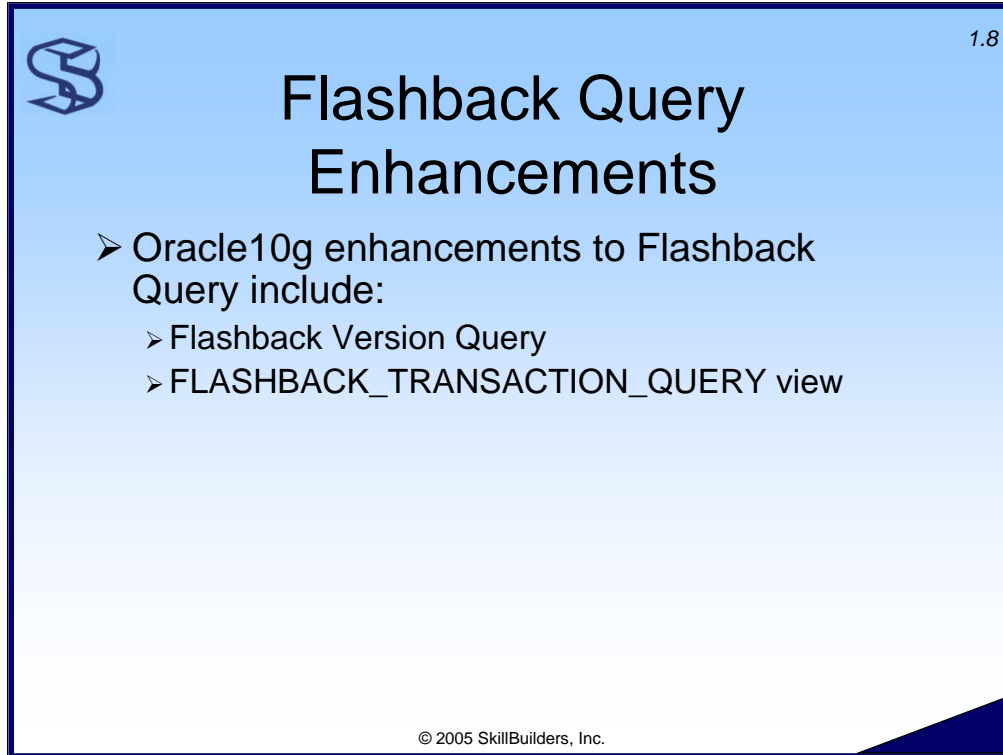
By setting the UNDO\_RETENTION parameter value equal to the longest running query (elapsed time) and enabling guaranteed retention, you can reduce or possibly eliminate “snapshot too old” errors.

In addition to helping to eliminate “snapshot too old” errors, it can be useful for debugging. For example, assume your application consumes ten hours of elapsed time and sometime during that period the application introduces bad data. By guaranteeing retention for the duration of debugging process, you’ll be able to flashback query for the entire run of the application and determine when the anomaly was introduced.

This feature is implemented via the RETENTION GUARANTEE parameter on CREATE UNDO TABLESPACE and ALTER TABLESPACE statements.

Turn off retention guarantee with the following statement:

```
LINUX> alter tablespace undotbs1 retention noguarantee;
```



1.8

## Flashback Query Enhancements

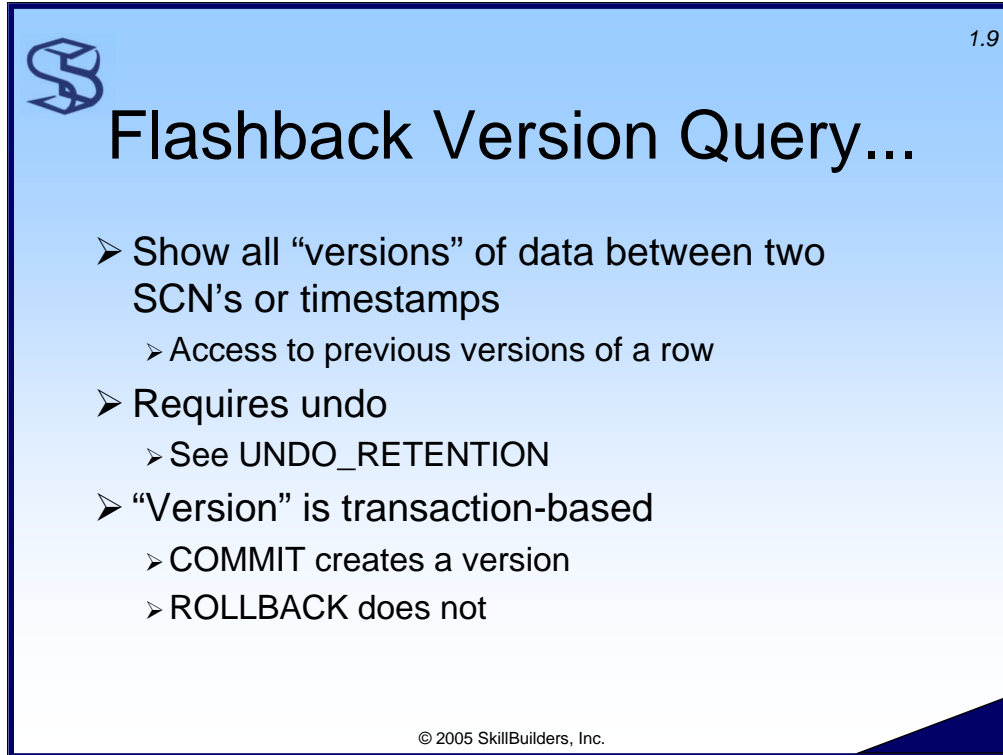
- Oracle10g enhancements to Flashback Query include:
  - Flashback Version Query
  - FLASHBACK\_TRANSACTION\_QUERY view

© 2005 SkillBuilders, Inc.

Release 1 of Oracle10g provides two enhancements to flashback query that we will discuss next:

- Flashback Version Query – Access to old versions of table data.
- The FLASHBACK\_TRANSACTION\_QUERY view – Access to undo SQL statements, user who made the change and other details about changes made to table data.



A slide with a light blue background and a dark blue border. In the top left corner is a stylized 'S' logo. In the top right corner is the number '1.9'. The main title is 'Flashback Version Query...'. Below the title is a bulleted list of features. At the bottom center is the copyright notice '© 2005 SkillBuilders, Inc.'

1.9

## Flashback Version Query...

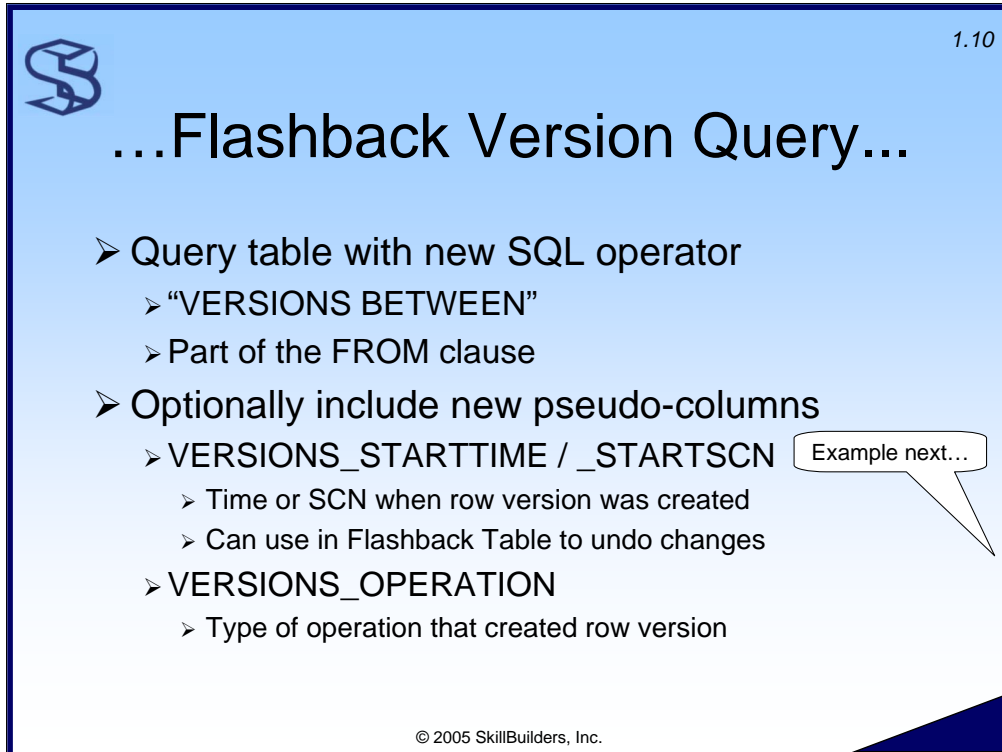
- Show all “versions” of data between two SCN’s or timestamps
  - Access to previous versions of a row
- Requires undo
  - See UNDO\_RETENTION
- “Version” is transaction-based
  - COMMIT creates a version
  - ROLLBACK does not

© 2005 SkillBuilders, Inc.

The Flashback Version Query feature let’s us see the values of table data as it changed over time. We request the database show us all versions between two points in time (in the form of timestamps) or between two system change numbers (SCN’s). Another way of describing this feature is to say it provides access to *old or previous* versions of data.

This feature requires that the undo records for the transactions that changed the table data are available. Remember that the UNDO\_RETENTION parameter is used to help preserve undo records for the specified period of time.

The concept of a “version” of a row is transaction-based: a version is created when a transaction involving a row commits. So, if there have been 10 transactions that have changed a row – and 8 were committed – between yesterday and today, and you request all versions of that row in that time period, you will see 8 rows of output – one for each version. Changes that were rolled back do not generate a version.



1.10

## ...Flashback Version Query...

- Query table with new SQL operator
  - “VERSIONS BETWEEN”
  - Part of the FROM clause
- Optionally include new pseudo-columns
  - VERSIONS\_STARTTIME / \_STARTSCN Example next...
    - Time or SCN when row version was created
    - Can use in Flashback Table to undo changes
  - VERSIONS\_OPERATION
    - Type of operation that created row version

© 2005 SkillBuilders, Inc.

Flashback Version Query is implemented via extensions to SQL. This includes:

- A FROM clause sub-clause called VERSIONS BETWEEN. In addition to the example shown on the next page, you can find more information on VERSIONS BETWEEN in Chapter 19 of the **Oracle10g SQL Reference**.
- New pseudo-columns including VERSIONS\_STARTTIME (contains the timestamp when a version was created) and VERSIONS\_OPERATION (i.e. how did the value in this version get this way? I for INSERT, U for UPDATE, etc.). Refer to Chapter 3 of the **SQL Reference** and Chapter 15 of the **Oracle10g Application Developers Guide – Fundamentals** for complete details on the VERSIONS pseudo-columns.

1.11

## ...Flashback Version Query...

```

DAVE@LINUX> select c2, versions_starttime, versions_endtime,
2             versions_startscn , versions_endscn,
3             versions_operation, versions_xid
4 from system.test
5     versions between timestamp
6             to_timestamp('11-SEP-04 12.02.00.000000000 PM',
7             'dd-mon-yy hh.mi.ss.ff PM')
8             and systimestamp
9 where c1 = '1'
10 order by versions_startscn nulls first;

```

C2	VERSIONS_STARTTIME	VERSIONS_ENDTIME	VERSIONS_STARTSCN
a		11-SEP-04 12.01.55 PM	
b	11-SEP-04 12.01.55 PM	11-SEP-04 12.01.55 PM	988714
c	11-SEP-04 12.01.55 PM	11-SEP-04 12.02.04 PM	988716
d	11-SEP-04 12.02.04 PM	11-SEP-04 12.02.17 PM	988719
x	11-SEP-04 12.02.17 PM		

© 2005 SkillBuilders, Inc.

Use the **VERSIONS BETWEEN** clause to code a “flashback versions query”. In this example we see that between the times specified there were 5 versions of one row in existence (why one row? see that on line 9 the query includes a filter on primary key).

The **VERSIONS\_STARTTIME / STARTSCN** value is the time or SCN when the row version was created. It can be used in a flashback table operation to undo application changes (flashback table is presented later in this lesson).

Columns not shown in slide due to space constraints:

```

VERSIONS_ENDSCN V VERSIONS_XID
-----
988714
988716 U 0500050044030000
988719 U 040006000B040000
988730 U 03000E00270C0000
          U 0900050015030000

```

The transaction ID can be used to locate the REDO and UNDO SQL. Refer to the section on **FLASHBACK\_TRANSACTION\_QUERY** later in this lesson.

In this example, all of the row versions were created by UPDATE statements (see the “U” value). The **VERSIONS\_XID** column contains the transaction id and can be used to find the username redo SQL, undo SQL, among other things.

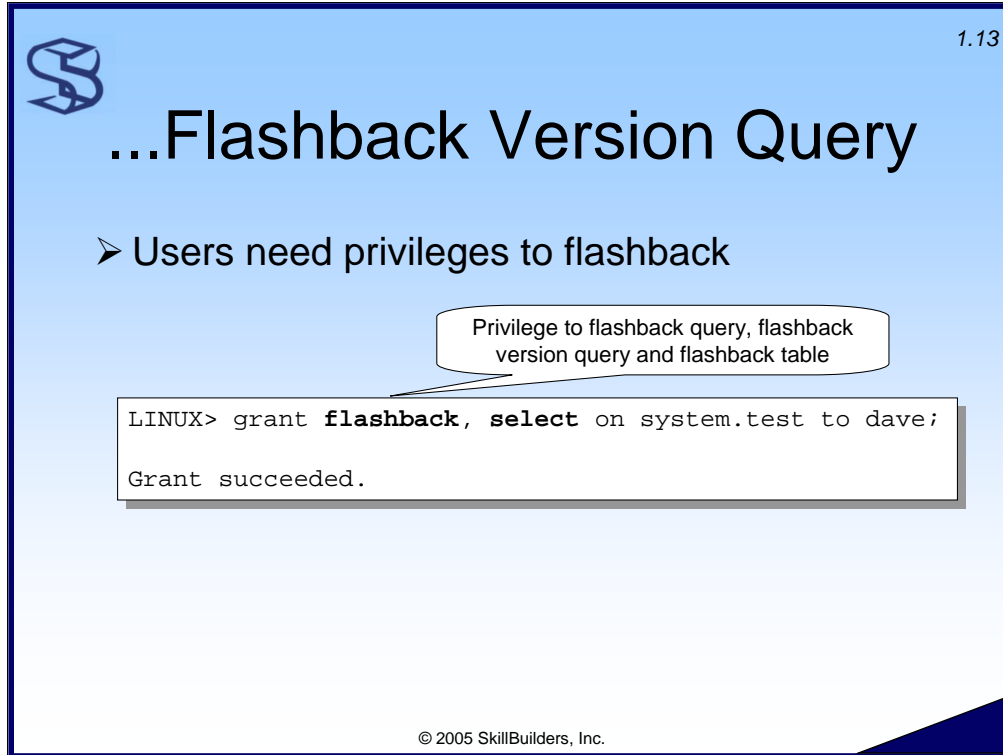
*Notes for this slide continue on the next page...*

See supplied script flashback\_version\_query.sql for a working copy of the example shown in this slide.

### Supplemental Notes

Oracle10g supports MINVALUE and MAXVALUE in the VERSIONS BETWEEN clause. These expressions give access to the “oldest and most recent data available.” (Source: **Oracle10g SQL Reference**.) For example:

```
select * from t2 versions between scn  
minvalue and maxvalue
```



1.13

## ...Flashback Version Query

- Users need privileges to flashback

Privilege to flashback query, flashback version query and flashback table

```
LINUX> grant flashback, select on system.test to dave;  
Grant succeeded.
```


© 2005 SkillBuilders, Inc.

DBAs will need to give flashback privileges to desired users. The FLASHBACK object privilege, introduced with Oracle9i, provides the ability to:

- use the AS OF clause to flashback a table, view or materialized view
- the VERSIONS BETWEEN clause to extract available versions of data
- use the FLASHBACK TABLE statement (discussed later in this lesson).

In addition to the examples above, you can also grant:

- Execute on DBMS\_FLASHBACK for session-level flashback query
- The system privilege 'flashback any table' to use flashback query / flashback version query or flashback table on any table the user has select privilege on.



## FLASHBACK\_TRANSACTION\_

## QUERY View

1.14

- Mine (audit) undo records for details on changes
  - Easier than log miner utility
- Get XID from flashback version query

```

LINUX> exec print_table('select logon_user, undo_sql -
>                        from flashback_transaction_query -
>                        where xid = ''0700010059020000'' ')
LOGON_USER   : SYSTEM
UNDO_SQL     : delete from "SYSTEM"."TEST"
              where ROWID = 'AAALquAABAAAaIiAAA'
```

```

LINUX> grant select any transaction to dave;

Grant succeeded.
```

Get XID from flashback version query

User needs this privilege

© 2005 SkillBuilders, Inc.

It is easy to see details about changes made to tables with the FLASHBACK\_TRANSACTION\_QUERY view. This can be helpful for auditing (i.e. who changed my table?), debugging and even performance analysis (how often is a table being changed?).

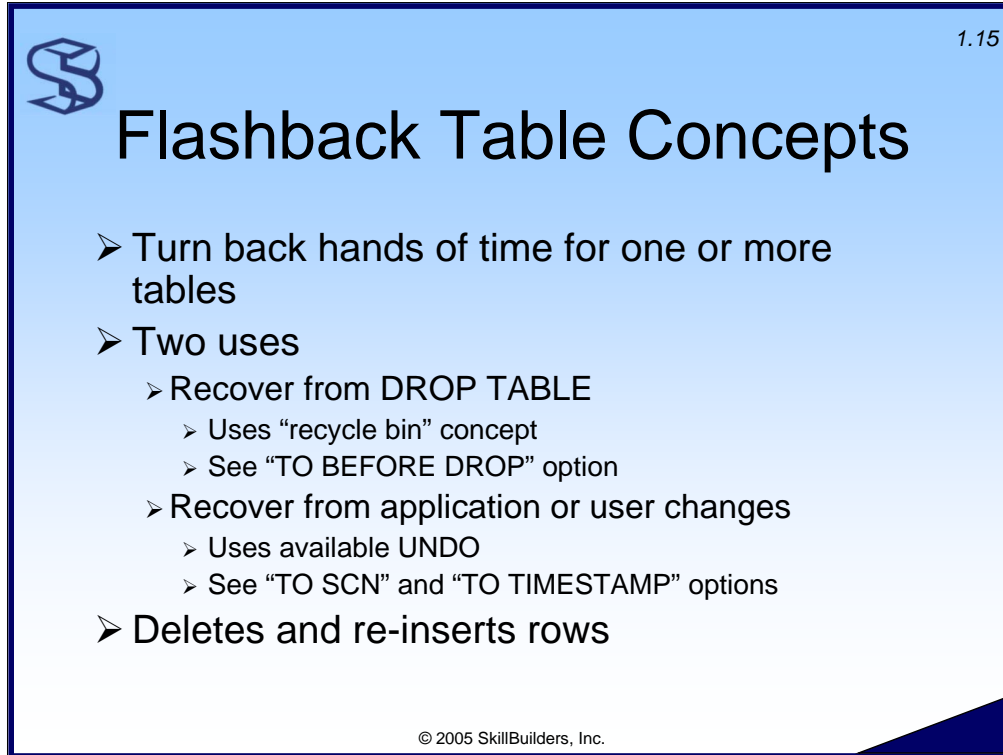
The SELECT ANY TRANSACTION privilege allows a user to see details about past transactions by querying the new (10g) FLASHBACK\_TRANSACTION\_QUERY view.

Note that – in my tests – the performance of querying the FLASHBACK\_TRANSACTION\_QUERY view is very poor. I have opened an iTAR on Metalink and will post any findings or solutions when they are available (see [www.skillbuilders.com](http://www.skillbuilders.com)).

See supplied script flashback\_transaction\_query.sql for a working copy of this code.

### Supplemental Notes

The SELECT ANY TRANSACTION privilege also allows a user to see details about transactions by using the log miner view V\$LOGMNR\_CONTENTS (9i).

A slide titled "Flashback Table Concepts" with a blue background and a dark blue border. In the top left corner is a stylized 'S' logo. In the top right corner is the number "1.15". The slide contains a bulleted list of concepts. At the bottom center, there is a copyright notice: "© 2005 SkillBuilders, Inc."

1.15

## Flashback Table Concepts


- Turn back hands of time for one or more tables
- Two uses
  - Recover from DROP TABLE
    - Uses “recycle bin” concept
    - See “TO BEFORE DROP” option
  - Recover from application or user changes
    - Uses available UNDO
    - See “TO SCN” and “TO TIMESTAMP” options
- Deletes and re-inserts rows

© 2005 SkillBuilders, Inc.

Use the Flashback Table feature to “turn back the hands of time” for a table. (Note that I first heard this feature described this way by Tom Kyte.)

It can be used to:

- Recover a dropped table. This type of recovery uses the new recycle bin concept which we explore in this section of the lesson.
- Recover from application changes. This type of recover requires UNDO.
- This feature is implemented with the FLASHBACK TABLE statement, which we will see examples of in this section of the lesson.

1.16

## Flashback Table Prep

- Enable row movement on table
- Grant privileges to desired user(s)

```
SYSTEM@LINUX> alter table test enable row movement;
Table altered.
SYSTEM@LINUX> grant flashback on test to dave;
Grant succeeded.
SYSTEM@LINUX> grant alter, select, update, delete,
2          insert on system.test to dave;
Grant succeeded.
```

© 2005 SkillBuilders, Inc.

There are two things you need to do before using the flashback table feature:

- Enable row movement. This is because – if you are recovering from (i.e. undoing) application changes – rows will be inserted, updated and deleted as part of the flashback table process. Row movement allows the ROWID to change. Note that this is required even if the table is not partitioned.
- Get necessary privileges. This includes the flashback object privilege (or the system privilege FLASHBACK ANY TABLE) and all the DML privileges listed above.



1.17

## Flashback Table: Undo Application Changes

```
DAVE@LINUX> select dbms_flashback.get_system_change_number
from dual;

----- X
1057181
```

Undo the last 30 minutes of changes

Optional, but can undo flashback if you know SCN prior to flashback

```
DAVE@LINUX> flashback table system.test to timestamp
2 systimestamp - interval '30' minute;
```

```
DAVE@linux3> exec dbms_stats.gather_table_stats
(ownname=>'system', tabname=>'test')
```

PL/SQL procedure successfully completed.

Update statistics after flashback table

© 2005 SkillBuilders, Inc.

Here we see the Flashback Table feature in action. In this example, my goal is to undo the last 30 minutes of changes. You can also flashback to a system change number, which you can get from a flashback version query (see the `VERSIONS_STARTSCN` pseudocolumn shown earlier in this lesson).

Note that the `FLASHBACK TABLE` statement is a DDL statement and is not rollback-able. However, if you capture the system change number prior to executing the statement, you can undo the first `FLASHBACK TABLE` statement by executing a second `FLASHBACK TABLE` statement, flashing back to the system change number in affect just prior to the first flashback. For example:

```
DAVE@LINUX> flashback table system.test to scn 1057181;
```

```
Flashback complete.
```

Collect statistics after executing `FLASHBACK TABLE`; statistics are not updated during the operation and will be stale.

See the supplied script `flashback_table.sql` for a demonstration of flashback table.

*Notes for this slide continue on the next page...*

## Supplemental Notes

There are several additional things about flashback table that I discovered during my testing that might help you:

- Flashback table honors *referential integrity constraints*. For example, if you attempt to flashback a parent table, and the result of the operation would leave constraint violations, the flashback will fail. However, you can flashback a referential group of tables. For example:

```
flashback table t1, t2 to scn 7930536;
```

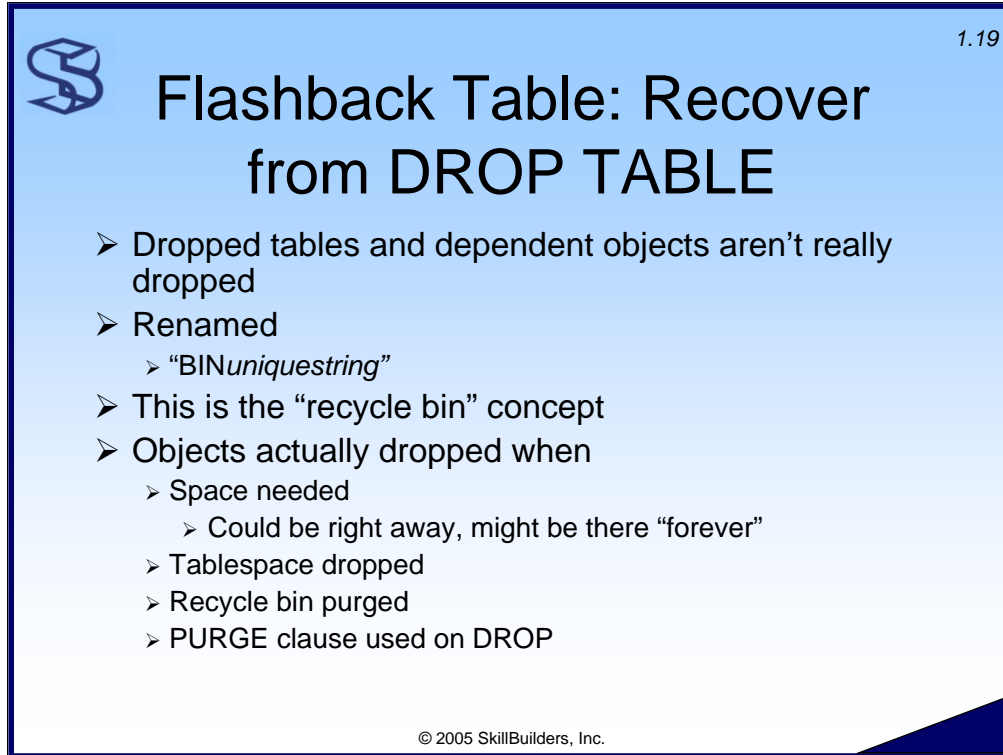
- You cannot flashback a table involved in an *active transaction* (all tables are locked during the flashback operation).
- Flashback table under Release 10.1.0.2 leaves a temporary table called `SYS_TEMP_FBT`. You can drop this table to cleanup the FLASHBACK TABLE process. (I think this is known bug #3076151, but lately I've been unable to find this bug documented on Metalink.)

There are several things I learned from Oracle docs:

- *Triggers* are disabled during the flashback operation, then enabled when the operation is complete. You can override this with the ENABLE TRIGGERS clause on the FLASHBACK TABLE statement. However, I'm not sure why you would want to take the performance hit of reexecuting the triggers on data already in the table. (Tom Kyte suggests that perhaps it might be useful to enable auditing-type triggers.)
- Indexes are kept in sync by the flashback operation. However, indexes are never created nor dropped by a flashback operation.

Flashback Table *cannot* be used with the following types of objects:

- Clusters
- Materialized Views
- Advanced Queuing tables
- Dictionary tables
- Remote Tables
- Object Types
- Nested Tables
- Partitions/Sub-partitions



The slide features a blue background with a dark blue border. In the top left corner, there is a stylized 'S' logo. The title 'Flashback Table: Recover from DROP TABLE' is centered at the top. A list of bullet points is on the left side. The slide number '1.19' is in the top right corner. A copyright notice is at the bottom center.

1.19

## Flashback Table: Recover from DROP TABLE

- Dropped tables and dependent objects aren't really dropped
- Renamed
  - "BINuniquestring"
- This is the "recycle bin" concept
- Objects actually dropped when
  - Space needed
    - Could be right away, might be there "forever"
  - Tablespace dropped
  - Recycle bin purged
  - PURGE clause used on DROP

© 2005 SkillBuilders, Inc.

In Oracle10g, the DROP TABLE statement, by default, actually renames a table – it does not drop it. The renamed table becomes part of the "recycle bin". The recycle bin is really nothing more than a bunch of dropped (renamed) tables and dependent indexes, sitting in their original location, albeit with strange system-defined names.

The table will reside in the recycle bin until tablespace storage becomes constrained, the tablespace is dropped or the recycle bin is purged.

You can specify the new PURGE clause on the DROP TABLE, which causes the table to actually be dropped immediately. (See the example later in this lesson.)

In addition to the examples in this lesson, refer to Chapter 14 of the **Oracle10g Administrator's Guide** and Metalink Note 266413 for more information on "Using Flashback Drop and Managing the Recycle Bin".

*Notes for this slide continue on the next page...*

**Supplemental Notes**

Objects in the SYSTEM tablespace are not put into the recycle bin. Only objects in non-system locally managed tablespaces are put in the recycle bin when dropped.

You can disable the recycle bin for the database (i.e. revert to Oracle9i treatment of DROP TABLE):

```
DAVE@linux3> alter system set "_recyclebin"=false;
```

```
System altered.
```

**Limitations**

External tables, materialized views and bitmap join indexes do not appear in the recycle bin and therefore cannot be recovered with the Flashback Table feature.

1.21

## Flashback Table: Recover from DROP

➤ Recover from DROP TABLE or unwanted application changes

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

SQL> **show recyclebin**

ORIGINAL NAME	RECYCLEBIN NAME
BIG	BIN\$Kq3scQsDQP+jkI0atdsqqQ== \$0

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

See also the optional "RENAME TO" clause

Table segment remains unless PURGE used

Table is recorded in "recycle bin"

Easy and fast to recover table

© 2005 SkillBuilders, Inc.

This example demonstrates the recycle bin and recovering a dropped table. Note that after table BIG is dropped, it appears in the recycle bin. Note the use of the new 10g SQL\*Plus command SHOW RECYCLEBIN.

The FLASHBACK TABLE TO BEFORE DROP statement recovers the dropped table.


The FLASHBACK TABLE statement supports an optional RENAME TO clause, which will rename the table as it recovers it from the recycle bin.

Refer to the supplied script flashback\_dropped\_table.sql for a working example of the FLASHBACK TABLE . . . TO BEFORE DROP statement.

### Supplemental Notes

Note also that dropped external tables do not appear in the recycle bin and therefore cannot be recovered with the Flashback Table feature.

Even if you recover a parent table (in a referential set), the dependent foreign keys are not recovered. You will have to rebuild them manually.



1.22

## Repeat Un-Drop

```

SQL> show recyclebin
ORIGINAL NAME      RECYCLEBIN NAME      OBJECT TYPE  DROP TIME
-----
T                  BIN$KbsjpLROts6+eGOGqifipQ==$0  TABLE      2004-09-19:14:59:30
T                  BIN$y0AR2dYXSnC0bRn+vJTF0Q==$0  TABLE      2004-08-19:16:52:54

SQL> flashback table t to before drop;

Flashback complete.

SQL> show recyclebin
ORIGINAL NAME      RECYCLEBIN NAME      OBJECT TYPE  DROP TIME
-----
T                  BIN$y0AR2dYXSnC0bRn+vJTF0Q==$0  TABLE      2004-08-19:16:52:54
SQL> rename t to t_old;

Table renamed.

SQL> flashback table t to before drop;

Flashback complete.

SQL> show recyclebin
SQL>

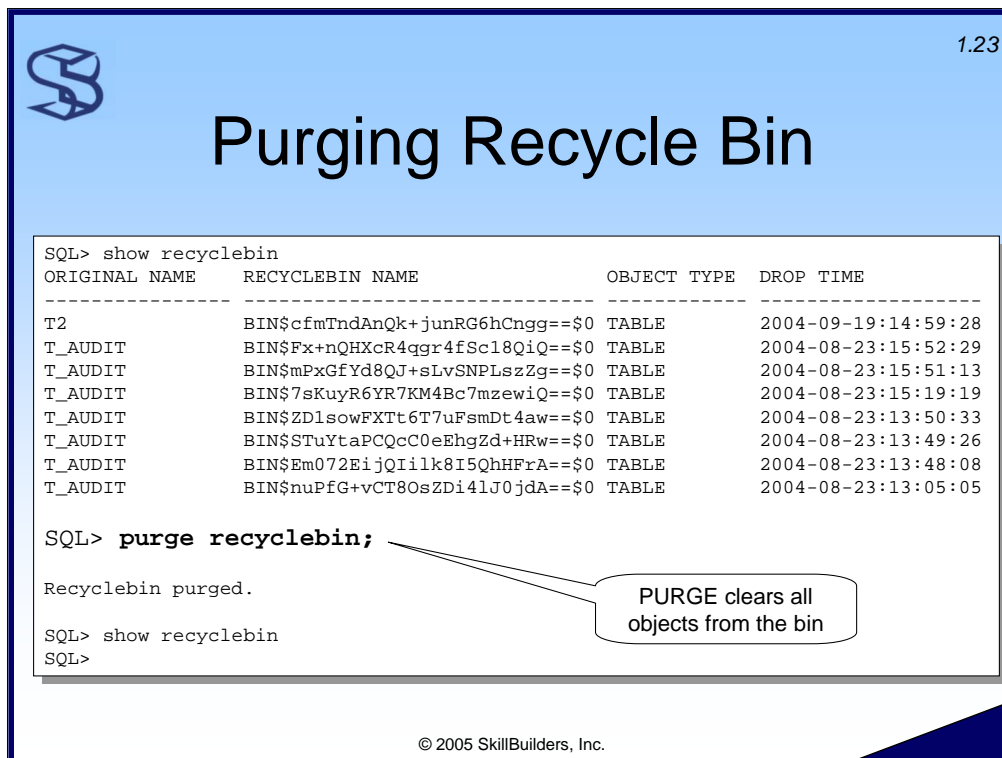
```

© 2005 SkillBuilders, Inc.

If a table was dropped, a new version created, dropped again, you will have multiple occurrences in the recycle bin. Multiple FLASHBACK TABLE statements, with a RENAME table in-between (or use the RENAME TABLE option on the FLASHBACK TABLE statement) will get you back to the original version of the dropped table.

### Supplemental Notes

The SQL\*Plus command SHOW RECYCLEBIN only shows dropped tables (not their dependents). Use SELECT \* FROM USER\_RECYCLEBIN to see the entire contents of your recycle bin. Query DBA\_RECYCLEBIN to see all recycle bin objects – from all schemas.



1.23

## Purging Recycle Bin

```

SQL> show recyclebin
ORIGINAL NAME      RECYCLEBIN NAME          OBJECT TYPE  DROP TIME
-----
T2                 BIN$cfmTndAnQk+junRG6hCngg==$0  TABLE      2004-09-19:14:59:28
T_AUDIT           BIN$FxnQHxcR4qgr4fSc18QiQ==$0  TABLE      2004-08-23:15:52:29
T_AUDIT           BIN$mPxGfYd8QJ+sLvSNPLszZg==$0  TABLE      2004-08-23:15:51:13
T_AUDIT           BIN$7sKuyR6YR7KM4Bc7mzewiQ==$0  TABLE      2004-08-23:15:19:19
T_AUDIT           BIN$ZD1sowFXTt6T7uFsmDt4aw==$0  TABLE      2004-08-23:13:50:33
T_AUDIT           BIN$STuYtaPCQcC0eEhgZd+HRw==$0  TABLE      2004-08-23:13:49:26
T_AUDIT           BIN$Em072EijQIilk8I5QhHFrA==$0  TABLE      2004-08-23:13:48:08
T_AUDIT           BIN$nnuPFG+vCT8OsZDi4lJ0jdA==$0  TABLE      2004-08-23:13:05:05

SQL> purge recyclebin;

Recyclebin purged.

SQL> show recyclebin
SQL>

```

PURGE clears all objects from the bin


© 2005 SkillBuilders, Inc.

The new SQL statement PURGE removes (drops) all objects in the recycle bin and frees the space used. PURGE RECYCLEBIN purges the current user's recycle bin.

There are many options for the PURGE statement including

- PURGE TABLE table-name
- PURGE INDEX index-name
- PURGE RECYCLEBIN
- PURGE DBA\_RECYCLEBIN (Purges the bin of every user. Requires SYSDBA privilege.)
- PURGE TABLESPACE tspace-name [USER username]

Note that the PURGE statement is not rollback-able. Refer to the **Oracle10g SQL Reference** for more details on the PURGE statement.



## Flash Recovery Area: Concepts

1.24

- Optional storage area for backup-related files
  - Online and archive logs
  - RMAN backups
    - Default location for RMAN backups if configured
  - Flashback logs
    - Required for FLASHBACK DATABASE
- “Automates management of backup-related files”
  - Convenient directory structure
  - Auto-delete obsolete files when space needed

Next topic is FB Database...

© 2005 SkillBuilders, Inc.


The Flash Recovery Area, introduced with Oracle10g, is a storage area for many types of recovery-related files. This includes flashback logs, required for the FLASHBACK DATABASE statement. You will learn more about flashback logs and the FLASHBACK DATABASE statement later in this lesson. It also optionally includes redo logs and RMAN backups.

The Oracle documentation says the Flash Recovery Area “Automates management of backup-related files.” I have discovered this to mean:

- Oracle creates a convenient and meaningful directory structure to the files store within it.
- Oracle automatically removes obsolete logs when space is required.
- RMAN provides a convenient technique for backing up the recovery area to tape. See the “BACKUP RECOVERY AREA” command.



1.25



## Flash Recovery Area: Configuration

- Set location
  - Separate from datafiles
- Set space limit
  - Maximum space dedicated to flashback area
- Set retention limit
  - How far back can we flashback database?

NAME	VALUE
db_recovery_file_dest	/mnt/mickeymantle
db_recovery_file_dest_size	10G
db_flashback_retention_target	1440
log_archive_dest_1	LOCATION=USE_DB_RECOVERY_FILE_DEST

Optionally write archive logs to flash recovery area

© 2005 SkillBuilders, Inc.

Set the location, space limit and retention limit for flashback logs when establishing the flash recovery area. Since this is a recovery-related area, keep this on separate disk devices from the database files. Optionally write the archive logs to this area as well.

Oracle will begin to “complain” if you start running out of space. By complain, I mean that it will write messages to the alert log. Refer to the next page for an example of an alert log entry.

### Supplemental Notes

1. A related step (for configuration of the flash recovery area) is to enable flashback for the database with the ALTER DATABASE FLASHBACK ON statement. An example of this statement is shown later in this lesson.
2. The Database Configuration Assistant supports the configuration of the Flash Recovery Area.
3. The LOG\_ARCHIVE\_START parameter is deprecated in Oracle10g.

This is the alert log entry if you run short of space in the flash recovery area:

```
Errors in file /u01/app/oracle/admin/orcl/udump/orcl_ora_2723.trc:
```

```
ORA-19815: WARNING: db_recovery_file_dest_size of 2147483648 bytes is  
92.12% used, and has 169270784 remaining bytes available.
```

```
*****
```

You have the following choices to free up space from  
flash recovery area:

1. Consider changing your RMAN retention policy.

If you are using dataguard, then consider changing your  
RMAN archivelog deletion policy.

2. Backup files to tertiary device such as tape using the  
RMAN command BACKUP RECOVERY AREA.

3. Add disk space and increase the db\_recovery\_file\_  
dest\_size  
parameter to reflect the new space.

4. Delete unnecessary files using the RMAN DELETE command.

If an OS command was used to delete files, then use  
RMAN CROSSCHECK and DELETE EXPIRED commands.

```
*****
```

The slide features a blue background with a white border. In the top left corner is the Oracle logo. The title 'Flash Recovery Area: Management' is centered at the top. Below the title is a sub-heading '➤ New V\$ view'. A SQL query is shown in a white box with a grey border, followed by its output. Two callout boxes with white backgrounds and black borders point to specific values in the output: one points to 'SPACE\_RECLAIMABLE' and the other points to 'NUMBER\_OF\_FILES'. The copyright notice '© 2005 SkillBuilders, Inc.' is at the bottom center.

1.27

## Flash Recovery Area: Management

➤ New V\$ view


```
SQL> exec print_table('select * from v$recovery_file_dest')
NAME                                : /mnt/mickeymantle/
SPACE_LIMIT                          : 10737418240
SPACE_USED                           : 2485816832
SPACE_RECLAIMABLE                    : 19995648
NUMBER_OF_FILES                      : 35
-----
```

How much space can be made available through delete of "obsolete, redundant or low priority files"

Total number of archive and flashback logs in the recovery area

© 2005 SkillBuilders, Inc.

A new view, `V$RECOVERY_FILE_DEST`, is available to describe the current flash recovery area. Here we can quickly find space allocated, used and "reclaimable". Reclaimable space can be made available via the database deleting "obsolete, redundant or low priority files." Hopefully, Oracle will choose wisely!




1.28

## Flashback Database...

- Alternative to point-in-time recovery
  - Very easy
- Performance versus point-in-time recovery
  - Often faster
  - Reapply changed blocks versus restore then recover
- Can re-flashback more than once
  - Flashback, then Query database
    - All OK?
    - If so, OPEN RESETLOGS

© 2005 SkillBuilders, Inc.

Oracle10g introduces an easy alternative to using RMAN (or other methods) to perform a point-in-time recovery: Flashback Database. It is easy to use (as you will see) and – after you have flashed back the database – you can query the database, confirm you like what you see, then use OPEN RESETLOGS to make the database generally available. If you do not like the point in time you have flashed back to, simply flash back again.



1.29

## ...Flashback Database

- Use *flashback logs* to recover database
  - Contain changed blocks
  - Ongoing creation of new logs to capture changes
  - Written to “Flash Recovery Area”
  - Transactions dictate frequency / size of logs
  - Open questions
    - Exactly what initiates log write? What is threshold?
  - Automatic deletion of obsolete logs

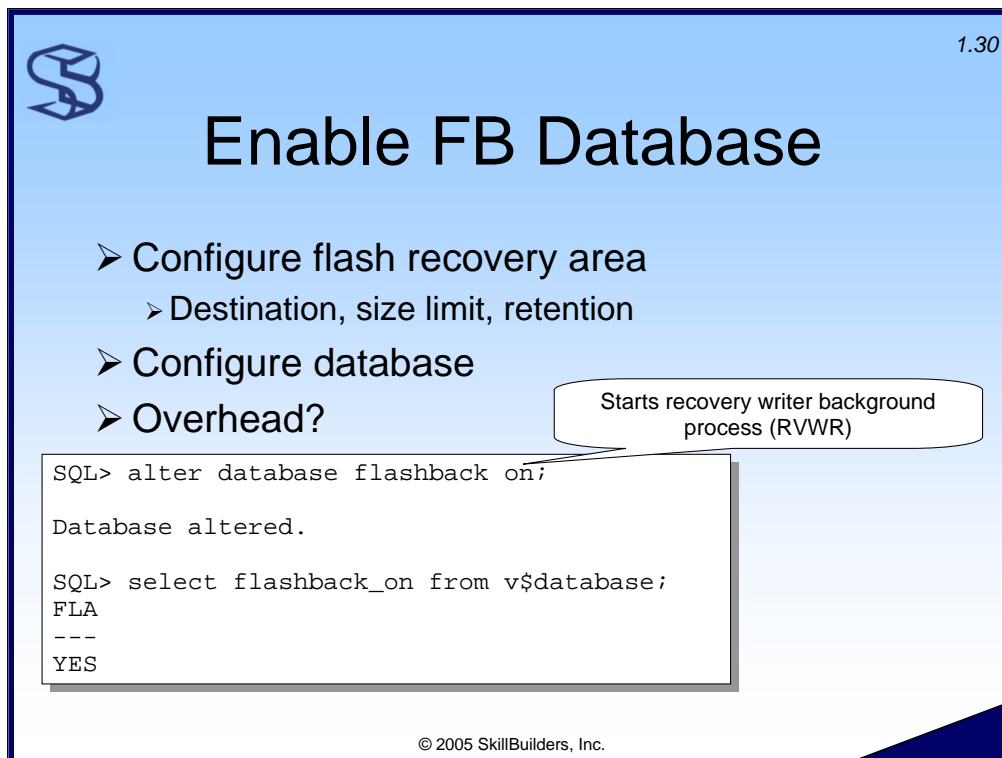
© 2005 SkillBuilders, Inc.

Flashback logs are written to the flash recovery area if the database is configured for flashback database support. (We will see how to do that next.) The logs contain changed blocks.

Though it is clear that the transaction rate on your database will have an affect on the number and possibly size of the logs, I was not able to determine what exactly initiates a flashback log write. Nor was I able to determine what the size of a log will be.

Logs are automatically deleted when:

- the flash recovery area becomes constrained
- the log is not needed for a recovery within the window specified in the `DB_FLASHBACK_RETENTION_TARGET` parameter.



1.30

## Enable FB Database

- Configure flash recovery area
  - Destination, size limit, retention
- Configure database
- Overhead?

```
SQL> alter database flashback on;

Database altered.

SQL> select flashback_on from v$database;
FLA
---
YES
```

Starts recovery writer background process (RVWR)

© 2005 SkillBuilders, Inc.

The database must be configured for flashback database support. In addition to configuring a flash recovery area, you must alter the database with the FLASHBACK ON clause. This starts a new background process (RVWR) to write the logs.

What is the overhead of configuring the database for flashback database support (i.e. creating a flash recovery area where flashback logs are written)? Oracle says negligible. However, my initial tests showed measurable decrease in the performance of my test application. HOWEVER I HAVE NOT DONE ENOUGH TESTING TO FULLY VERIFY THIS. You must test.

The screenshot shows a SQL\*Plus session with the following commands and responses:

```
SQL> startup mount
SQL> flashback database to scn 1427369;
Flashback complete.
SQL> alter database open read only;
Database altered.
SQL> alter database open resetlogs;
Database altered.
```

Callouts provide additional information:

- "Flashback to SCN or timestamp. Need SYSDBA privilege." points to the flashback command.
- "Can open READ ONLY to check things out" points to the "alter database open read only;" command.
- "If necessary, re-mount, flashback to different scn or time" points to the "alter database open read only;" command.
- "Resetlogs deletes old FB logs" points to the "alter database open resetlogs;" command.
- "NOW DO FULL BACKUP!" points to the "alter database open resetlogs;" command.

© 2005 SkillBuilders, Inc.

Here is a simple example of using flashback database. Here, I flashback to an SCN which I might have gotten from a flashback version query (or logminer).

### Supplemental Notes

Offline datafiles are *not* flashed back. However, if there are referential integrity issues because you have not flashed back one or more datafiles, you will receive the error:

```
ORA-01152: file 4 was not restored from a sufficiently old backup
```

So, essentially, you cannot do this – you will have to either recover or drop the file. Thus I recommend you group referentially dependent tables in the same tablespace if possible.



1.32

## Flashback Summary...

- Flashback Query
  - Session or sub-statement level
  - Query data at a previous point-in-time
- Flashback Version Query
  - “VERSIONS BETWEEN” clause
  - Show changes made by transactions
  - Details about transaction
- Flashback\_Transaction\_Query View
  - Access UNDO records, including undo-SQL

© 2005 SkillBuilders, Inc.





1.33

## ...Flashback Summary

- Flashback Table
  - “Reset” table to previous point-in-time
  - Recover from dropped table
- Flashback Database
  - Point-in-time recovery for entire database
  - Must configure database for this...

© 2005 SkillBuilders, Inc.



1.34

## Resources

- Oracle10g SQL Reference
- Application Developers Guide – Fundamentals
- Oracle10g Database Reference

© 2005 SkillBuilders, Inc.



1.35

# Thank You

- Please keep SkillBuilders in mind for your training and consulting needs:
  - Oracle
  - XML
  - Java / J2EE
  - Web Services
  - Web Development
- Contact: Gary Belke
- 888-803-5607
- [www.skillbuilders.com](http://www.skillbuilders.com)

© 2005 SkillBuilders, Inc.