ORACLE

ORACLE THIRTY YEARS 30

Oracle Database 11g - Security

Dražen Patarić
Senior Sales Consultant

Tablespace Encryption

- New at-rest data encryption feature in Oracle11g
- Based on block level encryption that encrypts on writes and decrypts on reads
- Data is encrypted/decrypted at the I/O (block) level and *not* in memory (unlike TDE, which performs the encryption in the PGA of the server process)
- Only encryption penalty is associated with I/O, so encryption performance overall is better than for TDE
- SQL access paths are unchanged and all data types are supported (could be some I/O penalty assigned by the CBO, however)

How Tablespace Encryption Works

- Data blocks are encrypted to and from the datafiles, the data is *not* encrypted in the buffer cache.
- Therefore, all encryption is performed at the I/O level
- Blocks that come from an encrypted tablespace are tracked so that when these rows are sent to temporary tablespaces the data is encrypted
- Data is also kept encrypted in undo tablespaces as well

Data Type Support

- There are no data type restrictions with tablespace data encryption!
- Since we are encrypting the entire tablespace at the block level, all Oracle provided data types are supported for encryption with this feature
  - This includes object types, LOBs, and even LONGs

Key Management

- Master encryption key maintained in an external wallet, in the same way as used by TDE
- There is a single encryption key maintained per tablespace, physically stored in the datafile header block
- For encrypted tablespaces that span more than one datafile, the tablespace key is stored redundantly in the headers of each datafile
- Master encryption key is used to decrypt the tablespace encryption key

More On Key Management

- Master wallet key can still be rotated like we have done with TDE (i.e., via ALTER SYSTEM SET ENCRYPTION KEY command)

- Currently it is not possible to rekey the tablespace-level encryption key
  - ALTER TABLE MOVE TABLESPACE …
  - ALTER INDEKS REBUILD TABLESPACE …

Oracle11g Plans For Rekeying TSE Key With HSMs

- Release 11.1.0.7 -- Will offer 'limited' support for tablespace encryption master keys in HSM devices
  - Customers will not be able to migrate an existing, wallet-based master key for encrypted tablespaces to an HSM based master key
  - In order to use HSM for tablespace encryption, customers must create (not migrate) a new master key.
- Release 11.2 -- Will offer full HSM support (create, migrate, rotate, expire/destroy keys)

So What's A Tablespace Key?

- Think of a tablespace key as "essentially" a table key except that it:
  - Applies to the entire tablespace and not just to a table
  - Isn't stored in the data dictionary like table keys

    Datafile Header
  - Has a separate master key that is stored in the external wallet

Example: Tablespace Encryption

1. Create or open the encryption wallet

```
SQL> ALTER SYSTEM SET ENCRYPTION KEY IDENTIFIED BY
    "welcome1";
```

2. Create a tablespace with the encryption keywords

```
SQL> CREATE TABLESPACE data02_enc
  2> DATAFILE '$ORACLE_HOME/dbs/data02.dbf' SIZE 100M
  3> ENCRYPTION USING '3DES192'
  4> DEFAULT STORAGE (ENCRYPT);
```

Restrictions

- Temporary and undo tablespaces cannot be encrypted (although selected blocks are encrypted)
- BFILEs and external tables are not encrypted
- Transportable tablespaces across different endian platforms is not supported
- Does not work with HSM
- Encryption key for an encrypted tablespace cannot be changed
  - *Current workaround*: ALTER OBJECT MOVE …

Supported Encryption Methods

- Supported algorithms:
  - 3DES168
  - AES128 (default for encrypted tablespaces)
  - AES192
  - AES256
- Query the dynamic view V$ENCRYPTED_TABLESPACES for current tablespace encryption settings (encrypted/nonencrypted, current algorithm)

Protection Of Data In Encrypted Tablespaces

- All encrypted data is protected during operations like joins, sorts, and merges
- Implication is that encrypted the data is safe when it is moved to temporary tablespaces for interim processing by the Oracle server
- Data in undo and redo logs is also protected

Indexes And Tablespace Encryption

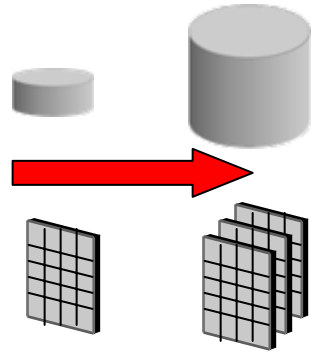
- Tablespace encryption keeps plain text versions of index blocks in the SGA (unlike column-level TDE, which keeps the encrypted index values in SGA)
- This is why tablespace encryption is able to use indexes for bounded/unbounded range queries, indexes for joins, etc.
- Being able to use indexes widely has shown tablespace encryption to outperform TDE
- I/O still a big performance contributor since tablespace encryption encrypts at block level

Using TDE Versus Tablespace Encryption

- Use TDE if:
  - Very few of table columns require encryption
  - Encrypted columns are not foreign keys or require non-equality searches, or are not searchable fields
  - Use of HSMs for protecting the external wallet is required
  - Flexible rekeying of stored data is required
  - Your databases are not 11g
- Use tablespace encryption if:
  - Encryption performance is a critical factor
  - Most table data requires encryption
  - Use of indexing on encrypted columns is important
  - You require data type support not provided by TDE (such as LONGs)

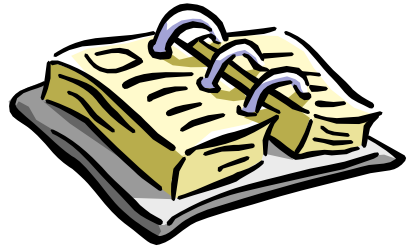
Data History and Retention

- Data retention and change control requirements are growing
  - Regulatory oversight and Compliance
    - Sarbanes-Oxley, HIPAA, Basel-II, Internal Audit
  - Business needs
    - Extract “temporal” dimension of data
    - Understand past behavior and manage customer relationships profitably
  - Change Control and Recovery
- Failure to maintain appropriate history & retention is expensive
  - Legal risks
  - Loss of Reputation
- Current approaches to manage historical data are inefficient and often ineffective



Data History and Retention - Requirements

- Historical data needs to be secure and tamper proof
  - Unauthorized users should not be able to access historical data
  - No one should be able to update historical data
- Easily accessible from existing applications
  - Seamless access
  - Should not require special interfaces or application changes
  - Minimal performance overhead
- Optimal Storage footprint
  - Historical data volume can easily grow into hundreds of terabytes
- Easy to set up historical data capture and configure retention policies



Managing Data History – Current Approaches

- Application or mid-tier level
  - Combines business logic and archive policies
  - Increases complexity
  - No centralized management
  - Data integrity issues if underlying data is updated directly
- Database level
  - Enabled using Triggers
  - High performance and maintenance overhead
- External or Third-party
  - Mine redo logs
  - History stored in separate database
  - May not be able to seamlessly query OLTP and history data
- None of the above approaches meet all customer requirements
  - Customers are therefore forced to make significant compromises

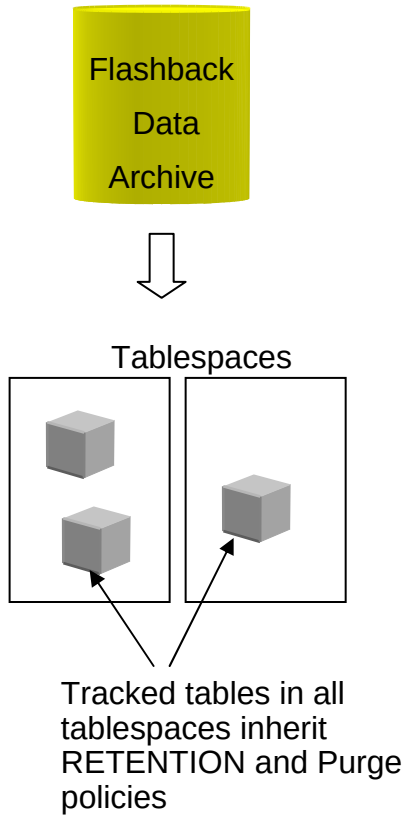
Total Recall

- **Transparently** tracks historical changes to all Oracle data in a highly secure and efficient manner
  - Historical data is stored in the database and can be retained for as long as you want
  - Special kernel optimizations to minimize performance overhead of capturing historical data
  - Historical data is stored in compressed form to minimize storage requirements
  - Automatically prevents end users from changing historical data
- Seamless access to archived historical data
  - Using “AS OF” SQL construct

```
select * from product_information AS OF TIMESTAMP
'02-MAY-05 12.00 AM‘ where product_id = 3060
```

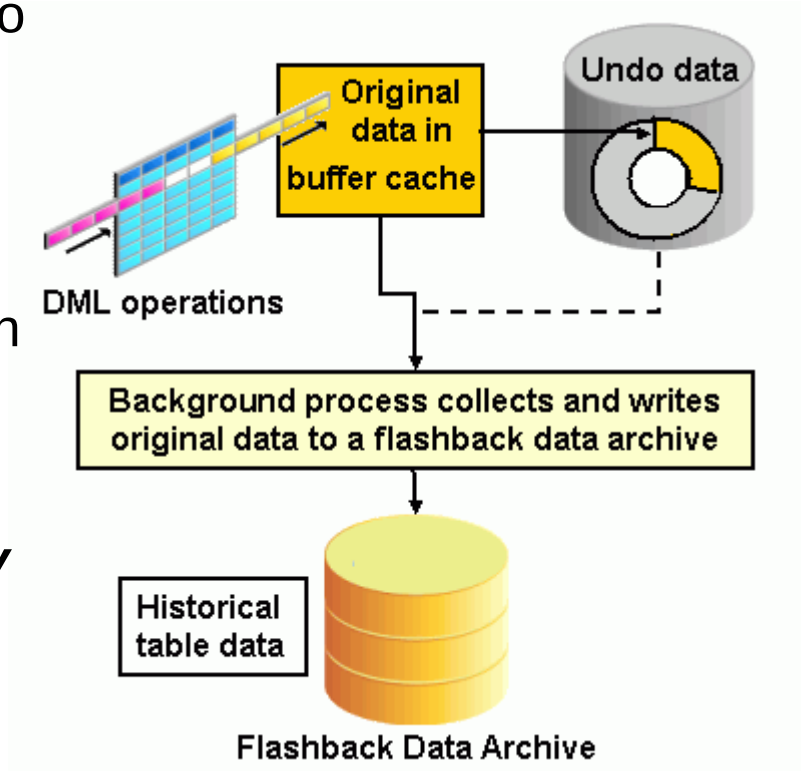
Historical Data Storage

- A new database object, flashback data archive, is a logical container for storing historical information
- Consists of one or more tablespaces
  - 'QUOTA' determines max amount of space a flashback data archive can use in each tablespace (default is Unlimited)
- Specify duration for retaining historical changes using 'RETENTION' parameter
- Tracks history for one or more tables
  - Tables should share the archiving characteristics
- Automatically purges aged-out historical data based on retention policy
- Create as many flashback data archives as needed
  - Group related tables by desired retention period



How Does Flashback Data Archive Work?

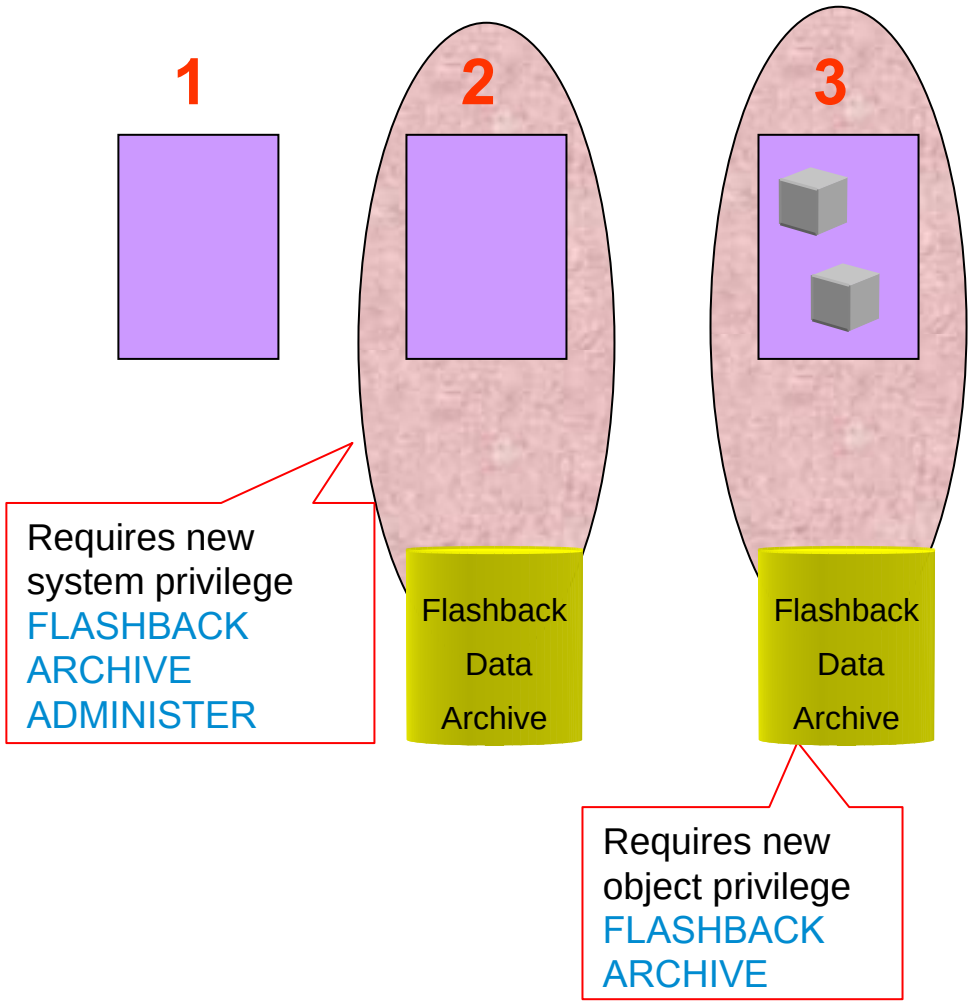
- Primary source for history is the undo data
- History is stored in automatically created history tables inside the archive
- Transactions and its undo records on tracked tables marked for archival
  - Undo records not recycled until history is archived
- History is captured ***asynchronously*** by new background process (fbda)
  - Default capture interval is 5 minutes
  - Capture interval is self-tuned based on system activities
  - Process tries to maximize undo data reads from buffer cache for better performance
  - INSERTs do not generate history records



Flashback Data Archive And DDLs

- Flashback Data Archive guarantees historical data capture and maintenance
  - Any operations that invalidates history or prevents historical capture will be disallowed
- Automatically disallows DDL on tracked tables that invalidates history
  - Dropping and truncating a tables
  - Dropping or modifying a column
  - Possible to add columns to tracked tables
- Must disable archiving before performing any major changes
  - Disabling archiving discards already collected history

Creating Flashback Data Archive & Enable History Tracking



1. Optional - create tablespace (ASSM and Automatic Undo is required)
2. Create a flashback data archive
   - Set the retention period

   ```
   CREATE FLASHBACK
       ARCHIVE fda1
   TABLESPACE tbs1
   RETENTION 5 YEAR;
   ```
3. Enable archiving on desired tables

   ```
   ALTER TABLE EMPLOYEES
   FLASHBACK ARCHIVE fda1;
   ```

Managing Flashback Data Archive

- Static data dictionary views
  - *_FLASHBACK_ARCHIVE - Displays information about Flashback Data Archives.
  - *_FLASHBACK_ARCHIVE_TS - Displays tablespaces of Flashback Data Archives.
  - *_FLASHBACK_ARCHIVE_TABLES - Displays information about tables that are enabled for flashback archiving.
- Alerts generated when flashback data archive is 90% full
- Automatically purges historical data after expiration of specified retention period
- Supports ad-hoc purge by administrators (privileged operation)
  - ALTER FLASHBACK ARCHIVE fla1 PURGE BEFORE TIMESTAMP (SYSTIMESTAMP - INTERVAL '1' DAY);

Managing Flashback Data Archive

- SYS_FBA_HIST_* - Internal History Table
  - Replica of tracked table with additional timestamp columns
  - Partitioned for faster performance
  - Indexes are NOT automatically replicated from tracked table - tune performance using indexes
  - Compression reduces disk space required
  - No modifications allowed to internal partitions
- Applications don't need to access internal tables directly
  - Use 'AS OF' to seamlessly query history

Summary

- Managing historical data should no longer be a difficult task
- Total Recall provides a secure, efficient, easy to use and applicant transparent solution
  - Easy to implement
  - Centralized, Integrated and query-able
  - Highly storage and performance efficient
  - Automatic, Policy-based management
- Reduce costs of compliance
- Can be used for variety of other purposes
  - Auditing, Human error correction, etc.


Q&A
QUESTIONS
ANSWERS

ORACLE IS THE INFORMATION COMPANY