



**ORACLE®**

# Oracle Rdb V7.3

Preliminary Feature Overview

# Overview

- Development Objectives
- Summary of each feature
- Questions?

# Disclaimer

- The following presentation describes features which have not been released
- Oracle Corporation is describing this information for planning purposes
- No commitment or guarantee of such functionality is implied

# Development Objectives

- Parallel development with Rdb V7.2
- Infrastructure changes (mostly not visible)
  - Such as using 64 bit language compilers
  - Moving structures to P2 space
- Change default behavior
  - Most defaults established in 1984 for smaller memory and slower CPU and I/O system
  - Better default behavior for modern systems
- Long cycle features
  - Typically small features get added to .1 releases
  - Some require longer prototype, development, test cycle

# Feature Overview

- Can't describe every feature
- Hope to give broad insight into future features
- Note that some features might appear in 7.3.2 rather than the initial release



# SQL Compatibility

# SQL features for Oracle RDBMS Compatibility

- SQL/Services and the OCI Interface must process SQL syntax from Oracle client applications
- Functions from Oracle
  - BIN\_TO\_NUM
  - NUMTODSINTERVAL
  - NUMTOYMINTERVAL
  - SYS\_GUID (ported to 7.2.5)
  - SYSTIMESTAMP (ported to 7.2.5)
  - And more

# New statements

- REPLACE statement
  - Based on MySQL syntax
- Acts like INSERT but will preserve uniqueness for PRIMARY KEY using a pre-DELETE action
- If there is no PRIMARY KEY it is a simple INSERT
- Activates BEFORE and AFTER INSERT trigger
- Activates BEFORE and AFTER DELETE trigger if a primary key value exists
- Valid statement for CREATE TRIGGER

# REPLACE without a PRIMARY KEY

```
SQL> select * from work_status;
  STATUS_CODE   STATUS_NAME   STATUS_TYPE
 0              INACTIVE     RECORD EXPIRED
 1              ACTIVE       FULL TIME
 2              ACTIVE       PART TIME
3 rows selected
SQL>
SQL> replace into WORK_STATUS
cont> values ('0', 'INACTIVE', 'RECORD EXPIRED');
1 row replaced
SQL> select * from work_status;
  STATUS_CODE   STATUS_NAME   STATUS_TYPE
 0              INACTIVE     RECORD EXPIRED
 1              ACTIVE       FULL TIME
 2              ACTIVE       PART TIME
 0              INACTIVE     RECORD EXPIRED
4 rows selected
SQL> rollback;
```

# REPLACE example with PRIMARY KEY

```
SQL> alter table WORK_STATUS
cont>      add constraint PK_WORK_STATUS
cont>      primary key (STATUS_CODE)
cont>      not deferrable;
SQL>
SQL> replace into WORK_STATUS
cont> values ('0', 'INACTIVE', 'RECORD EXPIRED');
1 row replaced
SQL> select * from work_status;
  STATUS_CODE    STATUS_NAME    STATUS_TYPE
-----
1              ACTIVE        FULL TIME
2              ACTIVE        PART TIME
0              INACTIVE      RECORD EXPIRED
3 rows selected
```

# Usage of REPLACE

- New qualifier on RMU Load
- /REPLACE\_ROWS
- Used to load unique rows only
- Must decide if /NOTRIGGERS is appropriate to avoid BEFORE/AFTER INSERT actions

```
$ rmu/load/replace mf_personnel_sql employees employees
```

# MERGE statement

- Based on Oracle and SQL Standards syntax
- Allows conditional INSERT or UPDATE based on rules specified in the statement

# MERGE example

```
SQL> merge into sales_fact d
cont> using sales_jul01 s
cont> on (d.time_id = s.time_id
cont>   and d.store_id = s.store_id
cont>   and d.region_id = s.region_id)
cont> when matched then
cont>   update
cont>   set d.parts = d.parts + s.parts,
cont>       d.sales_amt = d.sales_amt + s.sales_amt,
cont>       d.tax_amt = d.tax_amt + s.tax_amt,
cont>       d.discount = d.discount + s.discount
cont> when not matched then
cont>   insert (d.time_id, d.store_id, d.region_id,
cont>           d.parts, d.sales_amt, d.tax_amt, d.discount)
cont>   values (s.time_id, s.store_id, s.region_id,
cont>           s.parts, s.sales_amt, s.tax_amt, s.discount);
```

# Savepoint support

- **SAVEPOINT** allows for a small inner part of a transaction to be named and managed.
- **RELEASE SAVEPOINT** is used to discard the controlled changed
- **ROLLBACK TO SAVEPOINT** is used to undo part of the current transaction

# SAVEPOINT example

```
set flags 'TRANSACTION';
begin
set transaction read write;

insert into SV_X values (10);

savepoint D;
insert into SV_X values (20);
insert into SV_X values (30);
insert into SV_X values (40);

rollback to savepoint D;

insert into SV_X values (50);
insert into SV_X values (60);

commit;
end;
```

# SAVEPOINT example

```
~T Start_transaction (8) on db: 1, db count=1
~T Savepoint "D" (8.2) on db: 1
~T Rollback to Savepoint "D" (8.2) on db: 1
~T Commit_transaction (8) on db: 1
~T Prepare_transaction (8) on db: 1
```

```
select * from SV_X order by V;
       V
       10
       50
       60
3 rows selected
```

# Savepoint support

- Each **savepoint** is given a unique name which is specific to a database attach
- If multiple database ALIAS are in use then use the ALIAS name to provide context.
- `SAVEPOINT db1.mysavepoint`
- If the **savepoint** name is re-used while active then the **savepoint** is implicitly released.
- If one of the ORACLE dialects is active then a new SAVEPOINT is established (as per Oracle behavior).

# Savepoint support

- Currently Rdb limits a session to just one active **savepoint**
- Extending this to multiple nested savepoints is planned for a future release
- Current limit imposed because of complexity of in memory metadata



# Security

# Audit Vault

- Changes the way Rdb handles OpenVMS privileges
- ALTER DATABASE ... AUDIT VAULT IS ENABLED;
- Requires SECURITY privilege to enable
- No longer inherit overrides from current system user
- **\*WARNING\*** Ensure there is a SECURITY user defined in the database because once AUDIT VAULT is enabled you can not use a VMS privilege to change it externally

# Encryption

- Transparent data encryption
- Table data is encrypted at-rest
- New ENCRYPTION USING clause for CREATE STORAGE AREA
- Decryption requires KEY value to be specified by RMU/OPEN

# Audit Support

- Now support audit of functions, procedures, modules and sequences
- New RMU/DUMP/AUDIT command
  - See later discussion



# Default Changes

## Expand capabilities

- COUNT statistics function return BIGINT values
- AVG (average) return DOUBLE PRECISION values and uses a BIGINT counter.
- STDDEV, and VARIANCE return DOUBLE PRECISION values and uses a BIGINT counter
- All DIVIDE operations return DOUBLE PRECISION results (ported to 7.2.5)

# Goal to improve database performance

- PERCENT FILL default increased from 70% to 85%
- Default page size increased from 2 to 4 blocks
- Default number of buffers increase from 20 to 250
- Default number of recovery buffers increase from 20 to 250
- Default buffer size increase from 3 to 4
- Default NODE SIZE increased to better utilize target page size

## Only changing defaults

- Fully specified definitions will see no differences in behavior
- For example an SQL EXPORT DATABASE will save these attributes in V7.2 and SQL IMPORT DATABASE will replicate them in the new database
- RMU/BACKUP and RMU/RESTORE, RMU/COPY, and so on will not change these values in existing databases



# Utility Improvements



# SQL EXPORT and IMPORT

# EXPORT with compression

- Table data is by default stored with run-length compression
- VARCHAR columns are exported with string length and full field width (allows fast IMPORT)
- EXPORT DATABASE often can produce an interchange file much larger the actual database

# EXPORT with compression

- EXPORT DATABASE will now compress table, and LIST OF BYTE VARYING column data
- Goal is to produce smaller interchange file (.rbr)
- Metadata is not compressed, may get further compression using ZIP or similar tools

# IMPORT

- `IMPORT DATABASE` will automatically decompress data during import



# RMU Commands

# RMU Unload now supports CSV formatted files

- CSV – comma separated values – can be read using Microsoft EXCEL and similar tools
- CSV includes an initial row with column headers
- Allows easier export of data to reporting and charting tools

# RMU/SET AUDIT

- Supports MODULE, ROUTINE and SEQUENCE audit
- MODULE audit is shortcut for all contained routines
- ROUTINE can be used for both external and SQL routines
- System sequences, modules and routines are not audited



# In Closing...

# Summary

- Changes to the product across all levels
- Internal restructuring for 64 bit support
- New syntax and capabilities
- Much more that has not been described
- Some number of changes were back ported to 7.2.5 after extensive testing in 7.3



# Questions?