MariaDB

LOWER TCO WITHOUT LOWERING STANDARDS

Migrate to MariaDB Enterprise from Oracle

WHITEPAPER

TABLE OF CONTENTS

LOWER TCO WITHOUT LOWERING STANDARDS: MIGRATE TO MARIADB ENTERPRISE FROM ORACLE

Built on 20+ years of open source engineering and production use, MariaDB Server is an enterprise open source database for transactional workloads, analytical workloads, or both – at scale. MariaDB Server is the foundation of MariaDB Enterprise – an integrated suite of enterprise-grade products for storing and accessing data.

MariaDB Enterprise is the only open source database with the same enterprise features found in proprietary databases, including Oracle Database compatibility (e.g., PL/SQL compatibility), temporal tables, sharding, point-in-time rollback, and transparent data encryption. And all of this comes at a much lower total cost of ownership (TCO) than Oracle.

MariaDB has displaced MySQL in leading Linux distributions, including Red Hat Enterprise Linux (RHEL) and SUSE Linux Enterprise Server (SLES). MariaDB Enterprise is used on premises, as well as on public and private cloud infrastructure platforms, including OpenStack. It is a trusted, proven, and reliable replacement for proprietary databases.

TOTAL COST OF OWNERSHIP & SCALING BENEFITS

MariaDB Enterprise frees companies from the costs, constraints, and complexity of proprietary databases, enabling them to reinvest in what matters most – rapidly developing innovative, customer-facing applications.

Trusted by organizations like Deutsche Bank, DBS Bank, Nasdaq, Red Hat, ServiceNow, Verizon, and Walgreens – MariaDB Enterprise meets the same core requirements as proprietary databases, but at a fraction of the cost. DBS Bank saved 90% of their total cost for databases by migrating mission-critical applications to MariaDB Enterprise.

Many see Oracle as expensive and inflexible with a high total cost of ownership and rigid licensing policies. Even simple features like replication can require complex licenses for specific features. In contrast, only support subscriptions are needed to run MariaDB Enterprise Server (the default version for customers using MariaDB Enterprise).

With MariaDB Enterprise, deployments are easier, faster, and cheaper and there's a choice of data center topologies – increasing CPU and server count without costly additional features.

MariaDB Enterprise can achieve a consolidation of architecture, can simplify deployments, and can use the same functionality throughout the ecosystem. With free performance scale-out through hardware and no notion of limited additions, it's cost-effective.

Saving Millions with MariaDB

The following costs of each database are calculated, using published list prices, based on a minimal configuration capable of meeting standard enterprise requirements for a database.

After three years running on three on-premises servers, each with two, 16-core processors:

- The total cost of Oracle is 84x higher than MariaDB Enterprise High availability
- The annual cost of Oracle is 33x higher than MariaDB Enterprise
- Organizations can save over $9 million after three years by choosing MariaDB Enterprise
- Organizations can save $1.1 million annually by replacing Oracle

After three years running on three cloud instances, each with 16 cores (32 hyperthreads):

- The total cost of Oracle is 72 to 151x higher than MariaDB Enterprise
- The annual cost of Oracle is 60 to 72x higher than MariaDB Enterprise
- Organizations can save $7.6 to 16.3 million after three years by choosing MariaDB Enterprise
- Organizations can save $2.1 to 2.5 million annually by replacing Oracle

See Appendix A for details.

MARIADB ENTERPRISE BENEFITS

MariaDB Enterprise provides features not found in proprietary database platforms:

MariaDB Enterprise's pluggable, purpose-built storage engines support workloads that previously required a variety of specialized databases. In MariaDB Enterprise, the storage engine can be set on a per-table basis, and transactions can query and join across multiple storage engines.

Organizations can now depend on a single complete database for all their needs, whether on commodity hardware or their cloud of choice. Deployed in minutes for transactional, analytical, or hybrid use cases, MariaDB Enterprise delivers unmatched operational agility without sacrificing key enterprise features, including real ACID compliance and full SQL.

Pluggable storage engines included with MariaDB Enterprise allow you to deploy a single stack with capabilities to handle row-based transactional data, write-heavy workloads, and columnar data storage.

ARIA	**InnoDB**	**MyRocks**	**Spider**	**ColumnStore**
Read-intensive	Mixed read/write	Write intensive	Scalable	Analytical

Smart transactions are built in. MariaDB Enterprise is the only open source relational database to use both row and columnar storage to perform both transactional and analytical processing at scale. Using InnoDB and/or MyRocks for row storage and transactional processing, and ColumnStore for distributed columnar storage and massively parallel analytical processing, MariaDB replicates data from row storage to columnar storage and routes queries to one or the other depending on whether they're transactional or analytical – and it's all transparent to applications.

MariaDB Enterprise incorporates open source technologies, with open and transparent development that ensures customers and the community have access to everything from test cases and security bugs to a shared product experience with users worldwide. While all vendors develop proprietary tools, MariaDB Corporation uses the Business Source License (BSL), guaranteeing they become open source.

MariaDB Enterprise development is open and transparent, leading to collaboration with innovators like Alibaba, Facebook, Google, and Tencent.

MariaDB Enterprise incorporates MariaDB MaxScale, an advanced database proxy, query router, and SQL firewall. MaxScale provides:

- Transparent proxying to conceal complex database architectures from applications
- Scale-out load balancing, enabling multiple servers to handle traffic
- Management of high availability failover in replicated and clustered environments

Flexible topology is built into the architecture. MariaDB Enterprise provides multi-node, virtually synchronous replication with Galera that can work over the wide area network (WAN), in-order parallelized asynchronous replication, semi-synchronous replication which can do failovers with no transaction loss, and plain asynchronous replication with very flexible data center topologies.

MariaDB Enterprise implements a multi-threaded architecture to scale performance with the number of cores/processors. MariaDB Enterprise improves scalability and performance using an advanced database proxy and multiple, purpose-built storage engines.

FEATURE PARITY

MariaDB Enterprise offers a comparable core set of enterprise features to those found in Oracle. In some areas MariaDB Enterprise offers more, such as pluggable storage engines and flexible cluster solutions, and in others Oracle has features, such as Materialized Views, that MariaDB doesn't. Comparing the differences and the TCO, you can decide if a feature is worth the cost.

MariaDB Enterprise and Oracle Database both include:

- ACID compliance, referential integrity, transactions
- Unicode UTF-8
- Transparent data encryption
- Foreign keys
- UNION, INTERSECT, EXCEPT, Joins
- Common Table Expressions (CTE), Window Functions
- Temporal tables (system versioned, application time-period, bitemporal)
- Cursors, triggers, functions, events, procedures, dynamic SQL, sequences
- Set operators, table value constructors
- User-defined aggregate functions, ordered-set aggregate functions
- Partitioning and sharding
- Point-in-time rollback
- Invisible columns, generated or virtual columns

Built-in Migration Compatibility

MariaDB-implemented Oracle Database compatibility simplifies the process of migrating from Oracle Database and reduces migration costs and time. DBAs and developers can continue to apply their Oracle Database knowledge. MariaDB Enterprise supports Oracle PL/SQL, sequences, dynamic SQL (i.e., EXECUTE IMMEDIATE) and packages, and offers direct execution of PL/SQL code with `sql_mode='ORACLE'`.

The PL/SQL compatibility parser – called SQL/PL in MariaDB Server – was added in version 10.3 for easier migrations from Oracle to MariaDB. Oracle SQL dialect can be understood by MariaDB Server 10.3 and above. The `ORACLE` mode understands about 80-90% of PL/SQL procedures and some of the remaining constructs and library calls can be migrated with some manual intervention.

Unlike in the past, there is no need to migrate SQL/PL logic to SQL/PSM or to an application layer when migrating from Oracle to MariaDB.

The SQL/PL `SQL_MODE` is used when syntax isn't compatible with the SQL/PSM standard that MariaDB Server uses normally. Syntactic differences between SQL/PSM and SQL/PL are addressed by this compatibility parser. The compatibility parser is not simply a translator, but rather an equal parser to the native SQL/PSM. There should be no performance differences between running a statement in SQL/PL or SQL/PSM in MariaDB Server.

SQL/PSM-based stored functions can still be used along with and within SQL/PL queries and functions as well as the reverse. This allows the migration team to leave Oracle SQL code unchanged as much as possible by just taking over the existing code. Modifications are done only as necessary.

SQL/PL becomes part of the new MariaDB-based application, and it never actually has to be phased out. This is a dramatic reduction of effort and cost.

Other compatibility features of MariaDB Server 10.3 and above include Oracle compatible packages and stored functions, and data type synonyms such as:

- `VARCHAR2`, a synonym to `VARCHAR`
- `NUMBER`, a synonym to `DECIMAL`
- `DATE` (with time portion), a synonym to `DATETIME`
- `RAW`, a synonym to `VARBINARY`
- `CLOB`, a synonym to `LONGTEXT`
- `BLOB`, a synonym to `LONGBLOB`

There are some Oracle data types that are not supported by MariaDB, such as `BFILE`. This is where the developer or DBA would look at alternative implementations, such as moving to a `VARCHAR` with a file or URL pointer instead of having it as a `BFILE`, or perhaps it could be a `BLOB`. MariaDB hasn't implemented `NVARCHAR2` for text but a text with UTF-8 safe collation accomplishes the same thing.

For an in-depth examination of data type compatibility, see Appendix B.

Infrastructure

MariaDB Enterprise delivers functionality that can be used to replace a range of Oracle components:

Oracle	**MariaDB**
Oracle Real Application Clusters (RAC)	MariaDB Cluster
Oracle Active Data Guard	MariaDB MaxScale – automatic failover
Oracle Application Continuity	MariaDB MaxScale – transaction replay
Oracle Recovery Manager (RMAN)	MariaDB Enterprise Backup
Oracle Flashback Query/Versions Query	Temporal tables (standard SQL)
Oracle Flashback Database/Table	MariaDB Flashback
Oracle Partitioning	Partitioning
Oracle Sharding	Spider
Oracle Database In-Memory	MariaDB ColumnStore
Oracle Advanced Compression	Table, column, and log compression
Oracle Advanced Security – encryption	Transparent data encryption (tables, logs, etc.)
Oracle Advanced Security – data masking	MariaDB MaxScale – dynamic data masking
Oracle Database Firewall	MariaDB MaxScale – database firewall
Oracle Location/Spatial and Graph	Geospatial data types and functions
Oracle PL/SQL	MariaDB SQL/PL (Oracle-compatible)

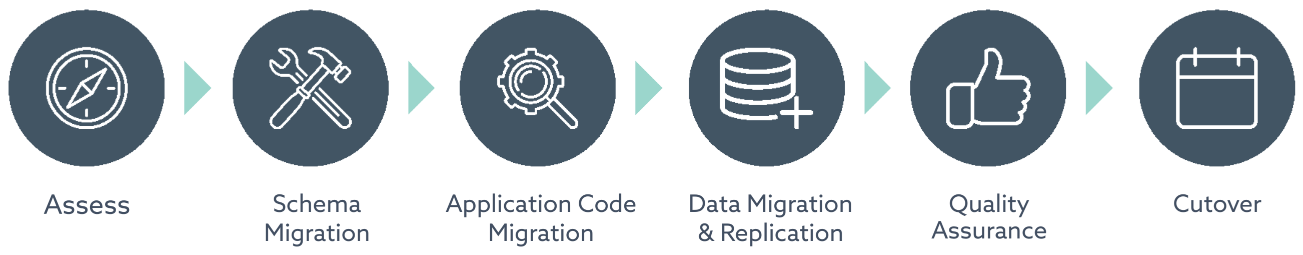
MIGRATION METHODOLOGY

MariaDB Corporation provides the expertise needed to upgrade from a legacy database management system (DBMS) to a MariaDB DBMS. With a full understanding of MariaDB Enterprise fundamentals, MariaDB PL/SQL compatibility, and other advanced features, MariaDB experts define an upgrade process and supporting migration team roles that ensure the success of your database migration.

Successful migration depends on company-wide acceptance and participation. All stakeholders must be included throughout the upgrade process. Stakeholders include legacy and MariaDB database administrators (DBAs), project managers (PMs), systems administrators, application developers, and product owners. Quality Assurance (QA) and Change and Incident Management teams can be invaluable in facilitating database migration.

Applications that use well-documented database interfaces, like Oracle with Java, C++, or PHP, and for which source code is available, are excellent candidates for migration. Vendor-neutral applications with clear, documented interfaces that use database drivers are also excellent candidates for migration. Other candidates are applications that can run on the new MariaDB Enterprise without significantly changing application logic.

The process to upgrade to MariaDB Enterprise involves six steps:



Assessment

The migration process begins with an assessment. This is where the go or no-go decision is made. The assessment gives stakeholders an understanding of what will be required to upgrade to MariaDB and serves as a basis for a migration plan.

The Assess phase considers three areas:

- **Environment:** MariaDB Enterprise Architects and Professional Services start the assessment with questions about the environment:
    - What is the application doing?
    - What problem is the application trying to solve?
    - How does the database help that application realize the use case?

- **Inventory:** The Assessment phase produces a work inventory by examining the database schema, procedures, data, and workload. This can usually be accomplished with an automated process that includes collecting system and database statistics during normal operation. Assessment is augmented with database and operational information and a MariaDB questionnaire on architecture, teams, and business processes.

- **Challenges:** From the collected data, potential issues are identified. As part of the decision to upgrade, a plan is developed that includes an incident response plan. MariaDB experts look at issues flagged in the assessment and perform mitigation planning.

Schema Migration

Schema migration follows the Assessment phase and the decision to move forward with the migration. The scope of schema migration includes only non-code Data Definition Language (DDL) objects such as:

- Tables
- Constraints
- Indexes
- Views

Automated processing can aid schema object migration, excluding stored SQL code to SQL/PSM or SQL/PL migration. Some helpful tools are SymmetricDS, Blitzz, SQLines, Amazon Web Services Schema Conversion Tool (AWS SCT), and DBeaver.

Application Code Migration

Schema migration is followed by application code migration, which involves porting and testing SQL and application code. This phase requires the migration project team to work closely with the development team and other project owners.

Application code and SQL migration should be done in parallel with SQL code migration, updating application code to work with MariaDB Enterprise. This can include replacing legacy database connectors with MariaDB connectors and adapting Object Relational Model (ORM) software settings for MariaDB Enterprise.

- The SQL code migration includes examining triggers, functions, and stored procedures. SQL code is transformed to SQL/PSM or MariaDB-compatible SQL/PL.
- Use scripting to import as much PL/SQL as possible without changes. Oracle tools such as DataPump can be used to export metadata only, excluding elements like system schemas, to get all PL/SQL in cleartext. PL/SQL can be used natively with MariaDB's SQL mode compatibility parsers, like `sql_mode=Oracle`.
- Automated processes can be used for the bulk of the work at this stage but some manual effort is usually necessary.

Code Testing and Freeze

It is important to do in/out unit testing as the code is brought into MariaDB. As developers make application code migration or changes, you can do basic side-by-side, in/out testing of the code. If 1 is input into Oracle functionA and 2 is returned, the same 2 must be returned when 1 is input into MariaDB functionA. Real data are not used in unit tests.

Any re-engineered application code must also be tested against the MariaDB schema. For example, make sure Hibernate is building good queries, including an up-to-date version, and that ORM queries work well with modern MariaDB.

After successful DDL and SQL code migration, developers and DBAs should avoid changing the DDL, especially the SQL code. This code freeze should last until the Switchover phase is completed to prevent introducing changes during the migration. If the DDL is not frozen, schema testing and SQL code migration must be repeated during the QA and Switchover phases.

Data Migration and Replication

The Data Migration and Replication phase follows importing the schema and SQL code.

Migration

Data migration loads the source database into the MariaDB database with some validation. Usually, a tool is used to get data from the source database to efficiently bulk load it into MariaDB. Tools include SymmetricDS, Blitzz, AWS Data Migration Services (DMS), MySQL replication, GoldenGate, and others.

Certain types of issues specific to data migration may occur at this stage, such as differences in character sets and collations, or different handling of padding in character columns. For example, `NVARCHAR` is a UTF-8 safe data type in Oracle. In MariaDB, `NVARCHAR` can either automatically specify UTF-8 collations using `sql_mode=”Oracle”` or the collation and character set can be adjusted to one that suits the data by defining `NVARCHAR` with `VARCHAR`. Navigating these kinds of issues are where experience is especially helpful.

Replication

Data replication follows data migration and is used for consistency and integrity to copy changes from the legacy database to MariaDB in real time over a defined time interval. Replication is either one-way to MariaDB or two-way between MariaDB and the legacy database. Two-way replication has the advantage of providing easier rollback.

MariaDB replication pulls binary logs from primary nodes, using the IO thread and applying the data using the SQL thread. MariaDB migrations can use binary log replication directly without needing extract-load-transform (ETL) processes.

Some tools and migration methods can require a DDL freeze during migration until the Cutover phase is completed. When a replication tool incorrectly migrates the DDL because of translations and transformations done by the migration team, DML-only replication is preferred although this can burden applications under active development during the Migration phase.

Replication should be monitored with alerting and ongoing QA because replication can unexpectedly change data, the data structure, or referential integrity. Monitor and alert on consistency issues or when replication fails. Continuously test all applications, APIs, and jobs.

Quality Assurance

After the data are migrated, QA becomes a vital step. Database testing requires query and procedure writing and table-checking expertise. Testing should be performed on both the application and the database.

Because of the complexity, testing should be done by legacy DBAs, MariaDB DBAs, and those with field experience. Testers should make sure that values have been added correctly for business rules. Complex use cases such as banking, finance, and health insurance may require more extensive database testing.

Use automated scripts to do checksums, counts, and smoke testing. In/out unit testing should be repeated for stored code using real data.

Several kinds of tests should be performed as part of the QA phase:

- **Data validity tests:** Are the data the same in the legacy and MariaDB databases? This requires expert SQL knowledge.
- **Data integrity tests:** Is the referential integrity consistent and are constraints effectively maintaining database integrity?
- **Performance tests:** Is performance the same in the legacy and MariaDB databases? This requires understanding the differences between MariaDB database design and legacy database design and applying best practices to ensure MariaDB provides expected or better-than-expected performance.
- **Query tests:** Do application queries return the same results in the legacy and MariaDB databases? If a query changed in the Migration phase, determine if it is semantically equivalent to the original query.
- **Stored code tests:** Include procedures, triggers, and functions using real data. Test the functionality of every application action, including deletion, addition, and save operations. Confirm that inserted records are added with the expected value, and that deleted records are removed.
- **Data and data structure tests:** Use manual and automated, script-driven processes. Compare records with checksums, counts, smoke tests, randomized reads, writes, and stored code calls. Run client applications, APIs, and batch jobs. Run in/out testing for stored code using real data.

Cutover

Cutover is when the production migration occurs. The Cutover phase involves four tasks:

- Production migration (optional)
- Setting up rollbacks
- Switchover
- Decommissioning the legacy database

Production Migration

Production migration is important when the QA and previous tests might have affected the data, schema, or SQL code. In this phase, previous non-destructive tests are repeated. If needed, the team again migrates all schema, SQL code, and data, using a repeatable process and the same tools.

This task is optional.

Rollback

If issues are found with the MariaDB application in production, the application can set up rollback and switch to the legacy database for a specified time interval until the problems are resolved. Live changes are replicated from MariaDB to the legacy Oracle database in real time. Reverse replication can guard against production outages or unforeseen migration issues.

The best way to set up rollback is to do the migration replication steps in reverse. For example, SymmetricDS Pro could be set up to replicate from MariaDB to an Oracle replica. Or a "switch" could be flipped during the production switchover to replicate in reverse from MariaDB to the legacy database.

Setup can be done in advance as circular replication or using a fresh legacy database installation as its replica.

Reverse replication should be maintained for a specified time interval following the switchover.

Switchover

Switchover, when production users and applications start using MariaDB Enterprise exclusively, is the most business-critical step.

Ideally, switchover should be done within a single maintenance window or as an atomic operation. All applications, APIs, load balancers, connectors, DSNs, and other database interfaces should be connected to MariaDB Enterprise during the switchover window.

Migration, incident, and change management teams – including contracted MariaDB Professional Services resources – should be on call or available in person.

Decommissioning

The legacy database is ready for decommissioning when there is a high level of confidence in MariaDB as the production database. This usually occurs at least a week after Cutover. It is important that stakeholders and users have confidence in the MariaDB migration before decommissioning the legacy database.

Decommission legacy databases according to internal decommissioning processes, which should include long-term durable backup considerations and securely erasing decommissioned disk drives.

NEXT STEPS

Now that you know how MariaDB stacks up against Oracle, and the basic migration process, it's time to take the next step in escaping your proprietary database.

Contact us! Whether you'd like help re-training your Oracle DBAs or want to learn about database migration services, we're here to ensure your success.

And to learn more about how MariaDB Enterprise can benefit your organization, check out the **enterprise documentation.**

APPENDIX A: TCO

Total Cost of Ownership (On Premises)

In this analysis, the database software and support costs are calculated for a three-node cluster running on commodity hardware. MariaDB Enterprise, with the clustering option, includes all of the features necessary to meet enterprise requirements – including MariaDB MaxScale, a database proxy for improved high availability, security, and scalability. However, Oracle requires licenses for multiple features in order to meet enterprise requirements.

	MariaDB	Oracle
Database	$9,600	$47,500
Clustering	$2,400	$23,000
Partitioning	Included	$11,500
Compression	Included	$11,500
Security	Included	$15,000
Firewall	Included	$6,000
TOTAL	**$12,000 (Server)**	**$114,500 (Core)**

MariaDB		Oracle	
Database Servers	**3**	**Database Server**	**3**
		Processors/Server	2
		Cores/Processor	16
		Total Cores	96
		Core Factor	0.5
		Adjusted Cores	48
TOTAL	**$36,000**	**TOTAL**	**$5,496,000**

	Year 1		Year 2		Year 3	
	MariaDB	Oracle	MariaDB	Oracle	MariaDB	Oracle
Subscription	$36,000	-	$36,000	-	$36,000	-
License	-	$5,496,000	-	$0	-	$0
Maintenance	-	$1,209,120	-	$1,209,120	-	$1,209,120
TOTAL	**$36,000**	**$6,705,120**	**$72,000**	**$7,914,240**	**$108,000**	**$9,123,360**

Total Cost of Ownership (Cloud)

The AWS configuration for Oracle Database includes one active server and two standby servers. Oracle RAC is not supported. The Oracle Cloud configuration includes two active servers and one standby server (no cost). Oracle RAC is supported, but it is limited to two servers. In both configurations, Oracle Active Data Guard enables reads on the standby servers.

	MariaDB (AWS)	Oracle (Oracle)	Oracle (AWS)
Database/Proxy	$9,600	Included	$47,500
Clustering	$2,400	Limited	N/A
Replication	Included	Included	$11,500
Partitioning	Included	Included	$11,500
Compression	Included	Included	$11,500
Security	Included	Included	$15,000
Firewall	Included	Included	$6,000
TOTAL	**$12,000 (Server)**	**$27,000 (Core)**	**$103,000 (Core)**

MariaDB(AWS)		Oracle (Oracle)		Oracle (AWS)	
Database Servers	**3 Actives**	**Database Servers**	**2 Active 1 Standby**	**Database Server**	**1 Active 2 Standby**
		Cores/Server	32	vCPU/Server	64
		Total Cores	96	Total vCPU	192
				HT Adjustment	0.5
				Total Cores	96
TOTAL	**$36,000**	**TOTAL**	**$2,592,000**	**TOTAL**	**$9,888,000**

	MariaDB (AWS)	Oracle (Oracle)	Oracle (AWS)
Year 1	$36,000	$2,592,000	$12,063,360
Year 2	$36,000	$2,592,000	$2,175,360
Year 3	$36,000	$2,592,000	$2,175,360
TOTAL	**$108,000**	**$7,776,000**	**$16,414,080**

Note:

Oracle updated its cloud license. It counts two AWS vCPUs as one core, but removed the core factor – it is no longer applied when calculating the number of processor licenses required to run on AWS. As a result, it now costs twice as much to run on AWS than it does on premises.

	vCPU	Cores	Core Factor	License	Total
On premise	-	96	0.5	$47,500	$2,280,000
AWS	192	96 (192/2)	-	$47,500	$4,560,000

APPENDIX B: DATA TYPES

Working Around Unsupported Data Types

While the data types found in most applications are available, some specific types, such as `BFILE` are not. In these cases, the DBA or developer can consider alternative implementations, such as moving to a `VARCHAR` with a file or URL pointer, or storing the data in a `BLOB` within the database instead of using a `BFILE`.

Floating and Fixed Point Data Types

MariaDB has many one-to-one mappings with legacy database data types. A consideration coming from Oracle is the difference between `NUMBER` and `INT` or `NUMBER` and numeric float `DOUBLE`. There are some types that are accurate and there are some that are float and may be imprecise to a certain level of decimals, depending on either the legacy database or MariaDB's implementation.

FLOATING & FIXED POINT DATA TYPES

ORACLE	MARIADB
DECIMAL(p,s), DEC(p,s)	DECIMAL(p,s), DEC(p,s)
DOUBLE PRECISION	DOUBLE PRECISION
FLOAT(p)	DOUBLE
INTEGER, INT	INT
NUMBER(p,0), NUMBER(p), 1 <= p < 3	TINYINT
NUMBER(p,0), NUMBER(p), 3 <= p < 5	SMALLINT
NUMBER(p,0), NUMBER(p), 5 <= p < 9	INT
NUMBER(p,0), NUMBER(p), 9 <= p < 19	BIGINT
NUMBER(p,0), NUMBER(p), 19 <= p <= 38	DECIMAL(p)
NUMBER(p,s), s > 0	DECIMAL(p,s)
NUMBER, NUMBER(*)	DOUBLE
NUMERIC(p,s)	NUMERIC(p,s)
REAL	DOUBLE
SMALLINT	DECIMAL(38)

String Data Types

In the table below, there is a mapping of string types. `CHAR(256)` or `VARCHAR(256)` are usually stored on page in engines such as InnoDB. When there are larger text columns, they may also be stored off page and this has some indexing, storage, and performance implications that should be considered carefully.

STRING DATA TYPES

ORACLE	MARIADB
CHAR(n), CHARACTER(n), n < 256	CHAR(n), CHARACTER(n)
CHAR(n), CHARACTER(n), n > 255	VARCHAR(n), TEXT, MEDIUMTEXT, LONGTEXT
CLOB	LONGTEXT
INTERVAL YEAR(p) TO MONTH	VARCHAR(30)
INTERVAL DAY(p) TO SECOND(s)	VARCHAR(30)
LONG	LONGTEXT
NCHAR(n), n < 256	NCHAR(n)
NCHAR(n), n > 255	NVARCHAR(n)
NCHAR VARYING(n)	NCHAR VARYING(n)
NCLOB	NVARCHAR(max), LONGTEXT
NVARCHAR2(n)	NVARCHAR(n)
ROWID	CHAR(18)
UROWID(n)	VARCHAR(n)
VARCHAR(n),VARCHAR2(n)	VARCHAR(n)
XMLTYPE	LONGTEXT

Date and Time Data Types

With DATE and TIME data types, an issue to be aware of is TIMESTAMP with timezone or DATE with timezone on the source database. From MariaDB Server 10.3 and above, DATE can store time as well – this is specifically for Oracle compatibility.

TEMPORAL DATA TYPES

ORACLE	MARIADB
DATE	DATETIME, DATE (10.3)
TIMESTAMP (p)	DATETIME (p)

Binary Data Types

There are also BINARY data types and the 256-byte rule applies here, just as with the string data types. A few features may need to be implemented slightly differently. Looking at something like a compressed BLOB, the table could now be compressed with InnoDB row or page compression.

BINARY DATA TYPES

ORACLE	MARIADB
BFILE	VARCHAR(255)
CLOB	LONGTEXT
LONG RAW	LONGBLOB
NCHAR(n), n < 256	NCHAR(n)
NCHAR(n), n > 255	NVARCHAR(n)
NCHAR VARYING(n)	NCHAR VARYING(n)
NCLOB	NVARCHAR(max), LONGTEXT
NUMERIC(p,s)	NUMERIC(p,s)
NVARCHAR2(n)	NVARCHAR(n)
RAW(n),n < 256	BINARY(n)
RAW(n), n > 255	VARBINARY(n)
UROWID(n)	VARCHAR(n)
VARCHAR(n),VARCHAR2(n)	VARCHAR(n)
XMLTYPE	LONGTEXT