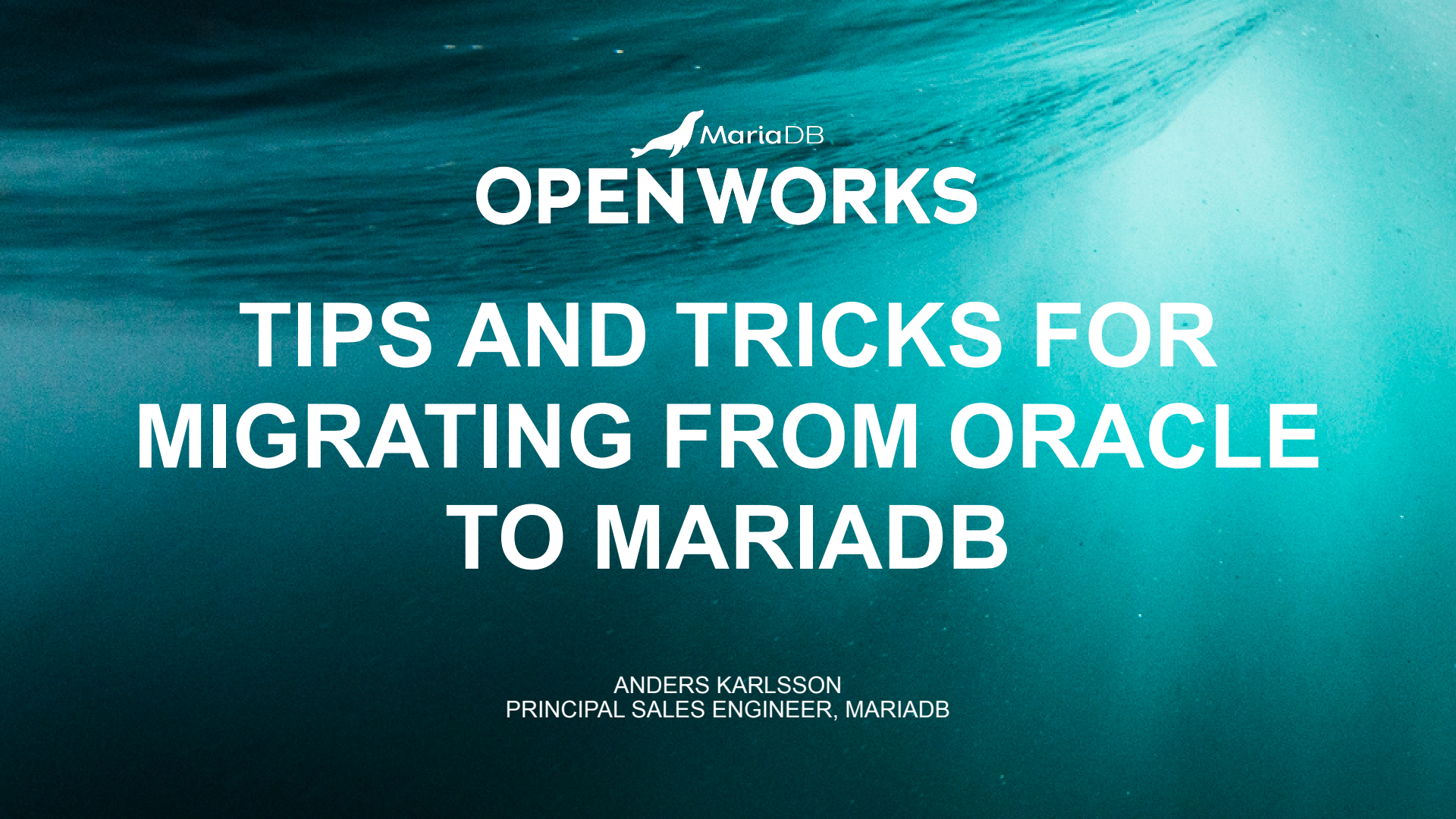

MariaDB
OPEN WORKS
BE UNSTOPPABLE



TIPS AND TRICKS FOR MIGRATING FROM ORACLE TO MARIADB

ANDERS KARLSSON
PRINCIPAL SALES ENGINEER, MARIADB

AGENDA

- Introduction to migration
- Cultural differences
- SQL_MODE=Oracle or not?
- Schema migration
- Procedures, Functions, Triggers and Packages
- Other Oracle proprietary features
- Questions and Answers

MariaDB

INTRODUCTION TO MIGRATIONS

MIGRATION FROM ORACLE

- Migrating an application from Oracle to MariaDB is **always possible**
- The question is **how much work** there is
- There are cases when a **rewrite** could be easier
- And there are cases when a migration can be done in **a day or two**
- There are few cases where **no application code** needs to be visited, although they do exist

MIGRATION ASSESSMENT AND PLANNING

- The recommendation is to **ALWAYS** do an assessment
- Gather as much relevant data as possible
  - Number of tables, indexes, views etc.
  - Number of procedural code, number of lines, use of Oracle proprietary features etc.
  - Data size, preferably per table
  - Application complexity and code, dynamic SQL etc.
- Create a plan and time estimates for the different steps
- Create test and staging for the migrated application

CULTURAL DIFFERENCES ORACLE VS. MARIADB

MariaDB

ORACLE APPLICATION SPECIFICS AND PRACTICES

- Much more **Oracle focused**, even on the application side
- As much as possible of the application is in Oracle procedures, packages etc.
- SQL often more complex, but not always very efficient
- Oracle **proprietary** features often used without an eye for portability
- **Outdated and deprecated** Oracle features are often used
- **Must-have** Oracle features aren't always a must-have when using MariaDB

MARIADB APPLICATION SPECIFICS AND PRACTICES

- MariaDB developers and infrastructure are much more focused on **the actual problem** and less on how it is solved
- In short, MariaDB best practices is **very pragmatic**
- MariaDB assumes to a somewhat larger extent that the **SQL is decently well written and optimized**
- MariaDB best practices includes using infrastructure components
  - Replication
  - MariaDB MaxScale
- MariaDB is less focused on solving all problems with one monolithic component
  - Using a **separate reporting replica**, using MariaDB ColumnStore for analytics etc.

SQL_MODE=ORACLE OR NOT?

MariaDB

SQL_MODE=ORACLE OR NOT?

- SQL_MODE=Oracle makes MariaDB emulate some Oracle behavior
  - SQL/PSM is replaced by SQL/PL
  - DECODE, LENGTH, TRIM and other functions added or different
  - INTERSECT precedence in line with Oracle (not SQL Standard)
  - DATE, RAW, BLOB, CLOB, VARCHAR2 and NUMBER datatype aliases
  - And more
- Do you want to migrate using SQL_MODE=Oracle or replace all of the above?
  - **It depends on**, if there is a lot to migrate, then you might want to do so
- But if not, maybe you want to might as well migrate and ignore SQL_MODE
- And: **Don't migrate bad code, it will not get better by being migrated!**

ORACLE SCHEMA MIGRATION TO MARIADB

MariaDB

ORACLE STRING AND TEMPORAL DATA TYPES

- **VARCHAR2** – Pretty much the same as **VARCHAR** in MariaDB. Note that VARCHAR2 can be specified as BYTE or CHAR (i.e. col1 VARCHAR2(10 BYTE)
- **NVARCHAR2** – National character set VARCHAR2 (16 bit). SQL_MODE=Oracle maps this to VARCHAR with **character set UTF8**
- **DATE** – DATETIME in MariaDB. And Oracle DATA hold both DATE and TIME, in MariaDB a DATE hold date only. **SQL_MODE_Oracle** maps DATE to DATETIME
- **RAW**, **LONG**, **LONG RAW**, **ROWID**, **UROWID** – Oracle specific types that are best migrated column by column. RAW, LONG and LONG ROW shouldn't be used in modern Oracle applications and are unusual in older ones also
- **TIMESTAMP** – Matches MariaDB DATETIME pretty well
- **BLOB** and **CLOB** – Matches the different BLOB and TEXT types

ORACLE NUMERIC DATA TYPES

- NUMBER is the most common numeric data types in Oracle by far
- NUMBER is a **variable length data type**, so independent of how a column is declared it is **the number that is stored that determines the size**
- NUMBER is also a **fixed point** decimal type
- NUMBER is specified with **optional precision and scale**
- If only scale is to be specified, use an **asterisk for precision**
  - **NUMBER(*,0)** can hence store the maximum size integer
- NUMBER may also have negative scale
  - **NUMBER(5,-2)** will store numbers in even 100's up to 9999900

ORACLE NUMERIC DATA TYPES – IN REAL LIFE

- You would expect that a proper data type for integers would be **NUMBER(*,0)**
- This is however not how Oracle application do things though
- For a numeric PRIMARY KEY, say one generated from a SEQUENCE, which generates integers, it is even more common that NUMBER, with maximum precision and scale, is used
- And this is not as bad as you might think, the integer value in question will **occupy the same space and work in exactly the same way**, independent if the column is declared as NUMBER(*,0) or NUMBER
- The only difference is that a column declared as NUMBER will happily store a **decimal value**, but we are never doing that, right?

ORACLE NUMERIC DATATYPES – MIGRATING

- In SQL_MODE=Oracle **NUMBER** is converted to **double** in MariaDB

```
CREATE TABLE emp(empno NUMBER,
  ename VARCHAR2(10));
```


```
CREATE TABLE emp(empno double,
  ename VARCHAR(10));
```

- And this will work also, right? We can store INTEGERs in a double?
- Yes, it will work but it has several issues
- A double needs **8 bytes for storage**, an INTEGER only 4 and a BIGINT 8
- A double in MariaDB is **floating point**, so when values get high, doing math with them gets "interesting"
- Allowing a decimal value where an INTEGER should be opens up for errors
- **The actual data in a NUMBER column should determine it's migrated type!**

MariaDB

MIGRATING ORACLE PACKAGES, PROCEDURES, FUNCTIONS AND TRIGGERS TO MARIADB

MariaDB

ORACLE PL/SQL

- PL/SQL is the programing language used by Oracle for **procedural code**
- **Oracle defines** what gets into PL/SQL and how it emerges
- PL/SQL includes procedural statements, but also a number of **supporting constructs** (more on this later)
- PL/SQL also includes a number of **built-in packages**
- MariaDB supports **SQL/PL** which includes the procedural statement aspect of PL/SQL and some supporting constructs
- MariaDB native procedural logic uses **SQL Standard SQL/PSM**
- In MariaDB SQL/PL and SQL/PSM can be **mixed**
- MariaDB SQL/PL requires SQL_MODE=Oracle

ORACLE PL/SQL VS. SQL/PL

- Control structs works as with SQL/PL as with PL/SQL
  - IF, WHILE, Numeric and Cursor FOR loops
- Cursor attributes: %ROWCOUNT, %ISOPEN, %FOUND and %NOTFOUND
- Table, column and Variable types: %ROWTYPE and %TYPE
- PL/SQL style exceptions
- SQL syntax differences as per SQL_MODE=Oracle

ORACLE DYNAMIC SQL

- **Dynamic SQL** is used quite often in Oracle applications
- In general, dynamic SQL is supported by MariaDB
- But in some cases Dynamic SQL is combined with, say, **refcursors** which are possible to migrate but this often turn into being rather inefficient
- The recommendation is to **limit use of Dynamic SQL** if possible
- More advanced dynamic SQL, such as what is implemented in Oracle in the **DBMS_SQL** package, is not supported by MariaDB
- DBMS_SQL is best migrated by **rewriting** the code segments using this

MariaDB

ORACLE PL/SQL – PIPELINE AND REFCURSOR

- **Pipelines** and **refcursors** in Oracle PL/SQL is a means of moving data from one procedure to another, sort of
- Neither is available out of the box in MariaDB Server though
- Three ways of dealing with these constructs are
  - Rewrite the code to not use them
  - Replace with **JSON**
  - Replace with **temporary tables**

ORACLE OBJECT TYPES AND TYPED TABLES

ORACLE OBJECT TYPES AND TYPED TABLES

- Oracle object types and typed tables implement kind of **object orientation** in Oracle applications
- Oracle object types and typed tables are strictly an **Oracle proprietary feature**
- In the real world, these are not used that much
- In most cases, the use of object types are in particular typed tables is simple and these can be **migrated away**
- In some cases, object types can be **replaced with JSON**
- **Collections / Arrays**
  - Use either JSON or put array members as rows in a separate table

WHEN JSON MIGHT AND MIGHT NOT BE A GOOD IDEA

- Oracle schema

```
CREATE TYPE dept_t AS OBJECT(deptno NUMBER(2,0),
  dname VARCHAR2(14),
  loc VARCHAR2(13))/

CREATE TABLE dept OF dept_t(PRIMARY KEY(deptno));
```

```
CREATE TYPE address_t AS OBJECT(
  street VARCHAR2(100), city VARCHAR2(100))/

CREATE TABLE emp(empno NUMBER(4,0),
  ename VARCHAR2(10),
  emp_addr address_t);
```

- In the first case, you are likely best off to **rewrite** the table schema, ignoring the OBJECT TYPE
- In the second case, **JSON could be useful** in some cases in other cases you might want to rewrite the table schema

CONCLUSION

MIGRATING FROM ORACLE - CONCLUSIONS

- A **migration assessment** is a really important first step
- **Cultural differences** needs to be taken into consideration, just because something can be done in some way doesn't mean it's a good idea to do so
- MariaDB SQL_MODE=Oracle is helpful, but needs to be understood to be used effectively
- Oracle and MariaDB data types are mostly easy to migrate, the key is to **understand the differences**
- Oracle has a lot of **proprietary features**, careful consideration needs to be taken when migration these


MariaDB
OPEN WORKS
THANK YOU


MariaDB
OPEN WORKS
BE UNSTOPPABLE