ORACLE

The Past, Present and Future of EXPLAIN

Norvald H. Ryeng
Software Development Director
MySQL Optimizer Team
February 2, 2025

Agenda

1. EXPLAIN features
2. How EXPLAIN works
3. Predicting the future

EXPLAIN features

EXPLAIN [ANALYZE] [FORMAT= ⟨ format ⟩] [INTO @*var*] [FOR SCHEMA ⟨ schema ⟩] ⟨ statement ⟩ ;

EXPLAIN [ANALYZE] [FORMAT= ⟨ format ⟩] [INTO @*var*] [FOR SCHEMA ⟨ schema ⟩] ⟨ statement ⟩ ;

SELECT
INSERT
UPDATE
DELETE
REPLACE
TABLE

EXPLAIN [ANALYZE] [FORMAT= ⟨ format ⟩] [INTO @*var*] [FOR SCHEMA ⟨ schema ⟩] ⟨ statement ⟩ ;

TRADITIONAL
JSON
TREE

SELECT
INSERT
UPDATE
DELETE
REPLACE
TABLE

EXPLAIN [ANALYZE] [FORMAT= ⟨ format ⟩] [INTO @*var*] [FOR SCHEMA ⟨ schema ⟩] ⟨ statement ⟩ ;

TRADITIONAL
JSON
TREE

SELECT
INSERT
UPDATE
DELETE
REPLACE
TABLE

```
mysql> EXPLAIN FORMAT=TRADITIONAL SELECT Name FROM country WHERE Code Like 'A%';
+----+-------------+---------+------------+-------+---------------+---------+---------+------+------+----------+-------------+
| id | select_type | table   | partitions | type  | possible_keys | key     | key_len | ref  | rows | filtered | Extra       |
+----+-------------+---------+------------+-------+---------------+---------+---------+------+------+----------+-------------+
|  1 | SIMPLE      | country | NULL       | range | PRIMARY       | PRIMARY | 12      | NULL |   17 |   100.00 | Using where |
+----+-------------+---------+------------+-------+---------------+---------+---------+------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

EXPLAIN [ANALYZE] [FORMAT= ⟨ format ⟩] [INTO @*var*] [FOR SCHEMA ⟨ schema ⟩] ⟨ statement ⟩ ;

TRADITIONAL
JSON
TREE

SELECT
INSERT
UPDATE
DELETE
REPLACE
TABLE

```
mysql> EXPLAIN FORMAT=TRADITIONAL SELECT Name FROM country WHERE Code Like 'A%';
+----+-------------+---------+------------+-------+---------------+---------+---------+------+------+----------+-------------+
| id | select_type | table   | partitions | type  | possible_keys | key     | key_len | ref  | rows | filtered | Extra       |
+----+-------------+---------+------------+-------+---------------+---------+---------+------+------+----------+-------------+
|  1 | SIMPLE      | country | NULL       | range | PRIMARY       | PRIMARY | 12      | NULL |   17 |   100.00 | Using where |
+----+-------------+---------+------------+-------+---------------+---------+---------+------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

```
mysql> EXPLAIN FORMAT=TREE SELECT Name FROM country WHERE Code LIKE 'A%';
+-----------------------------------------------------------------------------------------------------------------------------------------------+
| EXPLAIN                                                                                                                                       |
+-----------------------------------------------------------------------------------------------------------------------------------------------+
| -> Filter: (country.`Code` like 'A%') (cost=3.67 rows=17)
    -> Index range scan on country using PRIMARY over ('A' <= Code <= 'A????????') (cost=3.67 rows=17) |
+-----------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set, 1 warning (0.00 sec)
```

EXPLAIN [ANALYZE] [FORMAT= ⟨ format ⟩] [INTO @*var*] [FOR SCHEMA ⟨ schema ⟩] ⟨ statement ⟩ ;

TRADITIONAL
JSON
TREE

SELECT
INSERT
UPDATE
DELETE
REPLACE
TABLE

EXPLAIN [ANALYZE] [FORMAT= ⟨ format ⟩] [INTO @*var*] [FOR SCHEMA ⟨ schema ⟩] ⟨ statement ⟩ ;

TRADITIONAL
JSON
TREE

SELECT
INSERT
UPDATE
DELETE
REPLACE
TABLE

New in 8.0.32

SET [GLOBAL | SESSION] explain_format = [DEFAULT | **TRADITIONAL** | JSON | TREE] ;

EXPLAIN [ANALYZE] [FORMAT= ⟨ format ⟩] [INTO @*var*] [FOR SCHEMA ⟨ schema ⟩] ⟨ statement ⟩ ;

TRADITIONAL
JSON
TREE

SELECT
INSERT
UPDATE
DELETE
REPLACE
TABLE

New in 8.0.32

SET [GLOBAL | SESSION] explain_format = [DEFAULT | **TRADITIONAL** | JSON | TREE] ;

EXPLAIN [ANALYZE] [FORMAT= ⟨ format ⟩] [INTO @*var*] [FOR SCHEMA ⟨ schema ⟩] ⟨ statement ⟩ ;

TRADITIONAL
JSON
TREE

SELECT
INSERT
UPDATE
DELETE
REPLACE
TABLE

SET [GLOBAL | SESSION] explain_format = [DEFAULT | **TRADITIONAL** | JSON | TREE] ;

SET [GLOBAL | SESSION] explain_json_format_version = [**1** | 2] ;

New in 8.3.0

EXPLAIN [ANALYZE] [FORMAT= ⟨ format ⟩] [INTO @*var*] [FOR SCHEMA ⟨ schema ⟩] ⟨ statement ⟩ ;

TRADITIONAL
JSON
TREE

SELECT
INSERT
UPDATE
DELETE
REPLACE
TABLE

SET [GLOBAL | SESSION] explain_format = [DEFAULT | **TRADITIONAL** | JSON | TREE] ;

SET [GLOBAL | SESSION] explain_json_format_version = [**1** | 2] ;

New in 8.3.0

```
mysql> EXPLAIN FORMAT=JSON SELECT Name FROM country WHERE Code LIKE 'A%';
+-----------------------------------------------------------------------------+
| EXPLAIN                                                          |
+-----------------------------------------------------------------------------+
| {
  "query_block": {
    "select_id": 1,
    "cost_info": {
      "query_cost": "3.67"
    },
    "table": {
      "table_name": "country",
      "access_type": "range",
      "possible_keys": [
        "PRIMARY"
      ],
      "key": "PRIMARY",
      "used_key_parts": [
        "Code"
      ],
      "key_length": "12",
      "rows_examined_per_scan": 17,
      "rows_produced_per_join": 17,
      "filtered": "100.00",
      "cost_info": {
        "read_cost": "1.97",
        "eval_cost": "1.70",
        "prefix_cost": "3.67",
        "data_read_per_join": "16K"
      },
      "used_columns": [
        "Code",
        "Name"
      ],
      "attached_condition": "(`world`.`country`.`Code` like 'A%')"
    }
  }
}                                                          |
+-----------------------------------------------------------------------------+
1 row in set, 1 warning (0.00 sec)
```

@*var*] [FOR SCHEMA ⟨ schema ⟩] ⟨ statement ⟩ ;

SELECT
INSERT
UPDATE
DELETE
REPLACE
TABLE

LT | **TRADITIONAL** | JSON | TREE] ;

SET [GLOBAL | SESSION] explain_json_format_version = [1 | 2] ;

New in 8.3.0

2025–02–02

```
mysql> EXPLAIN FORMAT=JSON SELECT Name FROM country WHERE Code LIKE 'A%';
+----------------------------------------------------------------------------+
| EXPLAIN                                                         |
+----------------------------------------------------------------------------+
| {
  "query_block": {
    "select_id": 1,
    "cost_info": {
      "query_cost": "3.67"
    },
    "table": {
      "table_name": "country",
      "access_type": "range",
      "possible_keys": [
        "PRIMARY"
      ],
      "key": "PRIMARY",
      "used_key_parts": [
        "Code"
      ],
      "key_length": "12",
      "rows_examined_per_scan": 17,
      "rows_produced_per_join": 17,
      "filtered": "100.00",
      "cost_info": {
        "read_cost": "1.97",
        "eval_cost": "1.70",
        "prefix_cost": "3.67",
        "data_read_per_join": "16K"
      },
      "used_columns": [
        "Code",
        "Name"
      ],
      "attached_condition": "(`world`.`country`.`Code` like 'A%')"
    }
  }
}                                                        |
+----------------------------------------------------------------------------+
1 row in set, 1 warning (0.00 sec)
```

```
mysql> EXPLAIN FORMAT=JSON SELECT Name FROM country WHERE Code LIKE 'A%'\G
EXPLAIN: {
  "query": "/* select#1 */ select `world`.`country`.`Name` AS `Name` from `world`.`country` where (`world`.`country`.`Code` like 'A%')",
  "query_plan": {
    "inputs": [
      {
        "ranges": [
          "('A' <= Code <= 'A????????')"
        ],
        "covering": false,
        "operation": "Index range scan on country using PRIMARY over ('A' <= Code <= 'A????????')",
        "index_name": "PRIMARY",
        "table_name": "country",
        "access_type": "index",
        "key_columns": [
          "Code"
        ],
        "schema_name": "world",
        "used_columns": [
          "Code",
          "Name"
        ],
        "estimated_rows": 17.0,
        "index_access_type": "index_range_scan",
        "estimated_total_cost": 3.668778400708174
      }
    ],
    "condition": "(country.`Code` like 'A%')",
    "operation": "Filter: (country.`Code` like 'A%')",
    "access_type": "filter",
    "estimated_rows": 17.0,
    "filter_columns": [
      "world.country.`Code`"
    ],
    "estimated_total_cost": 3.668778400708174
  },
  "query_type": "select",
  "json_schema_version": "2.0"
}
1 row in set, 1 warning (0.01 sec)
```

SET [GLOBAL | SESSION] explain_json_format_version = [1 | 2] ;

New in 8.3.0

  2025–02–02

```
mysql> EXPLAIN FORMAT=JSON SELECT Name FROM country WHERE Code LIKE 'A%';
+--------------------------------------------------------------------------+
| EXPLAIN                                                                  |
+--------------------------------------------------------------------------+
| {
  "query_block": {
    "select_id": 1,
    "cost_info": {
      "query_cost": "3.67"
    },
    "table": {
      "table_name": "country",
      "access_type": "range",
      "possible_keys": [
        "PRIMARY"
      ],
      "key": "PRIMARY",
      "used_key_parts": [
        "Code"
      ],
      "key_length": "12",
      "rows_examined_per_scan": 17,
      "rows_produced_per_join": 17,
      "filtered": "100.00",
      "cost_info": {
        "read_cost": "1.97",
        "eval_cost": "1.70",
        "prefix_cost": "3.67",
        "data_read_per_join": "16K"
      },
      "used_columns": [
        "Code",
        "Name"
      ],
      "attached_condition": "(`world`.`country`.`Code` like 'A%')"
    }
  }
}                                                                        |
+--------------------------------------------------------------------------+
1 row in set, 1 warning (0.00 sec)
```

```
mysql> EXPLAIN FORMAT=JSON SELECT Name FROM country WHERE Code LIKE 'A%'\G
EXPLAIN: {
  "query": "/* select#1 */ select `world`.`country`.`Name` AS `Name` from `world`.`country` where (`world`.`country`.`Code` like 'A%')",
  "query_plan": {
    "inputs": [
      {
        "ranges": [
          "('A' <= Code <= 'A????????')"
        ],
        "covering": false,
        "operation": "Index range scan on country using PRIMARY over ('A' <= Code <= 'A????????')",
        "index_name": "PRIMARY",
        "table_name": "country",
        "access_type": "index",
        "key_columns": [
          "Code"
        ],
        "schema_name": "world",
        "used_columns": [
          "Code",
          "Name"
        ],
        "estimated_rows": 17.0,
        "index_access_type": "index_range_scan",
        "estimated_total_cost": 3.668778400708174
      }
    ],
    "condition": "(country.`Code` like 'A%')",
    "operation": "Filter: (country.`Code` like 'A%')",
    "access_type": "filter",
    "estimated_rows": 17.0,
    "filter_columns": [
      "world.country.`Code`"
    ],
    "estimated_total_cost": 3.668778400708174
  },
  "query_type": "select",
  "json_schema_version": "2.0"
}
1 row in set, 1 warning (0.01 sec)
```

New in 9.2.0

SET [GLOBAL | SESSION] explain_json_format_version = [1 | 2] ;

New in 8.3.0

EXPLAIN [ANALYZE] [FORMAT= ⟨ format ⟩] [INTO @*var*] [FOR SCHEMA ⟨ schema ⟩] ⟨ statement ⟩ ;

TRADITIONAL
JSON
TREE

SELECT
INSERT
UPDATE
DELETE
REPLACE
TABLE

SET [GLOBAL | SESSION] explain_format = [DEFAULT | **TRADITIONAL** | JSON | TREE] ;

SET [GLOBAL | SESSION] explain_json_format_version = [**1** | 2] ;

EXPLAIN [ANALYZE [FORMAT= ⟨ format ⟩] [INTO @*var*] [FOR SCHEMA ⟨ schema ⟩] ⟨ statement ⟩ ;

TRADITIONAL
JSON
TREE

SELECT
INSERT
UPDATE
DELETE
REPLACE
TABLE

SET [GLOBAL | SESSION] explain_format = [DEFAULT | **TRADITIONAL** | JSON | TREE] ;

SET [GLOBAL | SESSION] explain_json_format_version = [**1** | 2] ;

EXPLAIN [ANALYZE [FORMAT= ⟨ format ⟩] [INTO @*var*] [FOR SCHEMA ⟨ schema ⟩] ⟨ statement ⟩ ;

~~TRADITIONAL~~
JSON
TREE

SELECT
INSERT
UPDATE
DELETE
REPLACE
TABLE

SET [GLOBAL | SESSION] explain_format = [DEFAULT | **TRADITIONAL** | JSON | TREE] ;

SET [GLOBAL | SESSION] explain_json_format_version = [~~1~~ | 2] ;

EXPLAIN [ANALYZE] [FORMAT= ⟨ format ⟩] [INTO @*var*] [FOR SCHEMA ⟨ schema ⟩] ⟨ statement ⟩ ;

TRADITIONAL
JSON
TREE

SELECT
INSERT
UPDATE
DELETE
REPLACE
TABLE

SET [GLOBAL | SESSION] explain_format = [DEFAULT | **TRADITIONAL** | JSON | TREE] ;

SET [GLOBAL | SESSION] explain_json_format_version = [**1** | 2] ;

EXPLAIN [ANALYZE] [FORMAT= ⟨ format ⟩] [INTO @*var* [FOR SCHEMA ⟨ schema ⟩] ⟨ statement ⟩ ;

TRADITIONAL
JSON
TREE

SELECT
INSERT
UPDATE
DELETE
REPLACE
TABLE

SET [GLOBAL | SESSION] explain_format = [DEFAULT | **TRADITIONAL** | JSON | TREE] ;

SET [GLOBAL | SESSION] explain_json_format_version = [**1** | 2] ;

EXPLAIN [ANALYZE] [FORMAT= ⟨ format ⟩] [INTO @*var* [FOR SCHEMA ⟨ schema ⟩] ⟨ statement ⟩ ;

~~TRADITIONAL~~
JSON
~~TREE~~

SELECT
INSERT
UPDATE
DELETE
REPLACE
TABLE

SET [GLOBAL | SESSION] explain_format = [DEFAULT | **TRADITIONAL** | JSON | TREE] ;

SET [GLOBAL | SESSION] explain_json_format_version = [**1** | 2] ;

EXPLAIN [ANALYZE] [FORMAT= ⟨ format ⟩] [INTO @*var*] [FOR SCHEMA ⟨ schema ⟩] ⟨ statement ⟩ ;

TRADITIONAL
JSON
TREE

SELECT
INSERT
UPDATE
DELETE
REPLACE
TABLE

SET [GLOBAL | SESSION] explain_format = [DEFAULT | **TRADITIONAL** | JSON | TREE] ;

SET [GLOBAL | SESSION] explain_json_format_version = [**1** | 2] ;

EXPLAIN [ANALYZE] [FORMAT= ⟨ format ⟩] [INTO @*var*] [FOR SCHEMA ⟨ schema ⟩] ⟨ statement ⟩ ;

TRADITIONAL
JSON
TREE

SELECT
INSERT
UPDATE
DELETE
REPLACE
TABLE

SET [GLOBAL | SESSION] explain_format = [DEFAULT | **TRADITIONAL** | JSON | TREE] ;

SET [GLOBAL | SESSION] explain_json_format_version = [**1** | 2] ;

EXPLAIN [ANALYZE] [FORMAT= ⟨ format ⟩] [INTO @*var*] [FOR SCHEMA ⟨ schema ⟩] ⟨ statement ⟩ ;

TRADITIONAL
JSON
TREE

SELECT
INSERT
UPDATE
DELETE
REPLACE
TABLE

SET [GLOBAL | SESSION] explain_format = [DEFAULT | **TRADITIONAL** | JSON | TREE] ;

SET [GLOBAL | SESSION] explain_json_format_version = [**1** | 2] ;

EXPLAIN [ANALYZE] [FORMAT= ⟨ format ⟩] [INTO @*var*] [FOR SCHEMA ⟨ schema ⟩] ⟨ statement ⟩ ;

TRADITIONAL
JSON
TREE

FOR CONNECTION ⟨ id ⟩

New in 5.7.2

SET [GLOBAL | SESSION] explain_format = [DEFAULT | **TRADITIONAL** | JSON | TREE] ;

SET [GLOBAL | SESSION] explain_json_format_version = [**1** | 2] ;

EXPLAIN ~~[ANALYZE]~~ [FORMAT= ⟨ format ⟩] ~~[INTO @*var*]~~ ~~[FOR SCHEMA ⟨ schema ⟩]~~ ⟨ statement ⟩ ;

TRADITIONAL
JSON
TREE

FOR CONNECTION ⟨ id ⟩

New in 5.7.2

SET [GLOBAL | SESSION] explain_format = [DEFAULT | **TRADITIONAL** | JSON | TREE] ;

SET [GLOBAL | SESSION] explain_json_format_version = [**1** | 2] ;

EXPLAIN [ANALYZE] [FORMAT= ⟨ format ⟩] [INTO @*var*] [FOR SCHEMA ⟨ schema ⟩] ⟨ statement ⟩ ;

TRADITIONAL
JSON
TREE

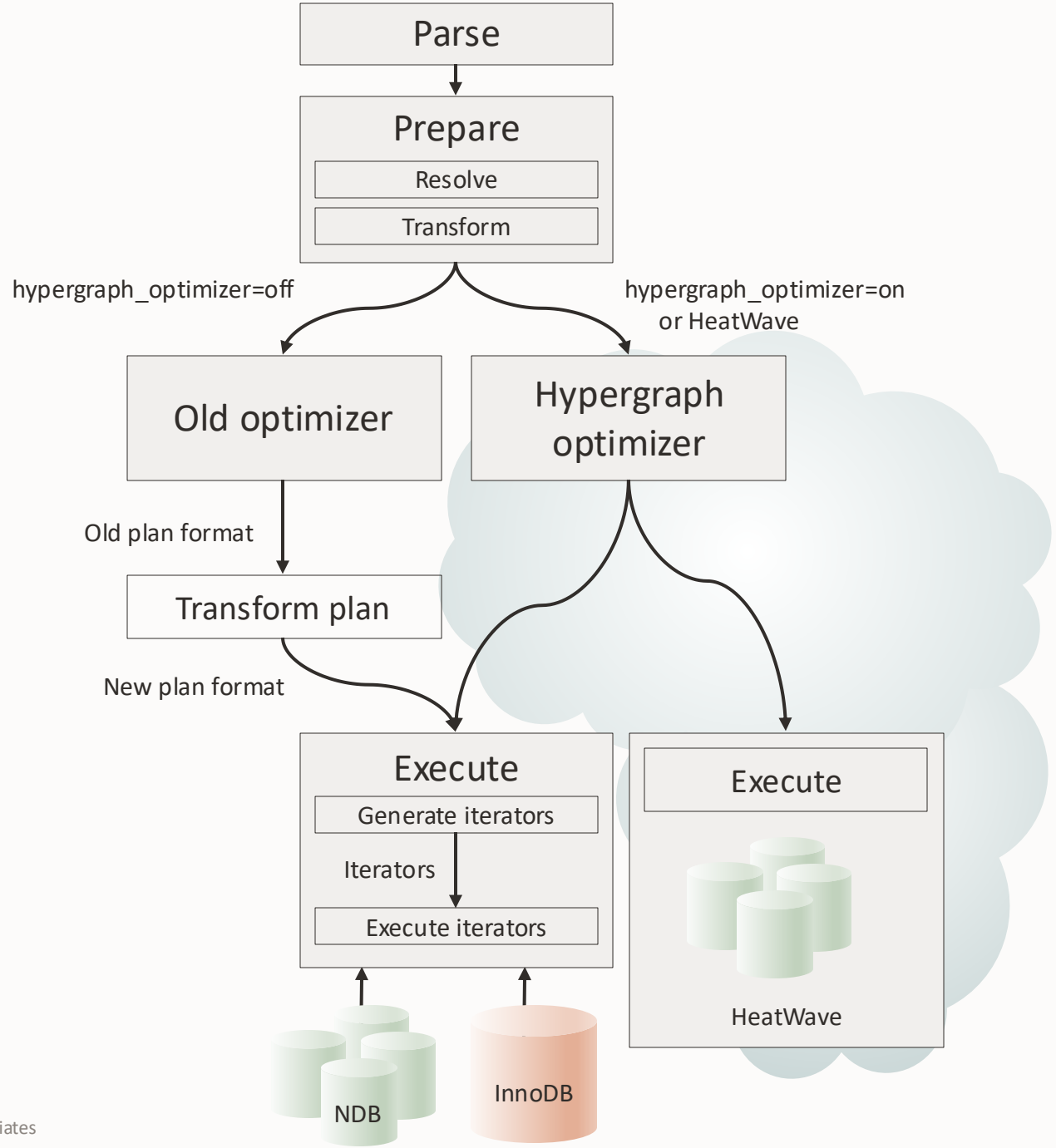
SELECT
INSERT
UPDATE
DELETE
REPLACE
TABLE

SET [GLOBAL | SESSION] explain_format = [DEFAULT | **TRADITIONAL** | JSON | TREE] ;

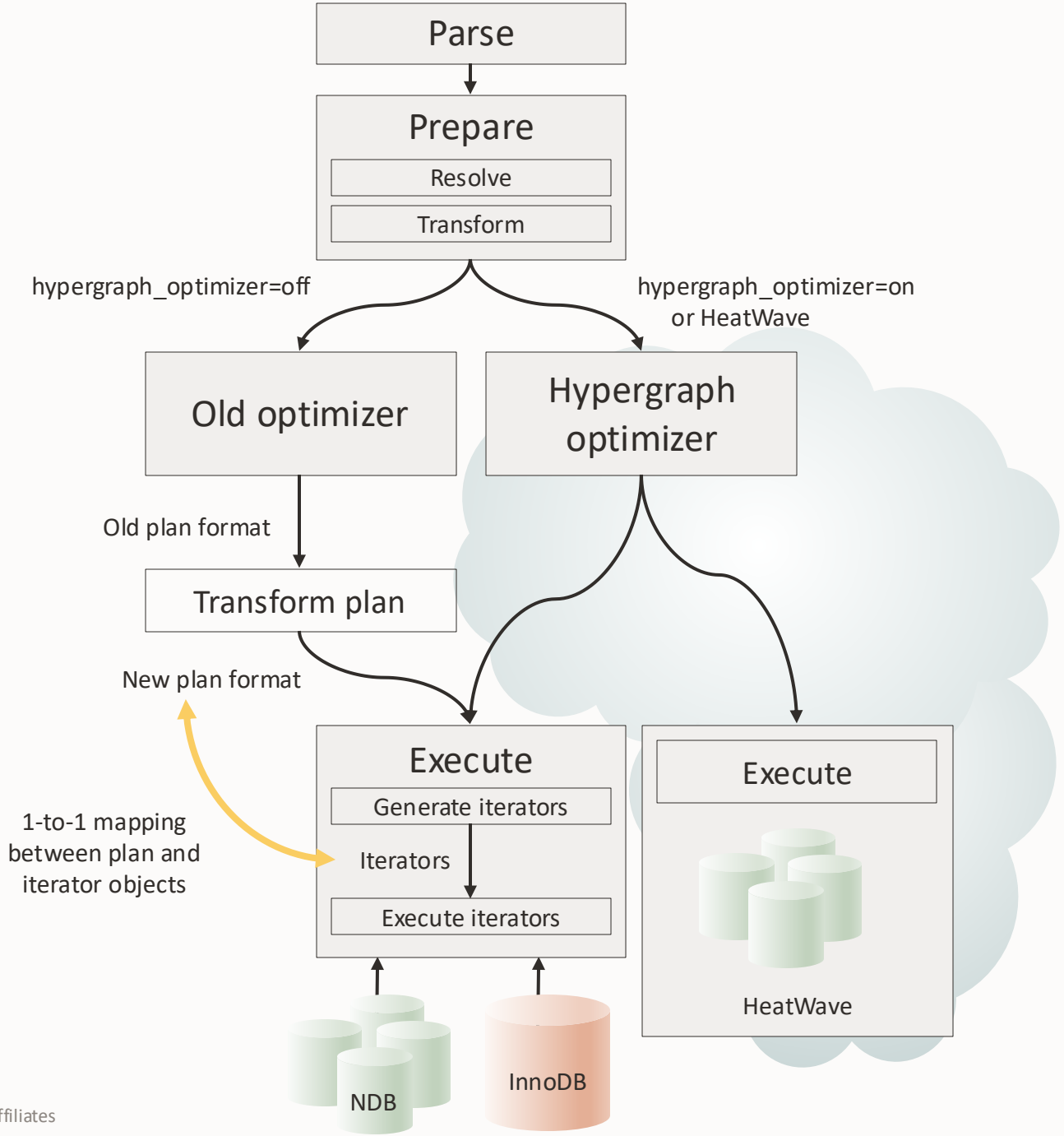
SET [GLOBAL | SESSION] explain_json_format_version = [**1** | 2] ;

How EXPLAIN works

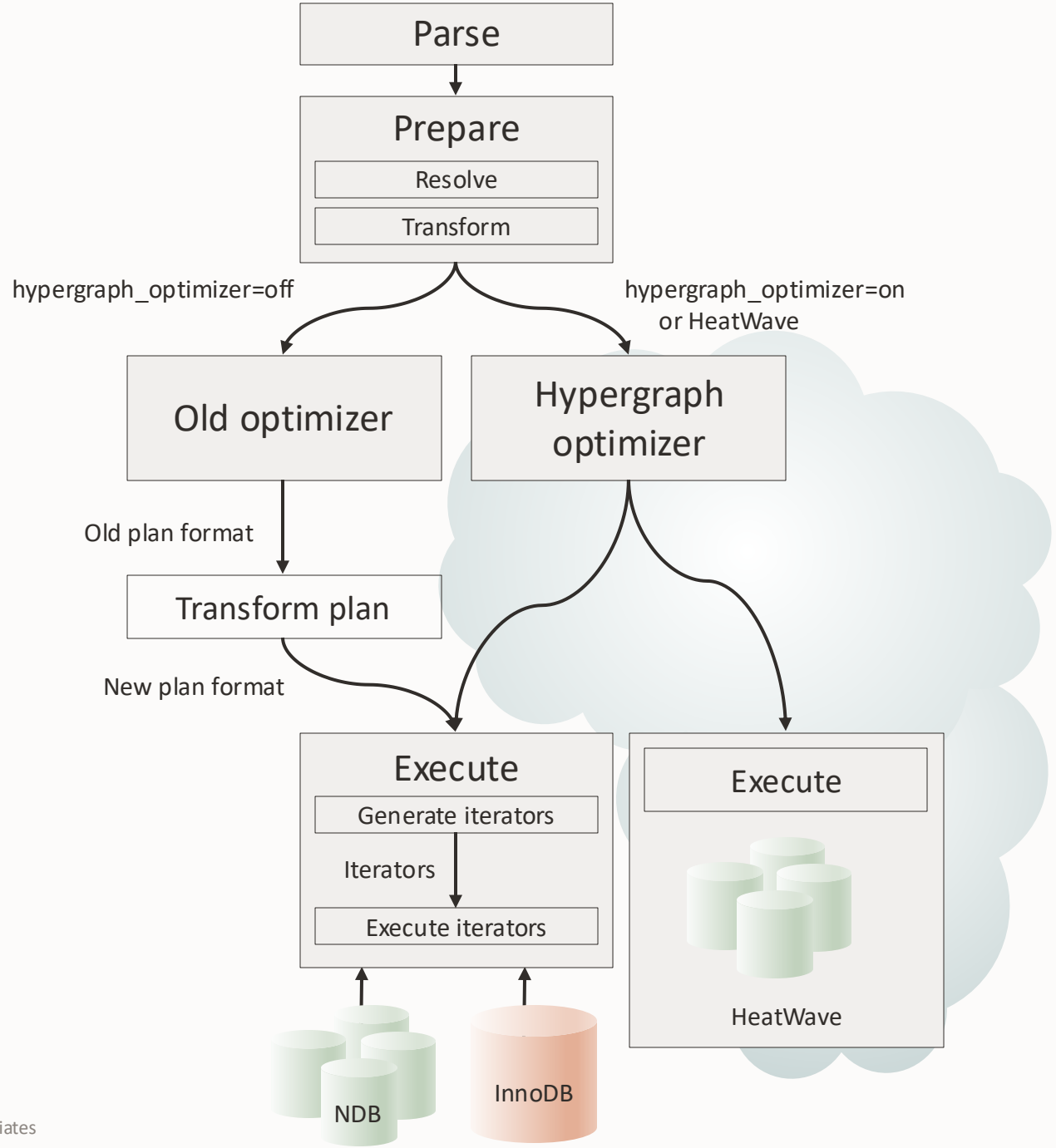
MySQL 9.0+



MySQL 9.0+

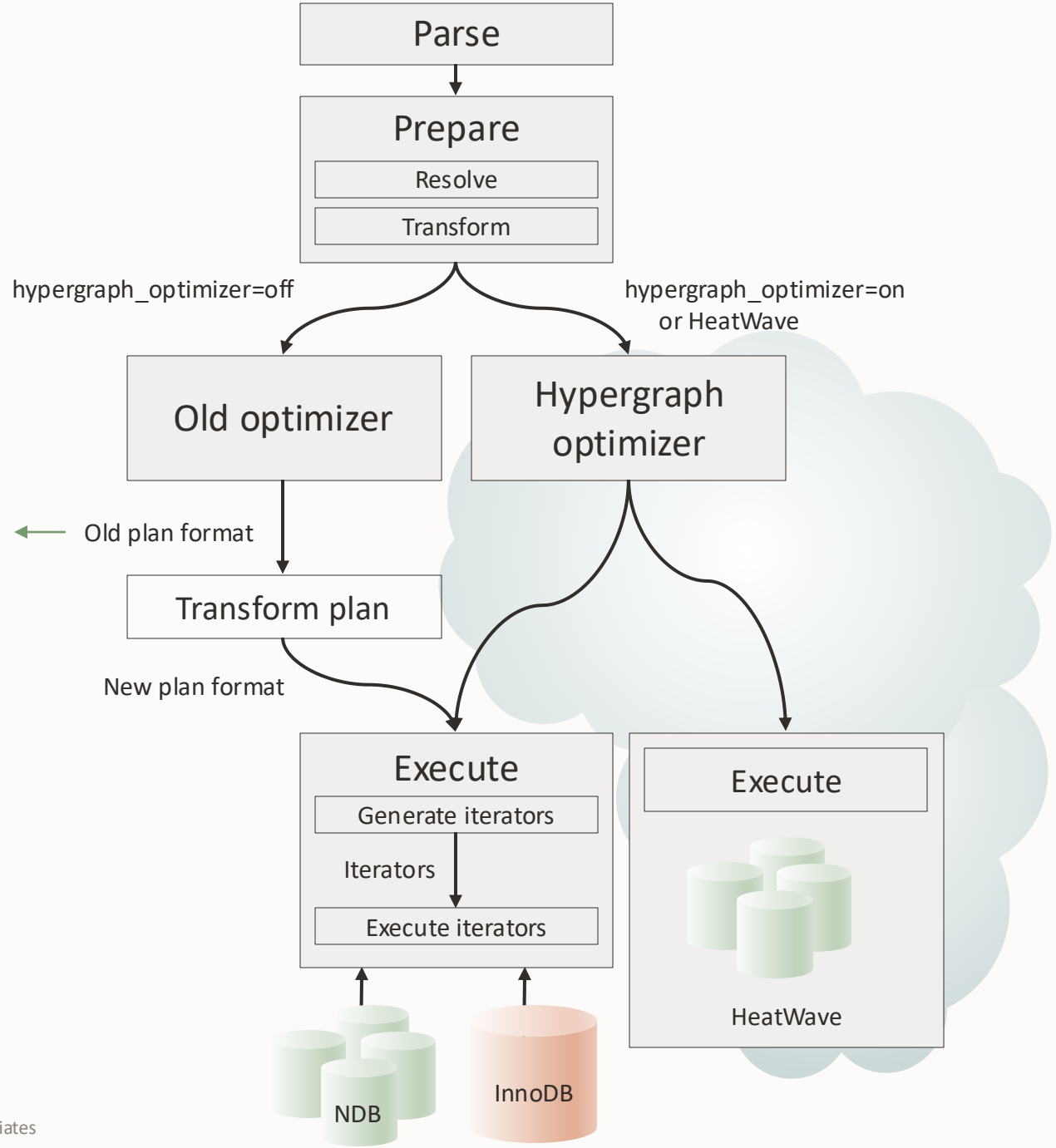


MySQL 9.0+



MySQL 9.0+

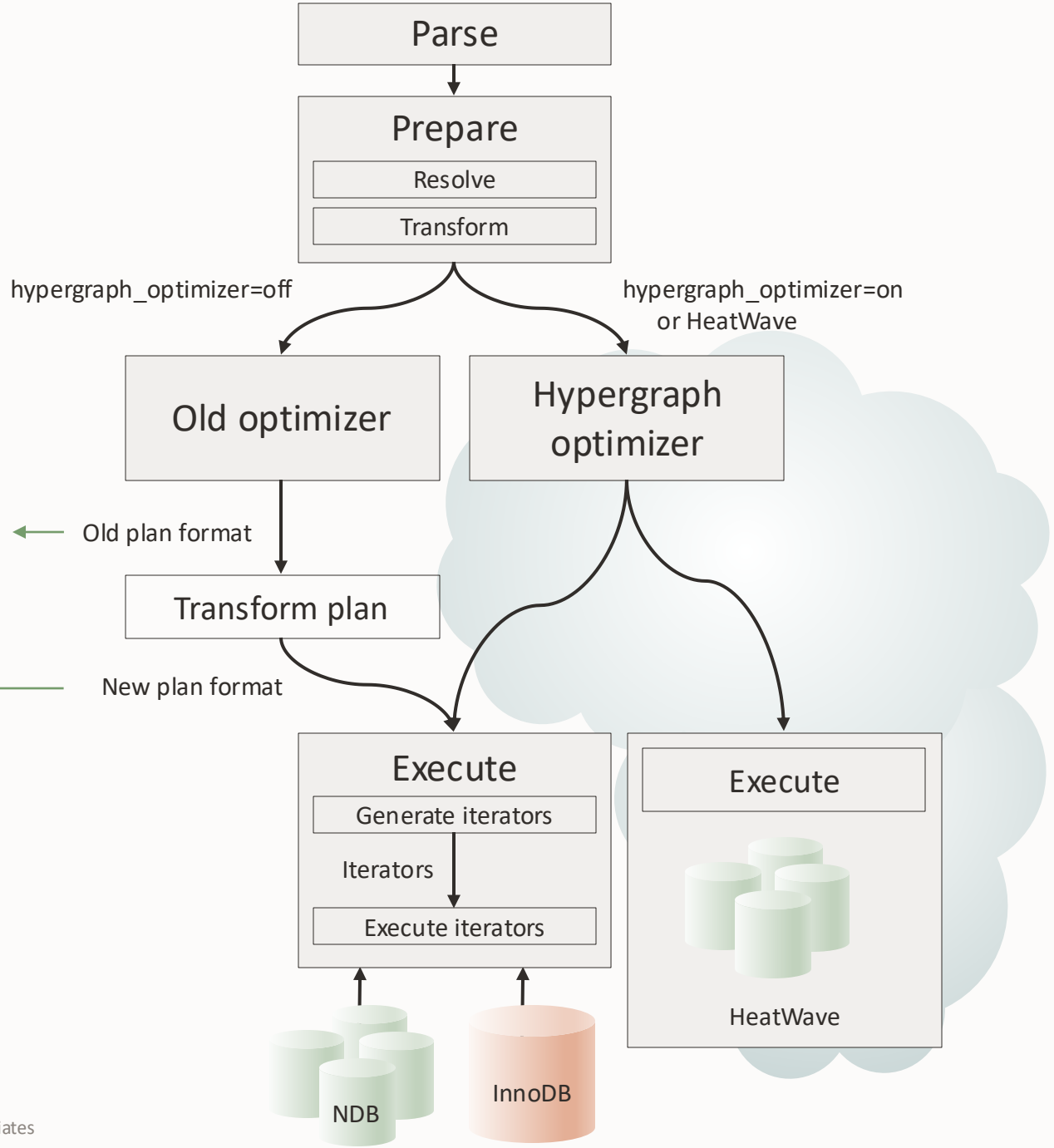
FORMAT=TRADITIONAL
FORMAT=JSON (v1)



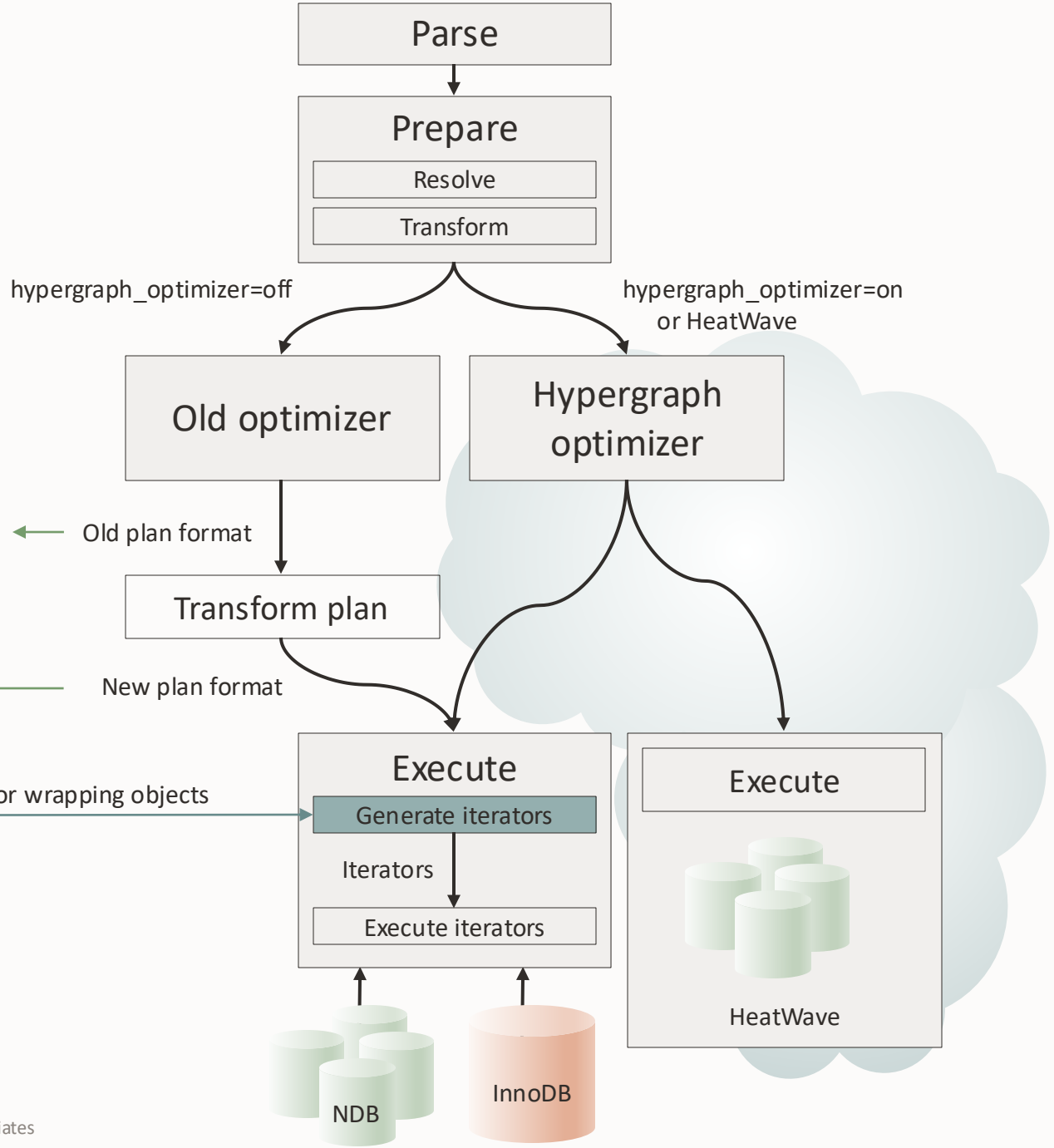
MySQL 9.0+

FORMAT=TRADITIONAL
FORMAT=JSON (v1)

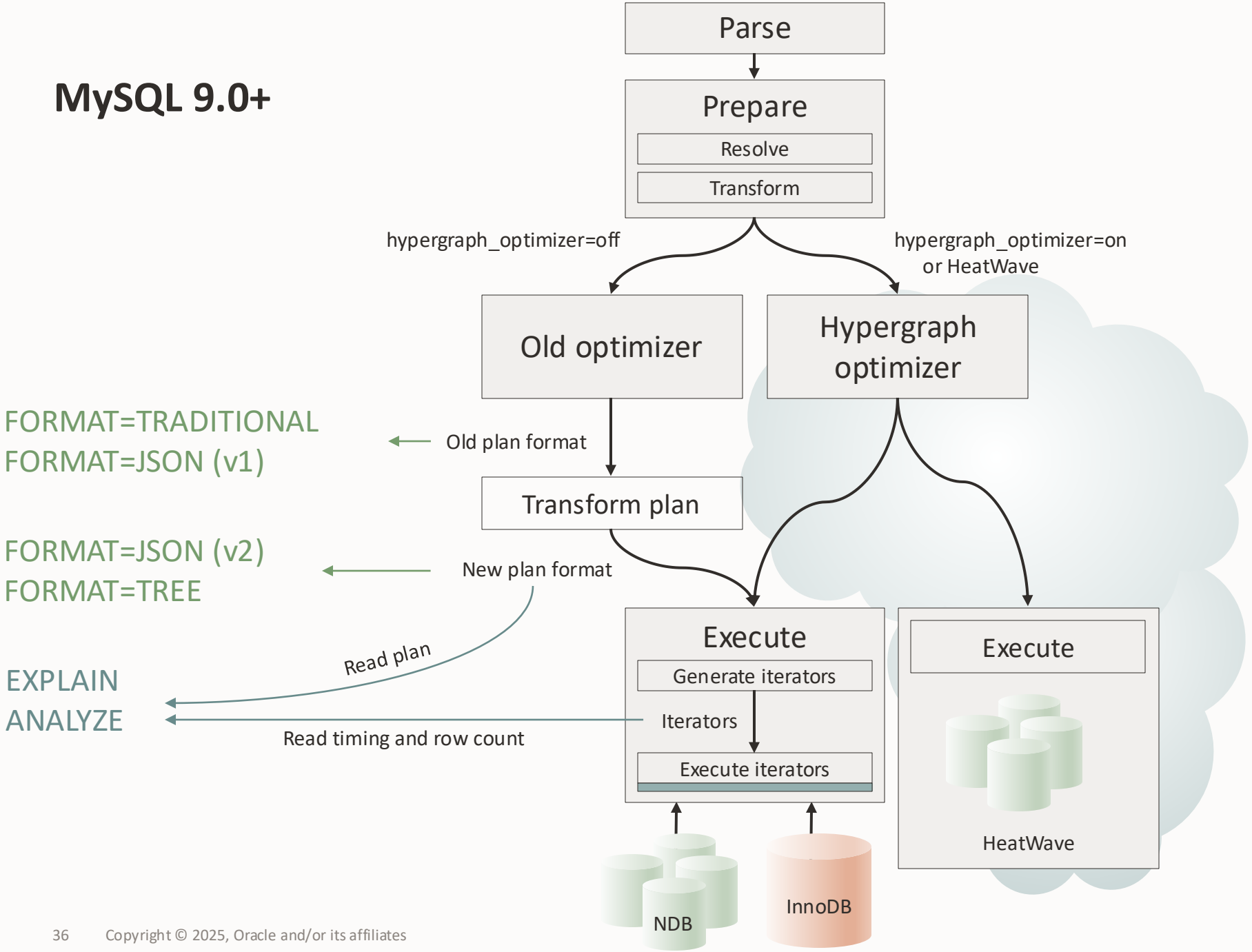
FORMAT=JSON (v2)
FORMAT=TREE



MySQL 9.0+



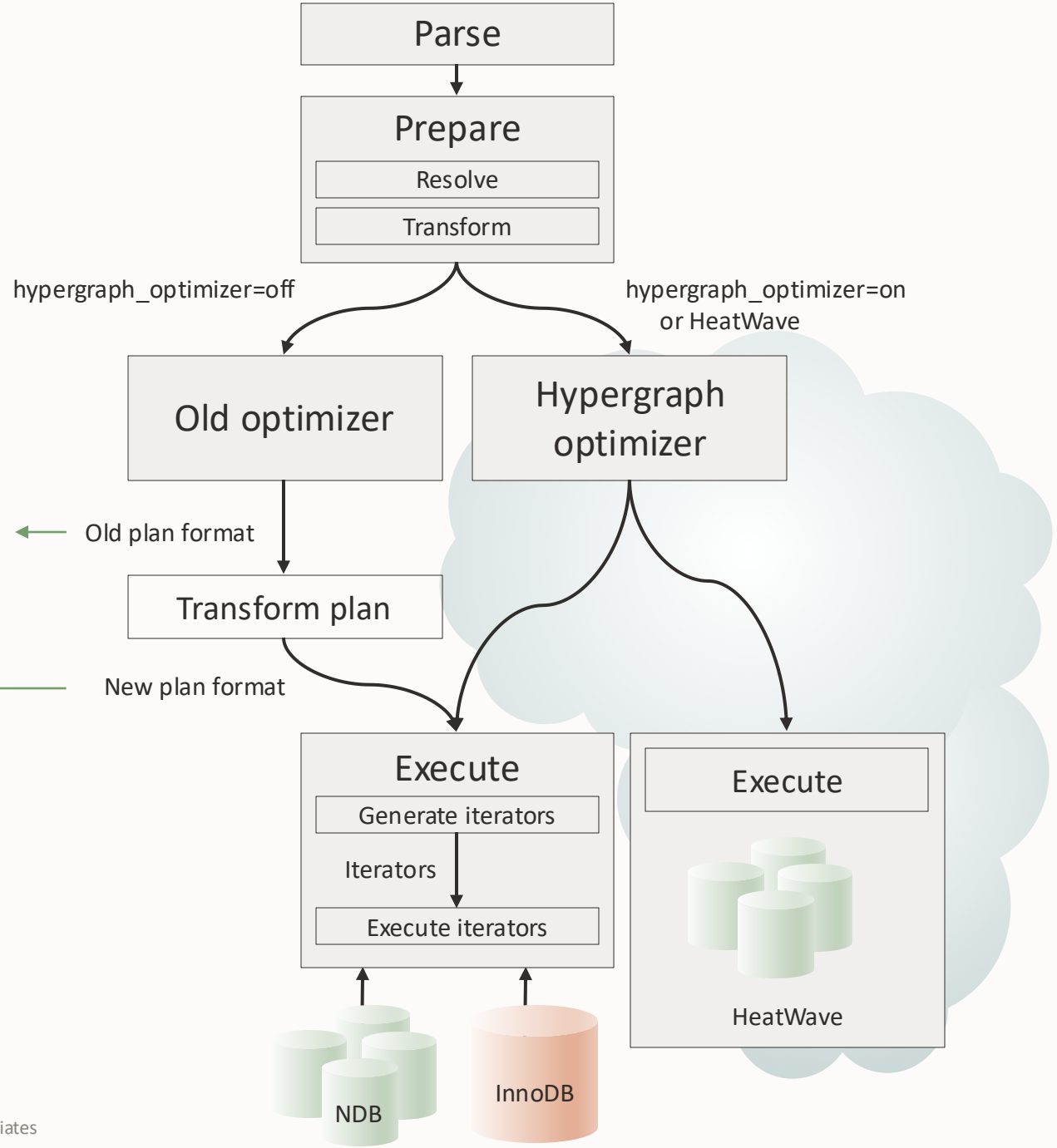
MySQL 9.0+



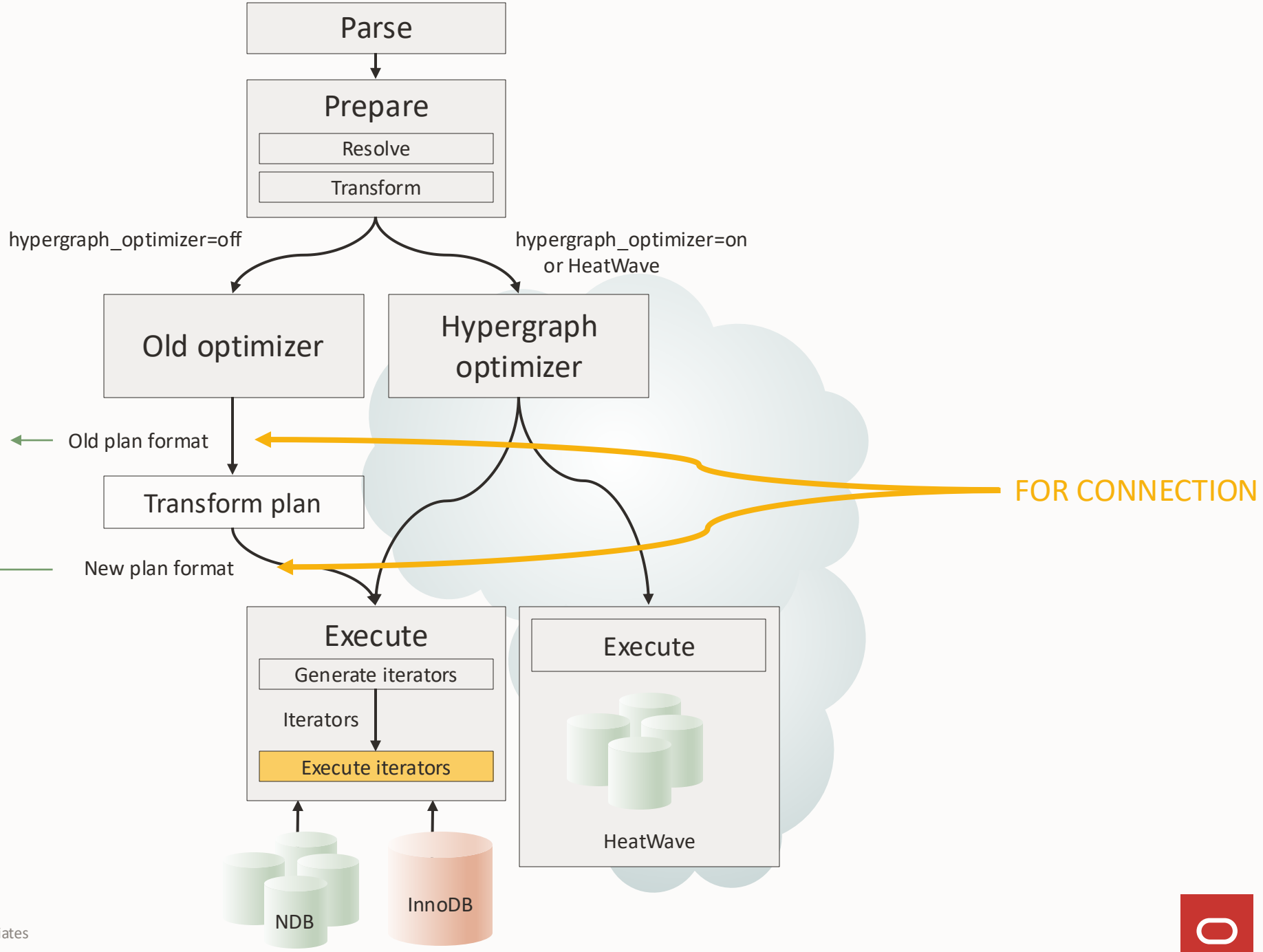
MySQL 9.0+

FORMAT=TRADITIONAL
FORMAT=JSON (v1)

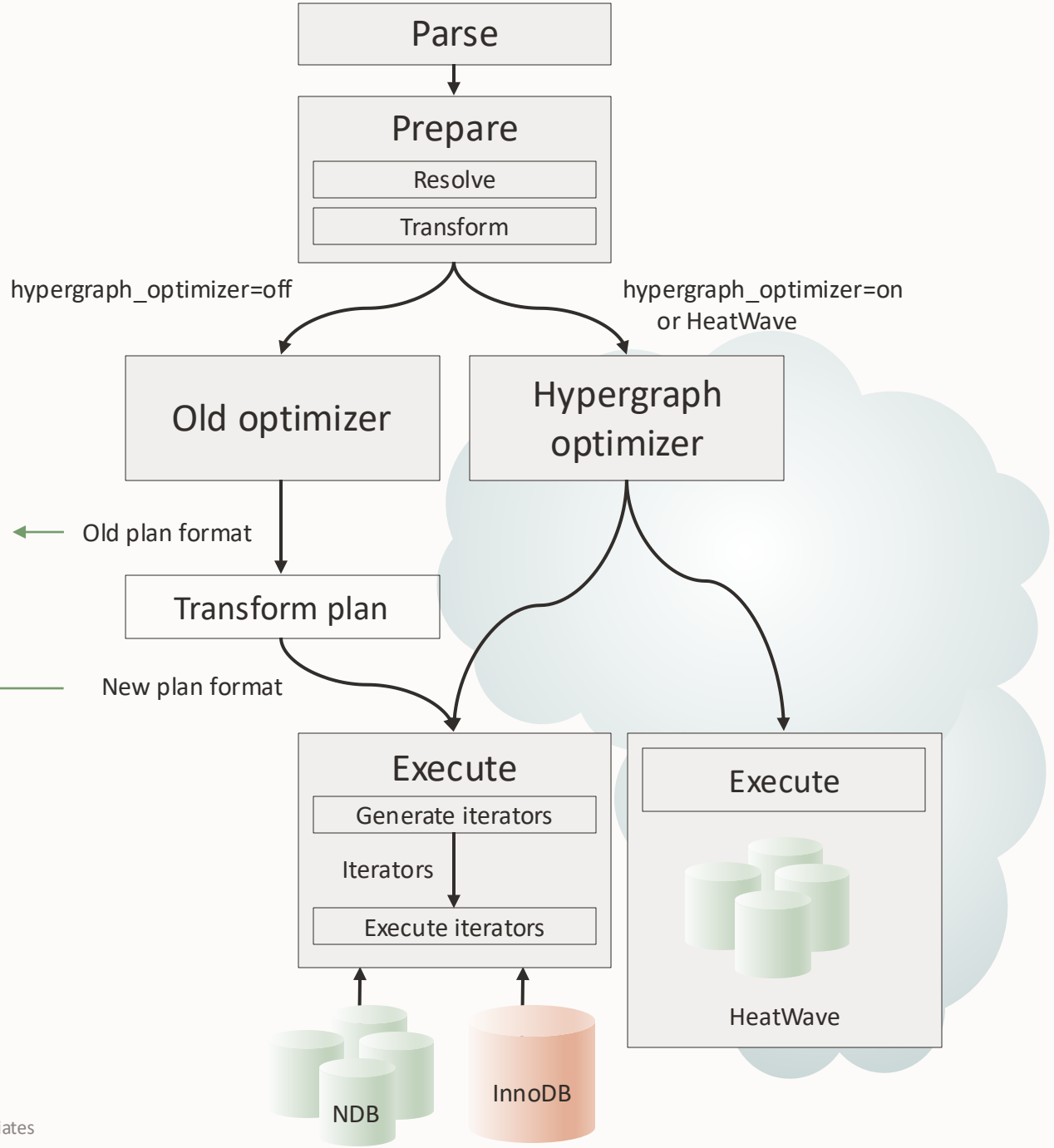
FORMAT=JSON (v2)
FORMAT=TREE



MySQL 9.0+



MySQL 9.0+



FORMAT=TRADITIONAL
FORMAT=JSON (v1)

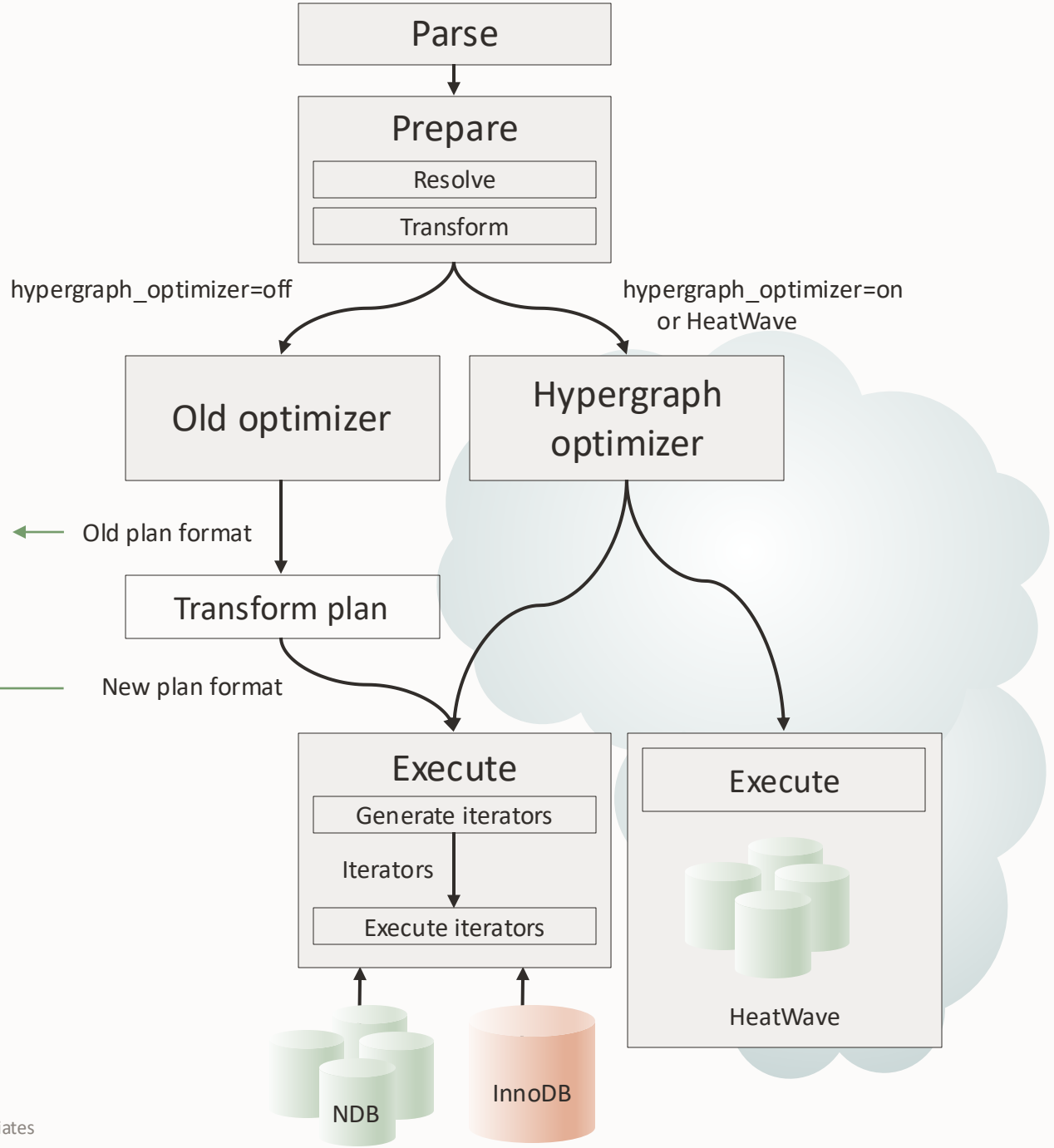
FORMAT=JSON (v2)
FORMAT=TREE

Predicting the future

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.

MySQL 9.0+



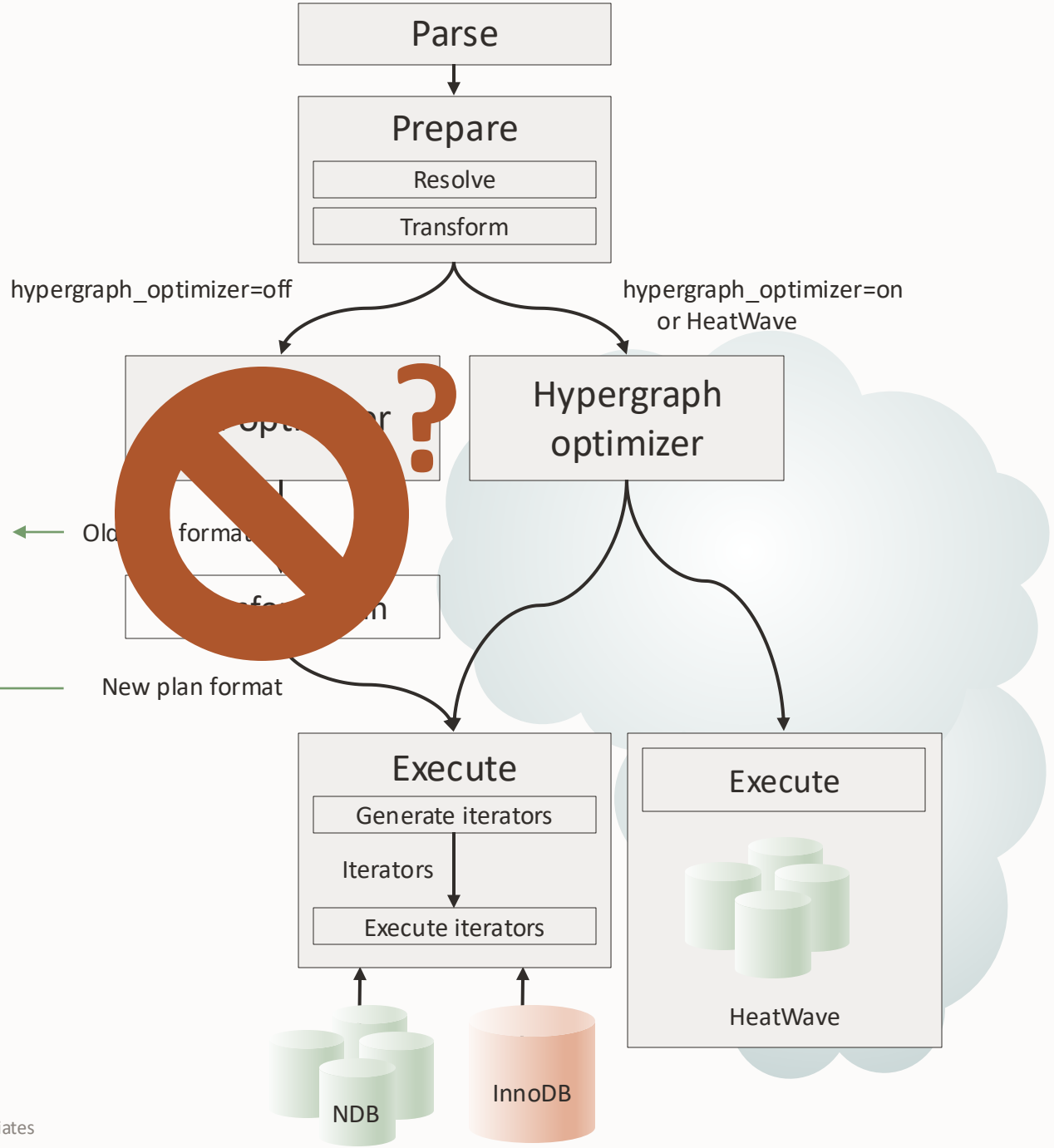
FORMAT=TRADITIONAL
FORMAT=JSON (v1)

FORMAT=JSON (v2)
FORMAT=TREE

Future MySQL?

FORMAT=TRADITIONAL
FORMAT=JSON (v1)

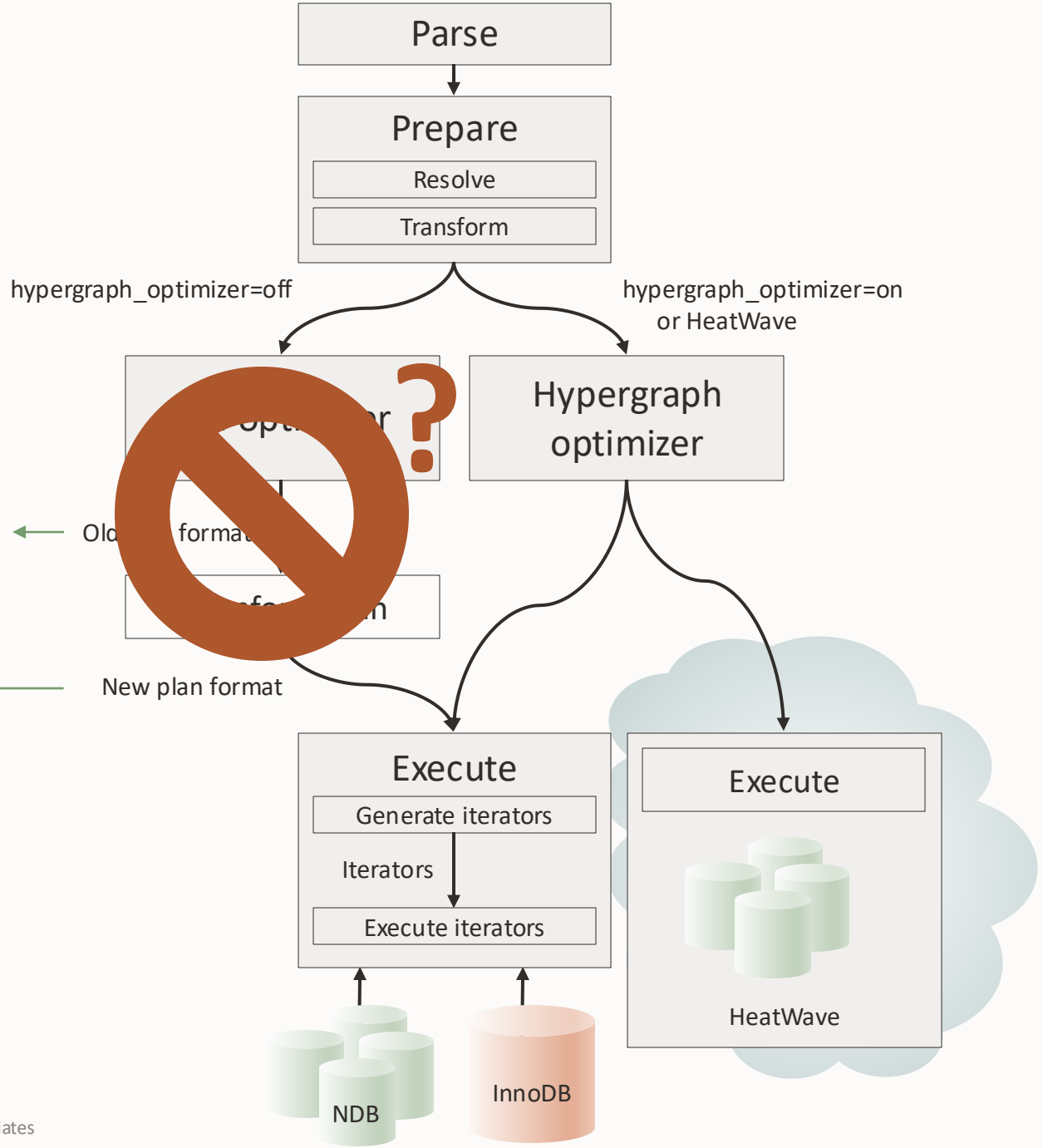
FORMAT=JSON (v2)
FORMAT=TREE



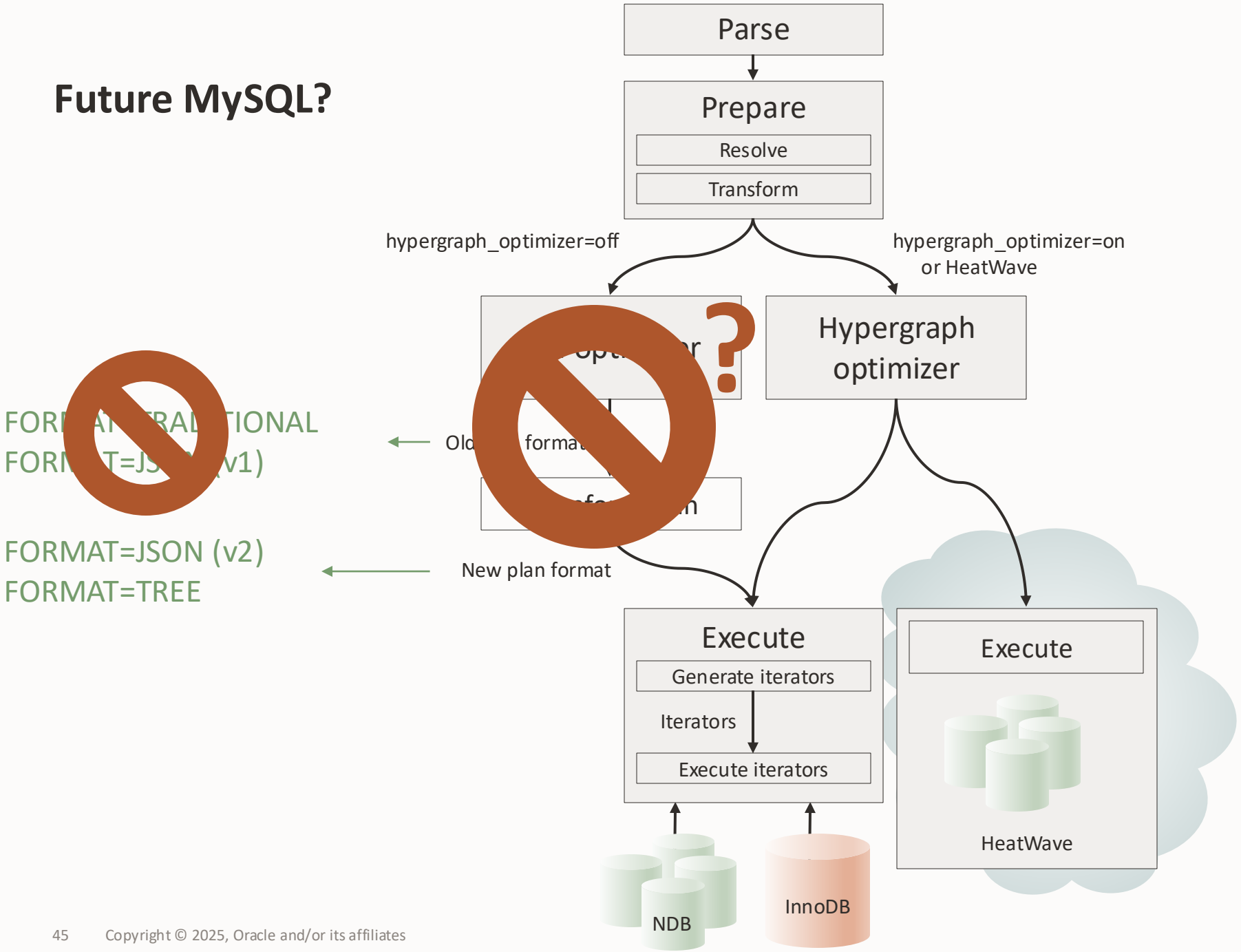
Future MySQL?

FORMAT=TRADITIONAL
FORMAT=JSON (v1)

FORMAT=JSON (v2)
FORMAT=TREE



Future MySQL?



Predicting the future

- Remove the old optimizer?
  - FORMAT=TRADITIONAL and version 1 of FORMAT=JSON will stop working
  - No plans to remove TRADITIONAL or JSON format version 1 while the old optimizer exists
- Add INTO @*var* FOR CONNECTION?
  - There's no reason not to
  - Features should be orthogonal if possible
- Add more attributes to version 2 of FORMAT=JSON?
  - Expected in innovation releases
  - Less likely in LTS releases
- Change the JSON format to version 3?
  - Doesn't look very likely in the foreseeable future
- Other formats? (Do we need more?)
- JSON Schema for EXPLAIN output?
- Automatic EXPLAIN in performance_schema for all running queries?

Preparing for the future

- Start using FORMAT=TREE for human readable output
  - FORMAT=TRADITIONAL only works with the old optimizer
  - TREE is often easier to read
  - TREE matches the execution better
  - TREE works with ANALYZE
- Start using FORMAT=JSON with explain_json_format_version=2 for machine readable output
  - Version 1 only works with the old optimizer
  - Version 2 matches the execution better
  - Version 2 works with ANALYZE
- Check json_schema_version
  - New attributes are being added (minor number bump)
  - Incompatible changes are much less likely (major number bump)

Thank you


ORACLE

Our mission is to help people see data in new ways, discover insights, unlock endless possibilities.