SQL_MODE=ORACLE From MariaDB 10.3

In MariaDB 10.3 and later, setting the sql_mode system variable to `Oracle` allows the server to understand a subset of Oracle's PL/SQL language. For example:

```
SET SQL_MODE='ORACLE';
```

All traditional MariaDB SQL/PSM syntax should work as before, as long as it does not conflict with Oracle's PL/SQL syntax. All MariaDB functions should be supported in both and Oracle modes.

Prior to MariaDB 10.3, MariaDB does not support Oracle's PL/SQL language, and `SET SQL_MODE=ORACLE` is only an alias for the following sql_mode in those versions:

```
SET SQL_MODE='PIPES_AS_CONCAT, ANSI_QUOTES, IGNORE_SPACE, NO_KEY_OPTIONS,
NO_TABLE_OPTIONS, NO_FIELD_OPTIONS, NO_AUTO_CREATE_USER';
```

In MariaDB 10.3 and later, `SET SQL_MODE=ORACLE` is same as:

```
SET SQL_MODE='PIPES_AS_CONCAT,ANSI_QUOTES,IGNORE_SPACE,ORACLE,NO_KEY_OPTIONS,
NO_TABLE_OPTIONS,NO_FIELD_OPTIONS,NO_AUTO_CREATE_USER,SIMULTANEOUS_ASSIGNMENT';
```

Supported Syntax in Oracle Mode

Stored Procedures and Stored Functions

Oracle mode makes the following changes to Stored Procedures and Stored Functions:

Oracle syntax	Description
CREATE PROCEDURE p1 `(param OUT INT)`	ANSI uses `(OUT param INT)`
CREATE PROCEDURE p1 `(a IN OUT INT)`	ANSI uses `(INOUT param INT)`
`AS` before function body	CREATE FUNCTION f1 RETURN NUMBER `AS` BEGIN...
`IS` before function body	CREATE FUNCTION f1 RETURN NUMBER `IS` BEGIN...
If function has no parameters then parentheses must be omitted	Example: CREATE PROCEDURE p1 AS BEGIN NULL; END;
CREATE PROCEDURE p1 AS BEGIN END `p1` ;	Optional routine name after END keyword. MDEV-12089
CREATE FUNCTION f1(a `VARCHAR`)	VARCHAR can be used without length for routine parameters and RETURN clause. The length is inherited from the argument at ca MDEV-10596
CREATE AGGREGATE FUNCTION f1( )	Creates an aggregate function, which performs the function against a set of rows and returns one aggregate result.
No `CALL` needed in Stored Procedures	In Oracle mode one can call other stored procedures with name only. MDEV-12107
`RETURN` . Can also be used in stored procedures	ANSI uses `RETURNS` . MariaDB mode only supports RETURNS in stored functions

Cursors

Oracle mode makes the following changes to Cursors:

Oracle syntax	Description
CREATE PROCEDURE p1 AS CURSOR cur IS (SELECT a, b FROM t1); BEGIN FOR rec IN cur ...	Explicit cursor with FOR loop. MDEV-10581
CREATE PROCEDURE p1 AS `rec IN (SELECT a, b FROM t1)`	Implicit cursor with FOR loop. MDEV-12098
`CURSOR` c(prm_a VARCHAR2, prm_b VARCHAR2) ... `OPEN c(1,2)`	Cursor with parameters. MDEV-10597
`CURSOR` c(prm_a VARCHAR2, prm_b VARCHAR2) ... FOR `rec in c(1,2)`	Cursor with parameters and FOR loop. MDEV-12314
s %ISOPEN, %ROWCOUNT, %FOUND, %NOTFOUND	Explicit cursor attributes. MDEV-10582

LOOP

Oracle mode makes the following changes to LOOP:

Oracle syntax	Description
`FOR i IN 1..10 LOOP ... END LOOP`	Numeric FOR loop. MDEV-10580
`GOTO`	GOTO statement. MDEV-10697
`<<label>>` used with GOTO	ANSI uses `label:` . MDEV-10697
To leave loop block: `EXIT [ label ] [ WHEN bool_expr ]`	ANSI syntax is `IF bool_expr THEN LEAVE label`
[`<<label>>`] WHILE boolean_expression LOOP statement... END LOOP [label] ;	Oracle style `WHILE` loop
`CONTINUE` [label] [WHEN boolean_expression]	`CONTINUE` is only valid inside a loop

Variables

Oracle syntax	Description
`var:= 10` ; Can also be used with MariaDB systemvariables	MariaDB uses `SET var= 10` ;
`var INT := 10`	Default variable value
`var1 table_name.column_name%TYPE`	Take data type from a table column. MDEV-10577
`var2 var1%TYPE`	Take data type from another variable
`rec1 table_name%ROWTYPE`	Take ROW structure from a table. MDEV-12133
`rec2 rec1%ROWTYPE`	Take ROW structure from ROW variable
CURSOR c1 IS SELECT a,b FROM t1; rec1 c1%ROWTYPE;	Take ROW structure from a cursor. MDEV-12011
Variables can be declared after cursor declarations	In MariaDB mode, variables must be declared before cursors. MDEV-10598
Triggers uses `:NEW` and `:OLD`	ANSI uses `NEW` and `OLD` . MDEV-10579
`SQLCODE`	Returns the number code of the most recent exception. Can only be used in Stored Procedures. MDEV-10578
`SQLERRM`	Returns the error message associdated to it's error number argument or `SQLCODE` if no argument is given. Can only be used in Stored Procedures. MDEV-10578
`SQL%ROWCOUNT`	Almost same as ROW_COUNT(). MDEV-10583

Exceptions

Oracle syntax	Description
BEGIN ... EXCEPTION WHEN OTHERS THEN BEGIN .. END; END;	Exception handlers are declared at the end of a block
TOO_MANY_ROWS, NO_DATA_FOUND, DUP_VAL_ON_INDEX	Predefined exceptions. MDEV-10839
RAISE TOO_MANY_ROWS ; .... EXCEPTION WHEN TOO_MANY_ROWS THEN ...	Exception can be used with RAISE and EXCEPTION...WHEN. MDEV-10840
CREATE OR REPLACE FUNCTION f1 (a INT) RETURN INT AS `e1 EXCEPTION` ...	User defined exceptions. MDEV-10587

BEGIN Blocks

Oracle syntax	Description
`BEGIN` to start a block	MariaDB uses `BEGIN NOT ATOMIC` for anyonymous blocks. MDEV-10655
`DECLARE` is used before `BEGIN`	DECLARE a INT; b VARCHAR(10); BEGIN v:= 10; END;
WHEN DUP_VAL_ON_INDEX THEN NULL ; NULL; WHEN OTHERS THEN NULL	Do not require BEGIN..END in multi-statement exception handlers in THEN clause. MDEV-12088

Simple Syntax Compatibility

Oracle syntax	Description
`ELSIF`	ANSI uses `ELSEIF`
SELECT `UNIQUE`	Same as `SELECT DISTINCT` . MDEV-12086
TRUNCATE TABLE t1 [`DROP STORAGE`] or [`REUSE STORAGE`]	DROP STORAGE and REUSE STORAGE are allowed as optional keywords for TRUNCATE TABLE. MDEV-10588

Functions

Oracle syntax	Description
CAST(expr `as VARCHAR(N)`)	Cast expression to a `VARCHAR(N)` . MDEV-11275
`LENGTH()` is same as `CHAR_LENGTH()`	MariaDB translates `LENGTH()` to `OCTET_LENGTH()` . In all modes on can use `LENGTHB()` as a synonym to `OCTET_LENGTH()`
`CHR(num)`	Returns a VARCHAR(1) with character set and collation according to @@character_set_database and @@collation_database
`substr('abc',0 ,3)` same as `substr('abc', 1 ,3)`	Position 0 for `substr()` is same as position 1
TRIM, LTRIM, RTRIM, LPAD and RPAD	Returns NULL instead of an empty string if returning an empty result. These functions can also be accessed outside of ORACLE mode by suffixing _ onto the end of the function name, such as TRIM_ORACLE.

Prepared Statements

Oracle mode makes the following changes to Prepared Statements:

Oracle syntax	Description
PREPARE stmt FROM 'SELECT `:1 , :2` '	ANSI uses `?` . MDEV-10801
EXECUTE IMMEDIATE 'INSERT INTO t1 SELECT (:x,:y) FROM DUAL' USING 10,20	Dynamic placeholders. MDEV-10801

Synonyms for Basic SQL Types

Oracle type	MariaDB synonym
VARCHAR2	VARCHAR

NUMBER	DECIMAL
DATE (with time portion)	MariaDB DATETIME
RAW	VARBINARY
CLOB	LONGTEXT
BLOB	LONGBLOB

This was implemented as part of MDEV-10343.

Packages

The following syntax has been supported since MariaDB 10.3.5:

- CREATE PACKAGE
- CREATE PACKAGE BODY
- DROP PACKAGE
- DROP PACKAGE BODY
- SHOW CREATE PACKAGE
- SHOW CREATE PACKAGE BODY

NULL Handling

Oracle mode makes the following changes to NULL handling:

NULL As a Statement

`NULL` can be used as a statement:

```
IF a=10 THEN NULL; ELSE NULL; END IF
```

Translating Empty String Literals to NULL

In Oracle, empty string ('') and NULL are the same thing,

By using `sql_mode=EMPTY_STRING_IS_NULL` you can get a similar experience in MariaDB:

```
SET sql_mode=EMPTY_STRING_IS_NULL;
SELECT '' IS NULL; -- returns TRUE
INSERT INTO t1 VALUES (''); -- inserts NULL
```

Concat Operator Ignores NULL

CONCAT() and || ignore NULL in Oracle mode. Can also be accessed outside of ORACLE mode by using CONCAT_OPERATOR_ORACLE. MDEV-11880 and MDEV-12143.

Reserved Words

There are a number of extra reserved words in Oracle mode.

See Also

- Using SEQUENCEs
- SQL_MODE=MSSQL
- SQL_MODE EMPTY_STRING_IS_NULL

← Function Differences between MariaDB and MySQL ↑ Compatibility & Differences ↑ SQL_MODE=MSSQL →

Comments

Call package method from other schema *2 months, 3 weeks*

I try to create package in database BV and call it from database XX

```
SET sql_mode=ORACLE;

DELIMITER $$

CREATE OR REPLACE PACKAGE BV.PA_TEST AS
    PROCEDURE SET_CONTEXT( context_id INT );
END; $$

CREATE OR REPLACE PACKAGE BODY BV.PA_TEST AS
    PROCEDURE SET_CONTEXT( context_id INT ) AS
    BEGIN
        SET @context_id = context_id;
    END;

BEGIN
END;
$$

-- USE BV;
$$

BEGIN BV.PA_TEST.SET_CONTEXT(111); END;
 $$

DELIMITER ;
```

this is work fine:

```
USE BV;
$$

BEGIN PA_TEST.SET_CONTEXT(111); END;
$$
```

this will drop error

```
BEGIN BV.PA_TEST.SET_CONTEXT(111); END;
$$
```

SQL Error [1064] [42000]: (conn=14) You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use ne '.SET_CONTEXT(111); END' at line 1 (conn=14) You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right synta use near '.SET_CONTEXT(111); END' at line 1 Query is: BEGIN BV.PA_TEST.SET_CONTEXT(111); END java thread: DBeaver: Read data [BEGIN BV.PA_TEST.SET_CONTEXT(111); END]