ORACLE®

ORACLE OPEN WORLD

MySQL 8.0: What Is New in Optimizer and Executor?

Norvald H. Ryeng
Software Development Senior Manager
MySQL Optimizer Team
October, 2018

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

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.

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

Program Agenda

- Common table expressions
- Window functions
- UTF-8 support
- GIS
- SKIP LOCKED, NOWAIT
- JSON functions
- Index extensions
- Cost model
- Hints

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

Program Agenda

- Common table expressions
- Window functions
- UTF-8 support
- GIS
- SKIP LOCKED, NOWAIT
- JSON functions
- Index extensions
- Cost model
- Hints

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

CTEs As Alternative to Derived Tables

- A derived table is a subquery in the FROM clause:

  SELECT … FROM (*subquery*) AS *derived, t1* …

- A CTE is just like a derived table, but the declaration comes before the query block:

  WITH *derived* AS (*subquery*)
  SELECT … FROM *derived, t1* …

- May precede SELECT, UPDATE, DELETE and be used in subqueries:

  WITH *derived* AS (*subquery*)
  DELETE FROM *t1* WHERE *t1.a* IN (SELECT *b* FROM *derived*);

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

CTEs vs. Derived Tables

- Better readability
- Can be referenced multiple times
- Can refer to other CTEs
- Improved performance

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

CTEs Provide Better Readability

Derived table:

SELECT …
FROM *t1* LEFT JOIN ((SELECT … FROM …) AS *dt* JOIN *t2* ON …) ON …

CTE:

WITH *dt* AS (SELECT ... FROM ...)
SELECT ...
FROM *t1* LEFT JOIN (*dt* JOIN *t2* ON ...) ON …

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

CTEs Can Be Referenced Multiple Times

- Derived tables can't be referenced twice

```
SELECT …
FROM (SELECT a, b, SUM(c) AS s FROM t1 GROUP BY a, b) AS d1
JOIN (SELECT a, b, SUM(c) AS s FROM t1 GROUP BY a, b) AS d2 ON d1.b = d2.a;
```

- CTEs can

```
WITH d AS (SELECT a, b, SUM(c) AS s FROM t1 GROUP BY a, b)
SELECT … FROM d AS d1 JOIN d AS d2 ON d1.b = d2.a;
```

- Better performance with materialization
  - Multiple CTE references are only materialized once
  - Derived tables and views will be materialized once per reference

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

CTEs Can Refer to Other CTEs

- Derived tables can't refer to other derived tables

```
SELECT …
FROM (SELECT … FROM …) AS d1, (SELECT … FROM d1 …) AS d2 …
ERROR: 1146 (42S02): Table ‘db.d1’ doesn’t exist
```

- CTEs can refer other CTEs

```
WITH d1 AS (SELECT … FROM …),
     d2 AS (SELECT … FROM d1 …)
SELECT …
FROM d1, d2 …
```

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

Recursive CTEs

```
WITH RECURSIVE cte AS
    (SELECT … FROM table_name                     /* "seed" SELECT */
    UNION [DISTINCT|ALL]
    SELECT … FROM cte, table_name)                /* "recursive" SELECT */
SELECT … FROM cte;
```

- A recursive CTE refers to itself in a subquery
- The "seed" SELECT is executed once to create the initial data subset
- The "recursive" CTE is executed repeatedly
  - Stops when iteration doesn't generate any new rows
  - Set cte_max_recursion_depth to limit recursion
- Useful to traverse hierarchies (parent/child, part/subpart)

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

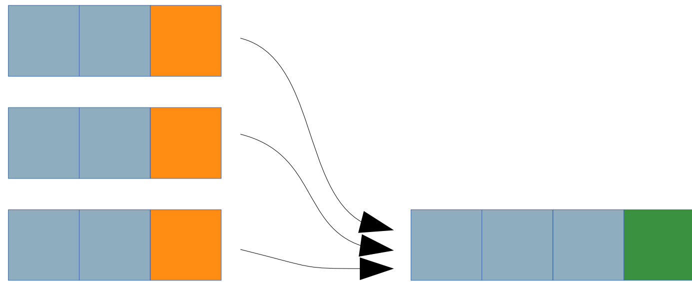
Program Agenda

- Common table expressions
- Window functions
- UTF-8 support
- GIS
- SKIP LOCKED, NOWAIT
- JSON functions
- Index extensions
- Cost model
- Hints

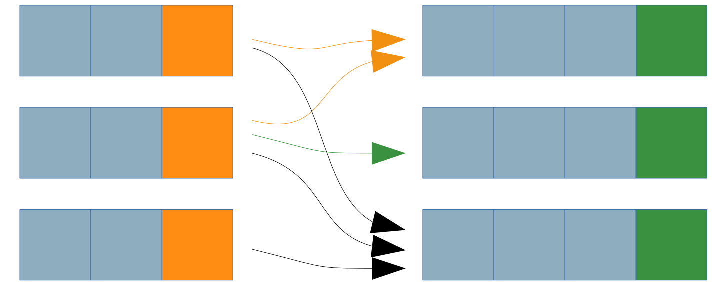
Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

Window Functions

- Similar to aggregation functions
  - Computes one value based on multiple rows
  - But does not group



Aggregation function



Window function

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

Window Function Example

Sum up total salary per department

```
SELECT name, dept_id, salary,
       SUM(salary) OVER (PARTITION BY
       dept_id) AS dept_total
FROM employee
ORDER BY dept_id, name;
```

The **OVER** keyword signals a window function

PARTITION ⇒ disjoint set of rows in result set

name	dept_id	salary
Newt	NULL	75000
Dag	10	NULL
Ed	10	100000
Fred	10	60000
Jon	10	60000
Michael	10	70000
Newt	10	80000
Lebedev	20	65000
Pete	20	65000
Jeff	30	300000
Will	30	70000

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

Window Function Example

Sum up total salary per department

```
SELECT name, dept_id, salary,
        SUM(salary) OVER (PARTITION BY
        dept_id) AS dept_total
FROM employee
ORDER BY dept_id, name;
```

The **OVER** keyword signals a window function

PARTITION ⇒ disjoint set of rows in result set

name	dept_id	salary	dept_total
Newt	NULL	75000	75000
Dag	10	NULL	370000
Ed	10	100000	370000
Fred	10	60000	370000
Jon	10	60000	370000
Michael	10	70000	370000
Newt	10	80000	370000
Lebedev	20	65000	130000
Pete	20	65000	130000
Jeff	30	300000	370000
Will	30	70000	370000

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

Program Agenda

- Common table expressions
- Window functions
- UTF-8 support
- GIS
- SKIP LOCKED, NOWAIT
- JSON functions
- Index extensions
- Cost model
- Hints

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

UTF-8 Support

- Collations based on Unicode 9.0
- utf8mb4 as default character set
  - utf8mb4_0900_ai_ci as default collation
- Accent and case sensitive collations
  - Including 20+ language specific collations
  - Now also Japanese and Russian
- Significantly improved performance 👍
- Unicode support in regular expressions

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

New Default Character Set

- No change to existing tables
- Only has effect on new tables/schemas when character set is not explicitly specified

Upgrade recommendations

- Upgrade first, change character set and collation afterwards
- Don't mix collations
  - Error: "Illegal mix of collations"
  - Slower queries because indexes can't be used

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

Program Agenda

- Common table expressions
- Window functions
- UTF-8 support
- GIS
- SKIP LOCKED, NOWAIT
- JSON functions
- Index extensions
- Cost model
- Hints

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

GIS

- Geography support
  - Latitude-longitude on an ellipsoid
- Spatial reference systems (SRSs)
  - More than 5000 SRSs from the EPSG Geodetic Parameter Dataset
  - INFORMATION_SCHEMA.ST_SPATIAL_REFERENCE_SYSTEMS
  - CREATE/DROP SPATIAL REFERENCE SYSTEM
- SRID aware spatial datatypes, indexes and functions
- ST_Transform to convert between SRSs — **New in MySQL 8.0.13!**
- Functions and expressions as default values — **New in MySQL 8.0.13!**
  - Including ST_GeomFromText

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

Program Agenda

- Common table expressions
- Window functions
- UTF-8 support
- GIS
- **SKIP LOCKED, NOWAIT**
- JSON functions
- Index extensions
- Cost model
- Hints

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

SELECT … FOR UPDATE SKIP LOCKED

Problem: Hot row contention, multiple worker threads accessing the same rows.

SKIP LOCKED Solution:

- Only read rows that aren't locked
- InnoDB skips a locked row, and the next one goes to the result set

Example:

– Booking system: Skip orders that are pending

```
START TRANSACTION;
SELECT * FROM seats WHERE seat_no BETWEEN 2 AND 3 AND booked = 'NO'
FOR UPDATE SKIP LOCKED;
```

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

SELECT … FOR UPDATE NOWAIT

Problem: Hot row contention, multiple worker threads accessing the same rows.

NOWAIT Solution:

- Fail immediately if any rows are locked
- Without NOWAIT, the query waits for innodb_lock_wait_timeout (default 50 seconds) before giving up

Example:

  – Booking system: Fail if rows are locked

```
START TRANSACTION;
SELECT * FROM seats WHERE seat_no BETWEEN 2 AND 3 AND booked = 'NO'
FOR UPDATE NOWAIT;
ERROR 3572 (HY000): Statement aborted because lock(s) could not be acquired ...
```

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

Program Agenda

- Common table expressions
- Window functions
- UTF-8 support
- GIS
- SKIP LOCKED, NOWAIT
- **JSON functions**
- Index extensions
- Cost model
- Hints

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

JSON Functions

JSON_ARRAY_APPEND()
JSON_ARRAY_INSERT()
JSON_ARRAY()
JSON_CONTAINS_PATH()
JSON_CONTAINS()
JSON_DEPTH()
JSON_EXTRACT()
JSON_INSERT()
JSON_KEYS()
JSON_LENGTH()

JSON_MERGE[_PRESERVE]()
JSON_OBJECT()
JSON_QUOTE()
JSON_REMOVE()
JSON_REPLACE()
JSON_SEARCH()
JSON_SET()
JSON_TYPE()
JSON_UNQUOTE()
JSON_VALID()

JSON_PRETTY()
JSON_STORAGE_SIZE()
JSON_STORAGE_FREE()
JSON_ARRAYAGG()
JSON_OBJECTAGG()
JSON_MERGE_PATCH()
JSON_TABLE()

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

JSON Data Type

```
CREATE TABLE t1 (json_col JSON);

INSERT INTO t1 VALUES (
    '{ "people": [
        { "name":"John Smith", "address":"780 Mission St, San Francisco, CA 94103"},
        { "name":"Sally Brown", "address":"75 37th Ave S, St Cloud, MN 94103"},
        { "name":"John Johnson", "address":"1262 Roosevelt Trail, Raymond, ME 04071"}
    ] }'
);
```

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

JSON_TABLE

Convert JSON documents to relational tables

```
SELECT people.*
FROM t1,
     JSON_TABLE(json_col, '$.people[*]' COLUMNS (
                name VARCHAR(40) PATH '$.name',
                address VARCHAR(100) PATH '$.address')) AS people;
```

name	address
John Smith	780 Mission St, San Francisco, CA 94103
Sally Brown	75 37th Ave S, St Cloud, MN 9410
John Johnson	1262 Roosevelt Trail, Raymond, ME 04071

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

JSON_TABLE

Filter JSON data on SQL expression

```
SELECT people.*
FROM t1,
        JSON_TABLE(json_col, '$.people[*]' COLUMNS (
                name VARCHAR(40) PATH '$.name',
                address VARCHAR(100) PATH '$.address')) AS people
WHERE people.name LIKE 'John%';
```

name	address
John Smith	780 Mission St, San Francisco, CA 94103
John Johnson	1262 Roosevelt Trail, Raymond, ME 04071

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

JSON_TABLE Nested Arrays


```
[
  { "father":"John", "mother":"Mary",
    "marriage_date":"2003-12-05",
    "children": [ { "name":"Eric", "age":12 },
                  { "name":"Beth", "age":10 } ]
  },
  { "father":"Paul", "mother":"Laura",
    "children": [ { "name":"Sarah", "age":9},
                  { "name":"Noah", "age":3} ,
                  { "name":"Peter", "age":1} ]
  }
]
```


id	father	married	child_id	child	age
1	John	1	1	Eric	12
1	John	1	2	Beth	10
2	Paul	0	1	Sarah	9
2	Paul	0	2	Noah	3
2	Paul	0	3	Peter	1

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

JSON_TABLE Nested Arrays

```
JSON_TABLE (families, '$[*]' COLUMNS (
    id FOR ORDINALITY,
    father VARCHAR(30) PATH '$.father',
    married INTEGER EXISTS PATH
       '$.marriage_date',
    NESTED PATH '$.children[*]' COLUMNS (
       child_id FOR ORDINALITY,
       child VARCHAR(30) PATH '$.name',
       age INTEGER PATH '$.age'
    )
))
```

id	father	married	child_id	child	age
1	John	1	1	Eric	12
1	John	1	2	Beth	10
2	Paul	0	1	Sarah	9
2	Paul	0	2	Noah	3
2	Paul	0	3	Peter	1

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

JSON Array Aggregation

```
CREATE TABLE t1 (id INT, grp INT,
jsoncol JSON);

INSERT INTO t1 VALUES (1, 1,
'{"key1":"value1","key2":"value2"}');

INSERT INTO t1 VALUES (2, 1,
'{"keyA":"valueA","keyB":"valueB"}');

INSERT INTO t1 VALUES (3, 2,
'{"keyX":"valueX","keyY":"valueY"}');
```

```
SELECT JSON_ARRAYAGG(jsoncol)
FROM t1;

[{"key1":"value1","key2":"value2"},
{"keyA":"valueA","keyB":"valueB"},
{"keyX":"valueX","keyY":"valueY"}]
```

Yet to come: JSON_ARRAYAGG as window function

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

JSON Object Aggregation

```
CREATE TABLE t1 (id INT, grp INT,
jsoncol JSON);

INSERT INTO t1 VALUES (1, 1,
'{"key1":"value1","key2":"value2"}');

INSERT INTO t1 VALUES (2, 1,
'{"keyA":"valueA","keyB":"valueB"}');

INSERT INTO t1 VALUES (3, 2,
'{"keyX":"valueX","keyY":"valueY"}');
```

```
SELECT grp, JSON_OBJECTAGG(id, jsoncol)
FROM t1
GROUP BY grp;
```

1	{"1":{"key1":"value1","key2":"value2"}, "2":{"keyA":"valueA","keyB":"valueB"}}
2	{"3":{"keyX":"valueX","keyY":"valueY"}}

Yet to come: JSON_OBJECTAGG as window function

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

Partial Update of JSON

- Automatically detect potential in-place updates
  - JSON_SET
  - JSON_REPLACE
  - JSON_REMOVE
  - Source and target columns are the same
  - Updating existing value, not adding new one
  - Sufficient space is available
    - Smaller than or same size as some old value
- Only the diff is sent in binlog

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

Program Agenda

- Common table expressions
- Window functions
- UTF-8 support
- GIS
- SKIP LOCKED, NOWAIT
- JSON functions
- **Index extensions**
- Cost model
- Hints

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

Invisible Index

- Index is maintained by the storage engine, but invisible to the optimizer
  - Override with SET optimizer_switch='use_invisible_indexes=on';
- Primary key can't be invisible
- Use case: Check for performance drope *before* removing index

ALTER TABLE *t1* **ALTER INDEX** *idx* **INVISIBLE**;
SHOW INDEXES FROM *t1*;

Table	Key_name	Column_name	Visible
t1	idx	a	NO

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

Descending Index

```
CREATE TABLE t1 (
   a INT,
   b INT,
   INDEX a_b (a DESC, b ASC)
);
```

- In 5.7: Index in ascending order is created, server scans it backwards
- In 8.0: Index in descending order is created, server scans it forwards
  - Works on B-tree indexes only
  - Benefits:
    - Use indexes instead of filesort for ORDER BY clause with ASC/DESC sort key
    - Forward index scan is slightly faster than backward index scan

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

Functional Index — **New in 8.0.13**

- Index over an expression

```
CREATE TABLE t1 (col1 INT, col2 INT, INDEX func_index ((ABS(col1))));
CREATE INDEX idx1 ON t1 ((col1 + col2));
CREATE INDEX idx2 ON t1 ((col1 + col2), (col1 - col2), col1);
ALTER TABLE t1 ADD INDEX ((col1 * 40));
```

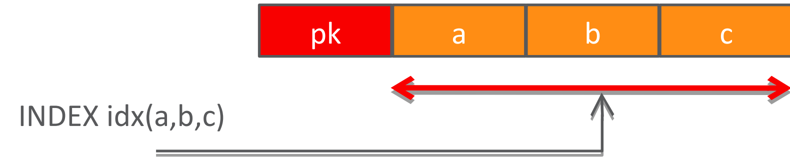
- No longer necessary to create an indexed generated column
- Not allowed as PRIMARY KEY

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

Index Skip-Scan — **New in 8.0.13**

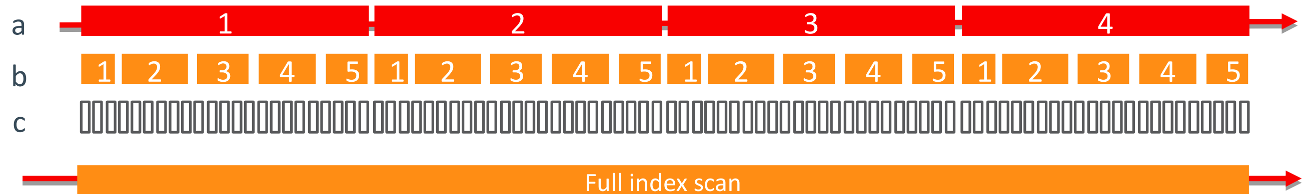
- Used for multi-column index
- Condition not on first part of index
- Cost-based choice



SELECT * FROM *t1* WHERE *b* < 3;

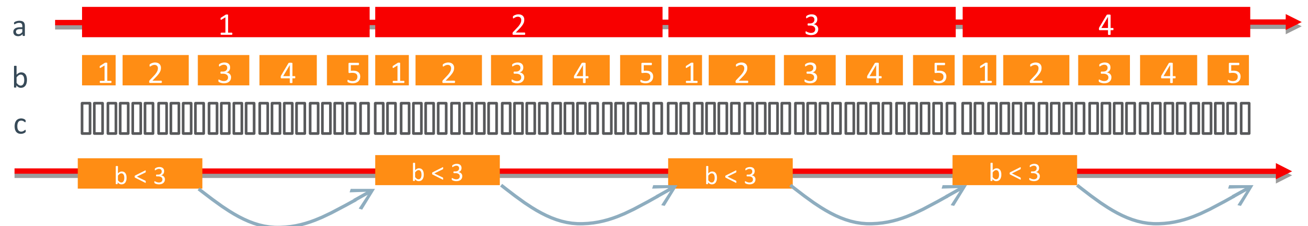
Before MySQL 8.0.13:

Full index scan



MySQL 8.0.13:

Skip-scan



Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

Thanks for the patch, Facebook!

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

Program Agenda

- Common table expressions
- Window functions
- UTF-8 support
- GIS
- SKIP LOCKED, NOWAIT
- JSON functions
- Index extensions
- Cost model
- Hints

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

Motivation for Improving the MySQL Cost Model

- Produce more correct cost estimates
  - Better decisions by the optimizer should improve performance
- Adapt to new hardware architectures
  - SSD, larger memories, caches
- More maintainable cost model implementation
  - Avoid hard coded “cost constants”
  - Refactoring of existing cost model code
- Configurable and tunable
- Make more of the optimizer cost-based

ORACLE

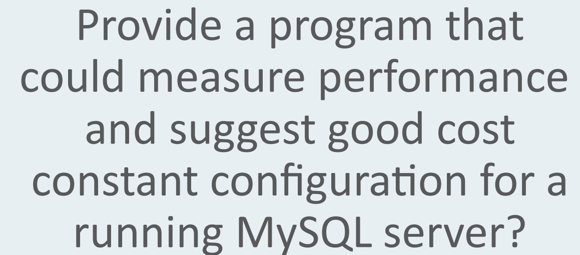
Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

New Storage Technologies

- Time to do a table scan of 10 million records:

Memory	5 s
SSD	20 - 146 s
Hard disk	32 - 1465 s

- Adjust cost model to support different storage technologies
- Provide configurable cost constants for different storage technologies

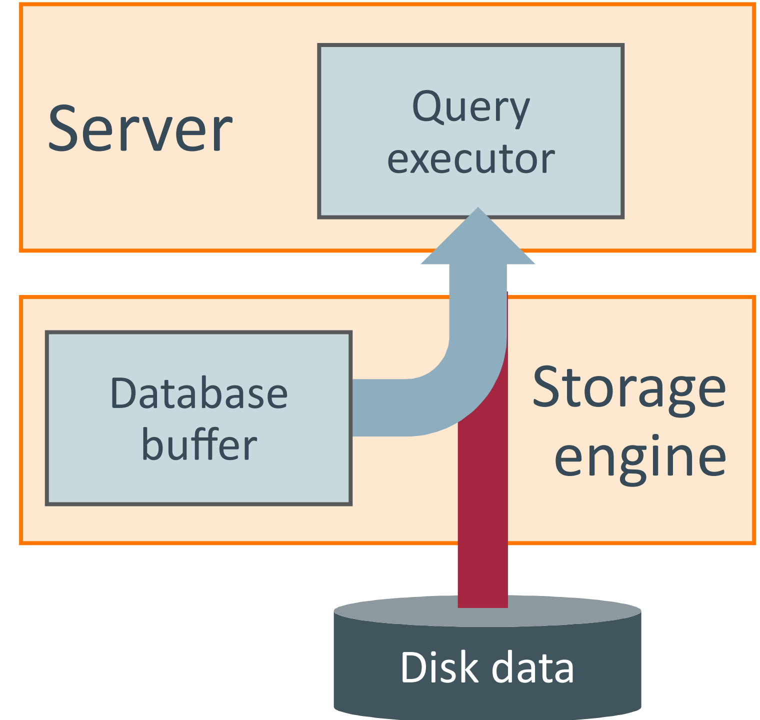


ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

Memory Buffer Aware Cost Estimates

- Storage engines:
  - Estimate for how much of data and indexes are in a memory buffer
  - Estimate for hit rate for memory buffer
- Optimizer cost model:
  - Take into account whether data is already in memory or need to be read from disk



Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

Histograms

- Provides the optimizer with information about column value distribution
- To create/recalculate histogram for a column:

  ANALYZE TABLE *table* **UPDATE HISTOGRAM ON** *column* **WITH** *n* **BUCKETS**;

- May use sampling
  - Sample size is based on available memory (histogram_generation_max_mem_size)
- Automatically chooses between two histogram types:
  - Singleton: One value per bucket
  - Equi-height: Multiple values per bucket

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

Program Agenda

- Common table expressions
- Window functions
- UTF-8 support
- GIS
- SKIP LOCKED, NOWAIT
- JSON functions
- Index extensions
- Cost model
- Hints

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

Join Order Hints

- No need to reorganize the FROM clause to add join order hints like you will for STRAIGHT_JOIN
- /*+ JOIN_ORDER(*t1*, *t2*, ...) */
- /*+ JOIN_PREFIX(*t1*, *t2*, ...) */
- /*+ JOIN_SUFFIX(..., *ty*, *tz*) */
- /*+ JOIN_FIXED_ORDER() */
  - Replacement for STRAIGHT_JOIN

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

View/Derived Table/CTE Merge Hints

- Views, derived tables and CTEs are, if possible, merged into outer query
- NO_MERGE can override default behavior

```
SELECT /*+ NO_MERGE(dt) */ *
FROM t1 JOIN (SELECT x, y FROM t2) dt ON  t1.x = dt.x;
```

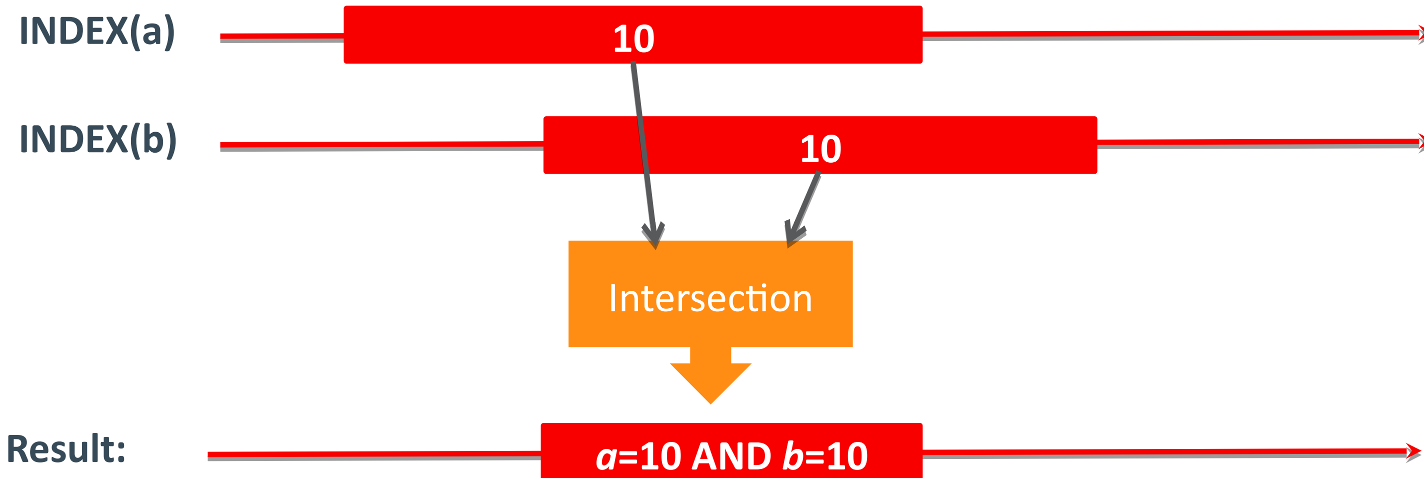
- MERGE will force a merge

```
SELECT /*+ MERGE(dt) */ *
FROM t1 JOIN (SELECT x, y FROM t2) dt ON  t1.x = dt.x;
```

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

Index Merge Execution



Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

Index Merge Hints

- Index merge: Merge rows from multiple range scans on a single table
- Algorithms: union, intersection, sort union
- Users can specify which indexes to use for index merge
  - /*+ INDEX_MERGE(*t1 idx1*, *idx2*, …) */
  - /*+ NO_INDEX_MERGE(*t1 idx1*, *idx2*, …) */

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

Hints to Set Session Variables

- Set a session variable for the duration of a single statement
- Examples:

```
SELECT /*+ SET_VAR(sort_buffer_size = 16M) */ name FROM people ORDER BY name;

INSERT /*+ SET_VAR(foreign_key_checks = OFF) */ INTO t2 VALUES (1, 1), (2, 2), (3, 3);

SELECT /*+ SET_VAR(optimizer_switch = 'condition_fanout_filter = off') */ *
FROM customer JOIN orders ON c_custkey = o_custkey
WHERE c_acctbal < 0 AND o_orderdate < '1993-01-01';
```

- Note: Not all session variables are settable through hints:

```
mysql> SELECT /*+ SET_VAR(max_allowed_packet=128M) */ * FROM t1;
Empty set, 1 warning (0,00 sec)
Warning (Code 4537): Variable 'max_allowed_packet' cannot be set using SET_VAR hint.
```

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

Feature descriptions and design details directly from the source.

http://mysqlserverteam.com/

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

Integrated Cloud

Applications & Platform Services

ORACLE

Copyright © 2018 Oracle and/or its affiliates. All rights reserved.

ORACLE®