
ORACLE®

# RMU Extract Secrets

## A database administrator tool

Ian Smith
Oracle Rdb Product Architect
Oracle Rdb Engineering
October, 2014

ORACLE

# Remote Speaker Photo

**Ian Smith**

Rdb Product Architect, Oracle

# Program Agenda

1. Introduction to Extract
2. Special Purpose Scripting
3. Case Studies
4. Miscellaneous
5. Final Comments

# Program Agenda

1. Introduction to Extract
2. Special Purpose Scripting
3. Case Studies
4. Miscellaneous
5. Final Comments

# Program Agenda

1. Introduction to Extract
2. Special Purpose Scripting
3. Case Studies
4. Miscellaneous
5. Final Comments

# Program Agenda

1. Introduction to Extract
2. Special Purpose Scripting
3. Case Studies
4. Miscellaneous
5. Final Comments

# Program Agenda

1. Introduction to Extract
2. Special Purpose Scripting
3. Case Studies
4. Miscellaneous
5. Final Comments

# Program Agenda

1. Introduction to Extract
2. Special Purpose Scripting
3. Case Studies
4. Miscellaneous
5. Final Comments

# Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

# Introduction

## History and goals of RMU Extract

# Notes

- This isn’t a qualifier by qualifier description of RMU Extract
- However, I’ll present some highlights and qualifier combinations you may not have seen before
- Some background to Extract development

# Extract

- Based on original RDO_EXTRACT created during Rdb V1.1 training session in Sydney, Australia
- Enhanced as tool used for consulting and support
- New rewritten version became part of Rdb V4.0
- Under the RMU command but maintained as a separate image RMUEXTRACTnn.EXE

# What does it do?

- Metadata reader and formatter
- Database definitions are stored as BLR, MBLR and structures in the root file
- Extract reads various structures and creates SQL definitions

# BLR

- Binary Language Representation
- Syntax of pure query
  - Some SQL sematic information is discarded
- Often structured differently to either original RDO or SQL query
- Contained full procedural language a decade before SQL supported stored procedures
- SET FLAGS ‘BLR,NOPREFIX’

# Could be generated by either query

```
$ rdo @a
RDO> database filename sql$database
RDO>
RDO> for w in work_status
cont>    sorted by w.status_code
cont>    print w.status_name;
cont> end_for;
 STATUS_NAME
 INACTIVE
 ACTIVE
 ACTIVE
RDO>
RDO> finish
```

# Could be generated by either query

```
$ sql$ @b
SQL> attach 'filename sql$database';
SQL>
SQL> select status_name
cont> from work_status
cont> order by status_code
cont> ;
 STATUS_NAME
 INACTIVE
 ACTIVE
 ACTIVE
3 rows selected
SQL>
SQL> disconnect all;
```

Simple query to select a column, sort by a different column

# Example

```
(VERSION 4
 BLR$K_BEGIN
  BLR$K_MESSAGE 1 2
   DSC$K_DTYPE_T 1
   DSC$K_DTYPE_T 11
  BLR$K_MESSAGE 2 0
  BLR$K_FOR
   BLR$K_RSE  1
    BLR$K_RELATION WORK_STATUS 1
    BLR$K_SORT  1
    BLR$K_ASCENDING
     BLR$K_FIELD 1 STATUS_CODE
   BLR$K_END
   BLR$K_BEGIN
    BLR$K_SEND  1
     BLR$K_BEGIN
      BLR$K_ASSIGNMENT
       BLR$K_LITERAL
        DSC$K_DTYPE_T 1      "N"
       BLR$K_PARAMETER 1 0
```

Table name

Sorting column name

# Example

```
      BLR$K_ASSIGNMENT
         BLR$K_FIELD 1 STATUS_NAME
         BLR$K_PARAMETER 1 1
      BLR$K_END
   BLR$K_END
  BLR$K_SEND  1
   BLR$K_BEGIN
    BLR$K_ASSIGNMENT
      BLR$K_LITERAL
       DSC$K_DTYPE_T 1      "Y"
      BLR$K_PARAMETER 1 0
   BLR$K_END
 BLR$K_END
BLR$K_EOC)
```

Selected column name

BLR$ tags are byte codes, not keywords

# BLR decoder

- Contains a BLR decoder
- Can generate output syntax for RDO (if possible) and SQL (mostly possible)
- Restricted to the subset used by constraints, triggers, default values, computed by, automatic columns, views, and so on
- Due to complexity stored routines are extracted using the original SQL sources
  - Sometimes artifacts of source capture mean they require editing

# MBLR

- Metadata Binary Language Representation
- Language independent attribute and action definition
- Stored in Rdb$EXTENSION_PARAMETERS or Rdb$DATABASE_PARAMETERS special LIST OF BYTE VARYING columns
- SET FLAGS ‘MBLR,NOPREFIX’

# Example.

## MBLR definition with BLR fragment

```
SQL> set flags 'mblr,noprefix';
SQL> create domain creator_username varchar(31)
cont> default lower (current_user);
MBLR$K_VERSION_1     (This is a DEFINE)
  MBLR$K_FIELD_DEF  "CREATOR_USERNAME"
    MBLR$K_DESCRIPTOR
      DSC$K_DTYPE_VT 31
    MBLR$K_DEFAULT_VALUE  "18"
      BLR$K_VERSION4
        BLR$K_TRANSLATE "RDBVMS$LOWER"
          BLR$K_GET_INFO (10) RDB$K_INF_CURRENT_USER
  MBLR$K_FIELD_END
```

# Interface to other structures

- Rdb$DATABASE_INFO
  API interface to return database, cache, journal, storage area attributes
- Rdb$EXTENSION
  API interface to return formatted ACL (access control list), CSET (character set attributes), and so on
  - Note: must resolve ACL on remote system so that UIC's correctly mapped

# Generating CREATE INDEX

- Must read Rdb$INDICES and Rdb$INDEX_SEGMENTS tables
  - Provides columns and column attributes
  - **size is**, **mapping values**, **desc**, **asc**, …
- Must read Rdb$STORAGE_MAPS, and Rdb$STORAGE_MAP_AREAS tables
  - Provides **store** … **in** … **with limit of** clauses
- Rdb$EXTENSION_PARAMETERS (LoBV column)
  - Provides attributes such as **node size**, **percent fill**, etc
- Call RDB$DATABASE_INFO to get THRESHOLDS for the logical areas

# RMU Extract

- Original goals
  - Migration tool from RDO to SQL interface
  - Allow definitions to be converted to SQL scripts
  - Document existing databases
- Support all RDO and SQL objects
- See /LANGUAGE qualifier
- SQL language now far more complex than RDO so extract as RDO is severely limited

# Using Extract for special purposes

# Item

- Typically use /ITEM=ALL to get whole database
- ALL doesn’t mean ALL
- Many more keywords for Option

# Security and Audited Items

- PROTECTION (grant)
- REVOKE_ENTRY (revoke)
- SECURITY (auditing)
- LOAD
  - Creates a DCL script to unload a table
- UNLOAD
  - Creates a DCL script to load a table

# Security and Audited Items

- VERIFY
  - Creates a DCL script to verify the database (see RMU documentation)
- WORKLOAD
  - Creates DCL script that can be applied to a new database
  - Contains all the workload data from the source
  - Very useful when gathering workload data from production and experimenting in testing area

# Alter Database and Import

- ALTER_DATABASE
  - Generate the JOURNAL attributes and all the journal definitions
- IMPORT
  - Generate an **import database** statement
  - Use /Option keywords like FILENAME_ONLY
  - Use /Defaults to specify **allocation** and **snapshot_allocation**

# Annotation Options

- AUDIT
  - Requests that creation and alter timestamps be displayed with each object along with the creating username
- HEADER
  - Requests that a header be added to each section

# Case Study

- Part of the daily maintenance of the production system is to use
- RMU /Extract /Option=(Header,Audit) –
  /Item=(All,Protection,Security)
- Audit adds extra details for these objects
- Report contains current state of the database metadata

# Fine Tuning

# Header

- Header identifies section being extracted
- Also name of the source database and time stamp of the report
- Often considered noise
- Use NOHEADER option

# FILENAME_ONLY

- File file specification extracted
  - Database name
  - Storage Area Names
- If creating testing database scripts ask for FILENAME_ONLY

# Example

**$ rmu/extract/item=module/option=(noheader) personnel**

```
set verify;
set language ENGLISH;
set default date format 'SQL92';
set quoting rules 'SQL92';
set date format DATE 001, TIME 001;
attach 'filename $1$DGA170:[TEST.DATABASES]PERSONNEL.RDB';
-- no modules defined
```

# Example

**$ rmu/extract … /option=(noheader,filename_only) personnel**

```
set verify;
set language ENGLISH;
set default date format 'SQL92';
set quoting rules 'SQL92';
set date format DATE 001, TIME 001;
attach 'filename PERSONNEL';
-- no modules defined
```

# Example

**$ rmu/extract … /option=(nodomains,match:candidates%) …**

```
create table CANDIDATES (
   LAST_NAME
      CHAR (14),
   FIRST_NAME
      CHAR (10),
   MIDDLE_INITIAL
      CHAR (1),
   CANDIDATE_STATUS
      VARCHAR (255)
      constraint CANDIDATE_STATUS_1
         check((((CANDIDATE_STATUS is null)
            or (CANDIDATE_STATUS > '    '))
            or (CANDIDATE_STATUS = '    ')))
         not deferrable);
```

# Example

**$ rmu/extract/item=(table,noconstraint)/option=(…) personnel**

```
create table CANDIDATES (
    LAST_NAME
        CHAR (14),
    FIRST_NAME
        CHAR (10),
    MIDDLE_INITIAL
        CHAR (1),
    CANDIDATE_STATUS
        VARCHAR (255));

commit work;
```

Use noconstraint if you don't want them

# Tables

# Tables

- Generates the definitions as you would expect
- However, a table is rarely isolated
  - Indices
  - Storage Maps
  - Indices with **placement via index** clause
- Use GROUP_TABLE to create family grouping
- Also note the MATCH option used in this example
  - Support LIKE wildcard matching “_” and “%”
  - Don’t forget names are stored with trailing spaces

# Example

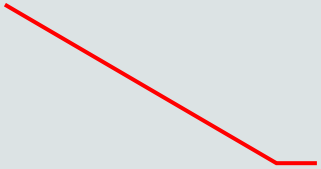
**$ rmu/extract/item=(table,storage,index) –**
**/option=(noheader,match=employees%,group_table) …**

```
create table EMPLOYEES (
    EMPLOYEE_ID ID_NUMBER,
    LAST_NAME LAST_NAME,
    FIRST_NAME FIRST_NAME,
    MIDDLE_INITIAL MIDDLE_INITIAL,
    ADDRESS_DATA_1 ADDRESS_DATA_1,
    ADDRESS_DATA_2 ADDRESS_DATA_2,
    CITY CITY,
    STATE STATE,
    POSTAL_CODE POSTAL_CODE,
    SEX SEX,
    BIRTHDAY STANDARD_DATE,
    STATUS_CODE STATUS_CODE);
```

# Example

## $ rmu/extract/item=(table,storage,index) – /option=(noheader,match=employees%,group_table) …

```
create unique index EMPLOYEES_HASH
    on EMPLOYEES (
    EMPLOYEE_ID)
    type is HASHED SCATTERED
    comment is
     ' hash index for employees'
    store
       using (EMPLOYEE_ID)
          in EMPIDS_LOW
             with limit of ('00200')
          in EMPIDS_MID
             with limit of ('00400')
          otherwise in EMPIDS_OVER;
```

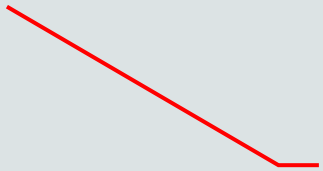


Then index used for placement

# Example

## $ rmu/extract/item=(table,storage,index) – /option=(noheader,match=employees%,group_table) …

```
create storage map EMPLOYEES_MAP
    for EMPLOYEES
    comment is
      ' employees partitioned by "00200" "00400"'
    placement via index EMPLOYEES_HASH
    store
       using (EMPLOYEE_ID)
          in EMPIDS_LOW
             with limit of ('00200')
          in EMPIDS_MID
             with limit of ('00400')
          otherwise in EMPIDS_OVER;
```



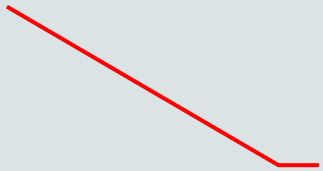
Then storage map for table

# Example

**$ rmu/extract/item=(table,storage,index) –**
**/option=(noheader,match=employees%,group_table) …**

```
create unique index EMP_EMPLOYEE_ID
      on EMPLOYEES (
      EMPLOYEE_ID
         asc)
      type is SORTED
      node size 430;

commit work;
```



All other indices

# Constraints

# Constraints

- Usually extracted as part of TABLE
- **primary key** and **not null** inline
- **check** and **foreign key** defined later using **alter table**

# Example

## $ rmu/extract/item=table/option=match:account% tp

```
create table "ACCOUNT" (
   ACCOUNT_NUMBER
      INTEGER
      default ACCOUNT_ID.nextval
      constraint ACCOUNT_PRIMARY_ACCOUNT_NUMBER
         primary key
         initially immediate not deferrable,
   FIRST_NAME
      VARCHAR (20),
   LAST_NAME
      VARCHAR (20),
   SEX
      CHAR (1),
   ACCOUNT_BALANCE MONEY);

alter table "ACCOUNT"
   add constraint ACCOUNT_CHECK1
      check(("ACCOUNT".ACCOUNT_BALANCE >= 20))
      initially immediate not deferrable;
```

Primary key inline

Check deferred

# Defer_Constraints option

- Extract will defer all the constraints

# Example

## $ rmu/extract … /option=(mat:account%,defer_constraints) …

```
create table "ACCOUNT" (
   ACCOUNT_NUMBER
      INTEGER
      default ACCOUNT_ID.nextval,
   FIRST_NAME
      VARCHAR (20),
   LAST_NAME
      VARCHAR (20),
   SEX
      CHAR (1),
   ACCOUNT_BALANCE MONEY);

alter table "ACCOUNT"
   add constraint ACCOUNT_CHECK1
      check(("ACCOUNT".ACCOUNT_BALANCE >= 20))
      initially immediate not deferrable
   alter column ACCOUNT_NUMBER
      constraint ACCOUNT_PRIMARY_ACCOUNT_NUMBER
         primary key
         initially immediate not deferrable;
```

# Case study

- Want script to define the table first
- Next load that table with data
- Then create the indices on that table
- Next need a script to add in constraints

# Example – create the table

## $ rmu/extract/item=(table,noconstraint)/option=(…,defer) tp

```
create table "ACCOUNT" (
   ACCOUNT_NUMBER
      INTEGER
      default ACCOUNT_ID.nextval,
   FIRST_NAME
      VARCHAR (20),
   LAST_NAME
      VARCHAR (20),
   SEX
      CHAR (1),
   ACCOUNT_BALANCE MONEY);

commit work;
```

# Example – load the table

**$ rmu/extract/item=load/option=(…,filename_only,noheader) tp**

```
$ RMU/LOAD -
                /TRANSACTION_TYPE=EXCLUSIVE -
                TP -
                "ACCOUNT" -
                ACCOUNT.UNL
$
$ EXIT
```

# Example – create the indices

**$ rmu/extract/item=index/opti=(mat:account%,group_table) tp**

```
set transaction read write reserving
   "ACCOUNT" for EXCLUSIVE WRITE;

create unique index ACCOUNT_PRIMARY_ACCOUNT_NUMBER
   on "ACCOUNT" (
   ACCOUNT_NUMBER
      asc)
   type is SORTED RANKED
   node size 430;

commit work;
```

group_table - the MATCH is on table name

# Example – now define the constraints

## $ rmu/extract/item=(notable,constraint)/option=(…,defer) tp

```
alter table "ACCOUNT"
   add constraint ACCOUNT_CHECK1
      check(("ACCOUNT".ACCOUNT_BALANCE >= 20))
      initially immediate not deferrable
   alter column ACCOUNT_NUMBER
      constraint ACCOUNT_PRIMARY_ACCOUNT_NUMBER
         primary key
         initially immediate not deferrable;

commit work;
```

# Miscellaneous

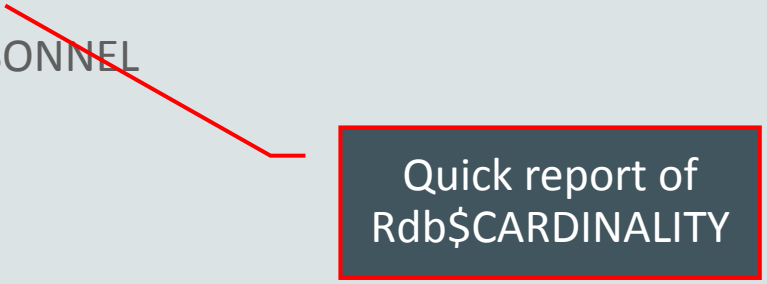
# What is PDL?

- Script language for retired product call Oracle EXPERT
- Generated volume script used for capacity planning
- PDL output might be useful today
- Item=Volume
- Option=Volume_Scan
- Option=Column_Volume

# Example

## $ rmu/extract/item=volume mf_personnel

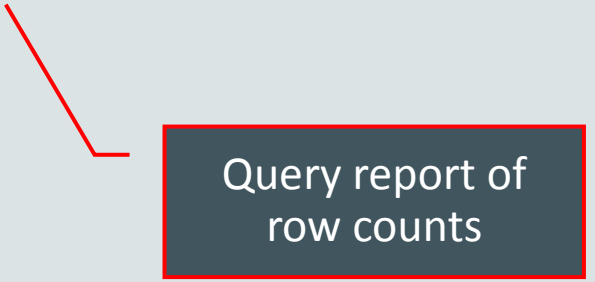
```
-- Volumes are approximate
Volume for schema MF_PERSONNEL
    Default volatility is 5;
    Table CANDIDATES all is 3;
    Table COLLEGES all is 16;
    Table DEGREES all is 165;
    Table DEPARTMENTS all is 26;
    Table EMPLOYEES all is 100;
    Table JOBS all is 15;
    Table JOB_HISTORY all is 274;
    Table RESUMES all is 0;
    Table SALARY_HISTORY all is 729;
    Table WORK_STATUS all is 3;
```



# Example

**$ rmu/extract/item=volume/option=volume_scan mf_personnel**

```
-- Volumes are derived from a full count of data
Volume for schema MF_PERSONNEL
   Default volatility is 5;
   Table CANDIDATES all is 3;
   Table COLLEGES all is 16;
   Table DEGREES all is 165;
   Table DEPARTMENTS all is 26;
   Table EMPLOYEES all is 100;
   Table JOBS all is 15;
   Table JOB_HISTORY all is 274;
   Table RESUMES all is 0;
   Table SALARY_HISTORY all is 729;
   Table WORK_STATUS all is 3;
```



# Example

**$ rmu/extract/item=volume/option=(column,volume_scan) …**

```
-- Volumes are derived from a full count of data
Volume for schema MF_PERSONNEL
   Default volatility is 5;
   Table CANDIDATES all is 3;
      List RESUME
         Cardinality IS 0;
   Table COLLEGES all is 16;
      Column COLLEGE_CODE all is 16;
   Table DEGREES all is 165;
      Column COLLEGE_CODE all is 13;
      Column EMPLOYEE_ID all is 99;
   Table DEPARTMENTS all is 26;
      Column DEPARTMENT_CODE all is 26;
…
```

Also collect index column uniqueness counts

# Order_By_Name

- When extracting definitions the default is to list them in the order they were created
- Use this option to sort names alphabetically
  - row cache, journal and storage area names
- Affects:
  - Database
  - Alter_Database
  - Import

# Example

**$ rmu/extract/item=database … /option=order_by_name**

```
create database …
    …
    create storage area DEPARTMENTS …
    create storage area EMPIDS_LOW …
    create storage area EMPIDS_MID …
    create storage area EMPIDS_OVER …
    create storage area EMP_INFO …
    create storage area JOBS …
    create storage area MF_PERS_SEGSTR …
    create storage area RDB$SYSTEM …
    create storage area SALARY_HISTORY …
…
;
```

# Omit_Disabled

- This option requests that disabled objects not be extracted
  - Indexes – **maintenance is disabled**
  - Constraints and Triggers - **disabled**
  - Users with **account is locked**
- Otherwise these objects are extracted with the disabling syntax

# Final Comments

# Oracle Usage

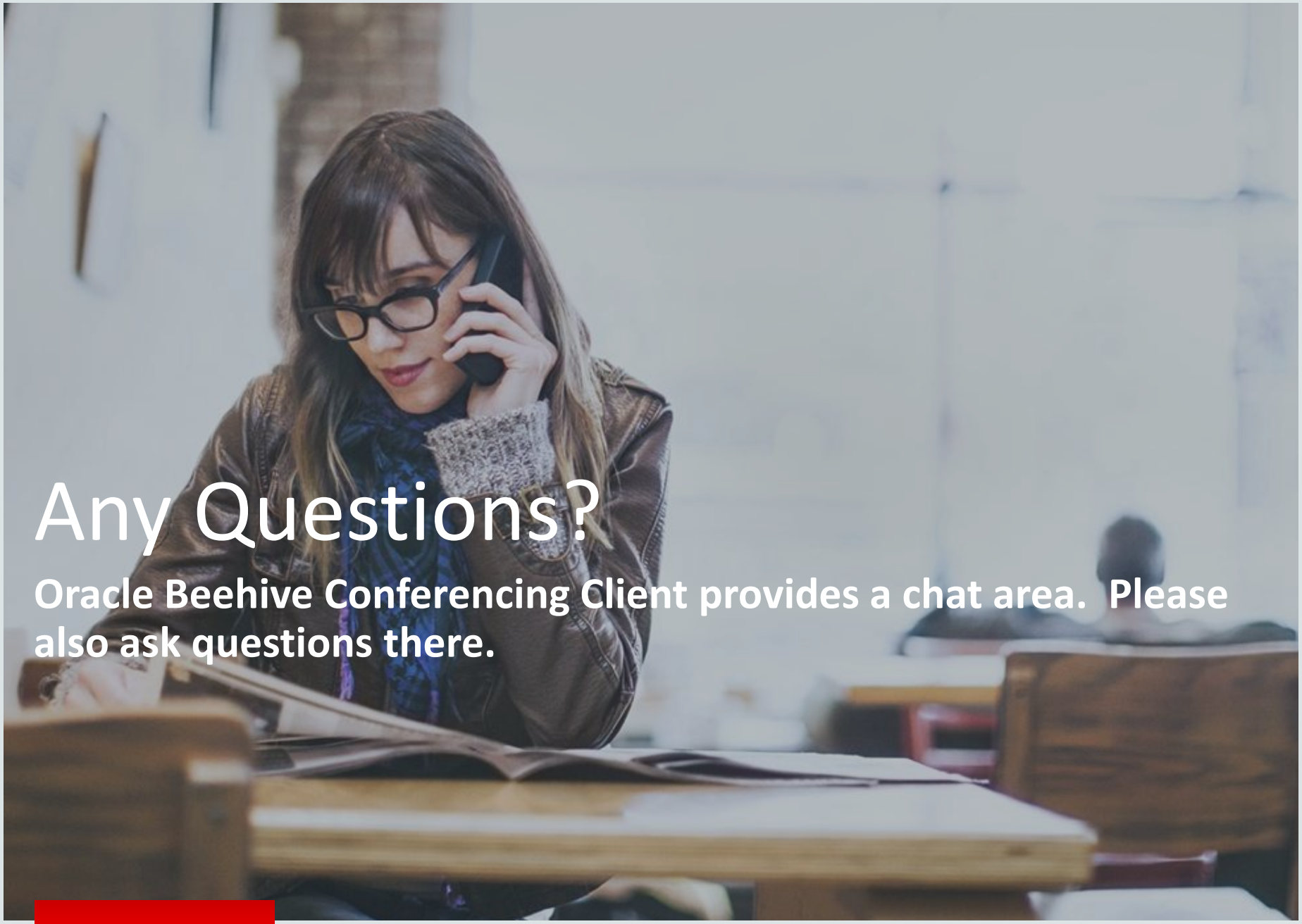
- Used in Rdb regression tests to generate scripts
  - For example, before and after scripts to check correct behavior
- Used by Support and Engineering groups to save time when handling customer databases
  - For example, generate SQL import scripts, or Load/Reload sequences for tests
  - Aside: also have a tool to generate IMPORT script directly from the interchange file (.rbr)
    - rmu/dump/export/option=import

# Continuing to develop

- We are always accepting enhancement requests for RMU Extract
- Recently added POSITION_COLUMN option to control behavior of Extract on tables changed with the BEFORE and AFTER column syntax

# Safe Harbor Statement

The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.



# Hardware and Software
## Engineered to Work Together


ORACLE®