Database Applications
Lecture 3: Query Optimisation

Santha Sumanasekara

July 2023

RMIT UNIVERSITY

1

1

Database Query Optimisation

- Overview
- Lifecycle of an SQL Query and how they are processed
- Query Optimiser – in detail
- What can you do to help with query optimiser?

RMIT UNIVERSITY

2

2

The Lifecycle of an SQL Query

And, how they are processed!

RMIT UNIVERSITY

3

The life-cycle of an SQL query

Parse → Optimise → Execute

RMIT UNIVERSITY

4

4

The life-cycle of an SQL query

The Parser makes sure that the query is syntactically correct (e.g. SQL grammar) and semantically correct (e.g. tables and attributes exist), and returns errors if not. If all OK, it sends the "**parsed Query**" to the optimiser.

RMIT UNIVERSITY

5

5

The life-cycle of an SQL query

The query planner and optimiser considers a number of different "query plans" which may have different optimisations, estimates the cost (mainly disk I/O) of each query plan based on various factors, then it picks the optimal plan and makes into an **execution plan**.

RMIT UNIVERSITY

6

6

The life-cycle of an SQL query

The **query executor** takes the optimal plan and turns it into operations for the database, returning the results back to us.

RMIT UNIVERSITY

7

The life-cycle of an SQL query

Today we focus on query evaluation and optimisation.

RMIT UNIVERSITY

8

Query Optimiser

We look at in detail how the query optimiser does its magic.

9

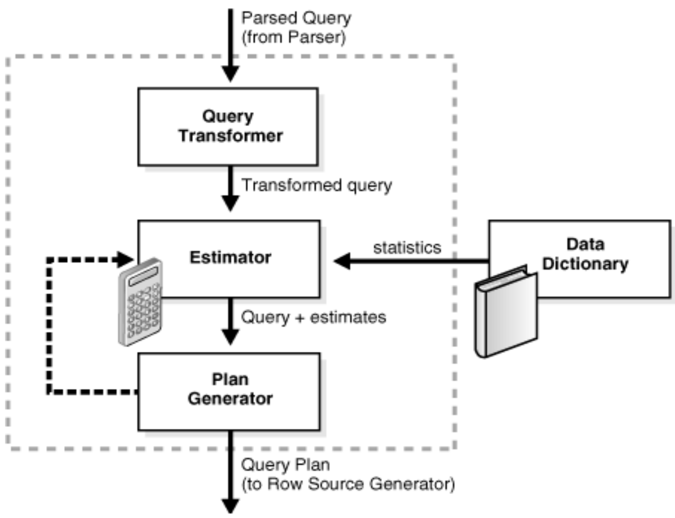
Query Optimiser

- The optimizer attempts to generate the most optimal execution plan for a SQL statement.
- The optimizer choose the plan with the lowest cost among all considered candidate plans.
- The optimizer uses available statistics to calculate cost.
- For a specific query in a given environment, the cost computation accounts for factors of query execution such as I/O, CPU, and communication.

10

10

Query Optimiser



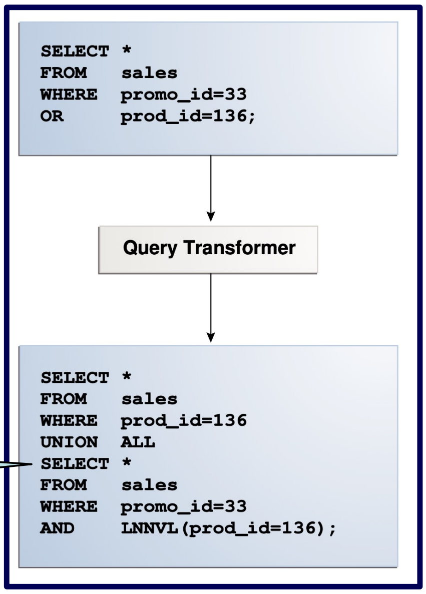
RMIT UNIVERSITY

11

11

Query Optimiser – Rewrite phase

- In this phase, the optimiser determines whether it is advantageous to rewrite the original SQL statement into a semantically equivalent SQL statement with a lower cost.
- Sometimes, the seemingly simple SQL statement may not necessarily the most-efficient.
- Replacing a statement with OR condition with two statements joined with UNION can be considered more efficient.



This is not the most-likely style any SQL programmer would write. Apparently, it is semantically similar to the original, and produce the result quicker.

12

12

Query Optimiser – Estimator

- The estimator is the component of the optimiser that determines the overall cost of a given execution plan.

- The estimator uses three different measures to determine cost:
  - Selectivity – The percentage of rows in the row set that the query selects;
  - Cardinality – The cardinality is the number of rows returned by each operation in an execution plan.
  - Cost – This measure represents units of work or resource used. The query optimizer uses disk I/O, CPU usage, and memory usage as units of work.

RMIT UNIVERSITY

13

13

Estimator -- Selectivity

- An example:

```
SELECT mvtitle
    FROM movie
    WHERE yrmde >= 1970
```

```
SELECT mvtitle
    FROM movie
    WHERE yrmde <= 1970
```

- In first example, the selectivity is very low, so, the optimiser may decide to use a full table scan (i.e. read the full table row-by-row and pick the matching rows).
- In second example, the selectivity is high, so, it may decide (if it exists) an index on yrmde column and pick the matching rows.

RMIT UNIVERSITY

14

14

Query Optimiser – Cardinality

- Consider the following SQL query:

```
SELECT mvtitle
    FROM movie m JOIN director d ON m.dirnumb = d.dirnumb
    WHERE d.dirname = 'Allen, Woody'
```

- You would expect that two tables are joined first, and then apply the WHERE condition to filter rows.
- However, considering cardinality at each stage (and also, we do not need both tables to execute the condition), the optimiser in its execution plan, may put WEERE condition ahead of the JOIN.

RMIT UNIVERSITY

15

15

Query Optimiser – Cardinality

- Oracle example

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			3	6
MERGE JOIN			3	6
TABLE ACCESS	DIRECTOR	BY INDEX ROWID	1	2
Filter Predicates				
D.DIRNAME='Allen, Woody'				
INDEX	SYS_C00655649	FULL SCAN	8	1
SORT		JOIN	24	4
Access Predicates				
M.DIRNUMB=D.DIRNUMB				
Filter Predicates				
M.DIRNUMB=D.DIRNUMB				
TABLE ACCESS	MOVIE	FULL	24	3

RMIT UNIVERSITY

16

16

Query Optimiser – Cardinality

- PostgreSQL example

Data Output | Messages | Explain ✕ | Notifications

Graphical | Analysis | Statistics

#	Node	Rows Actual	Loops
1.	→ Hash Inner Join (rows=3 loops=1) **Hash Cond**: (m.dirnumb = d.dirnumb)	3	1
2.	→ Seq Scan on movie as m (rows=24 loops=1)	24	1
3.	→ Hash (rows=1 loops=1) **Buckets**: 1024 **Batches**: 1 **Memory Usage**: 9 kB	1	1
4.	→ Seq Scan on director as d (rows=1 loops=1) **Filter**: (dirname = 'Allen, Woody'::bpchar) **Rows Removed by Filter**: 7	1	1

RMIT UNIVERSITY

17

17

Query Optimiser – Cost

- To estimate cost, the optimizer considers factors such as the following:
  - System resources, which includes estimated I/O, CPU, and memory
  - Estimated number of rows returned (cardinality)
  - Size of the initial data sets
  - Distribution of the data
  - Access structures

RMIT UNIVERSITY

18

18

Query Optimiser – Cost

- The plan generator explores various plans by trying out different access paths, join methods, and join orders.
- Many plans are possible because of the various combinations that the database can use to produce the same result. The optimizer picks the plan with the lowest cost.

RMIT UNIVERSITY

19

19

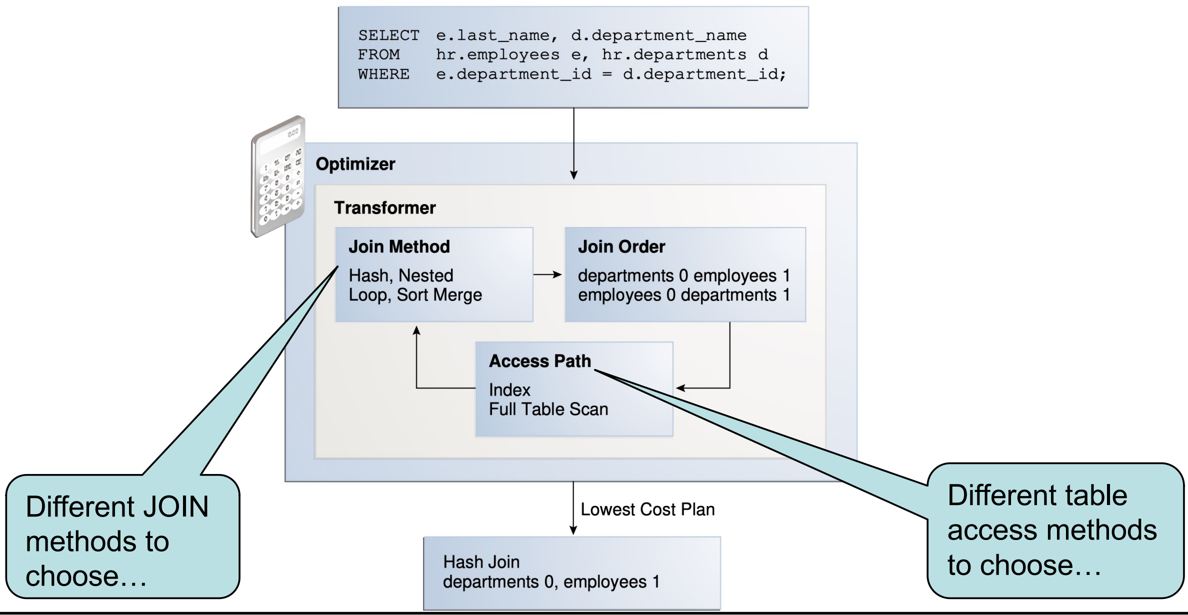
Plan Generator

We look at in detail how the query optimiser does its magic.

RMIT UNIVERSITY

20

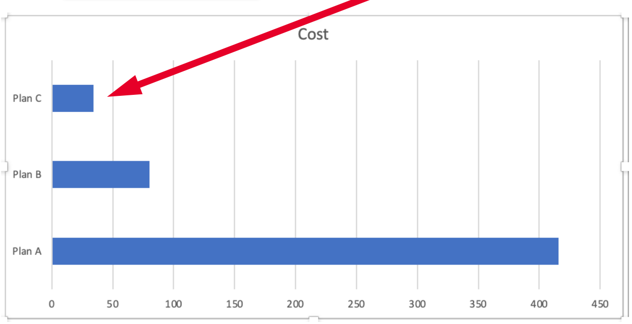
Query Optimiser – Plan Generation



21

Query Optimiser – Plan Generation

- Considering all of the above metrics, a number of potential execution plans are generated and each execution plan gets assigned an execution cost. Then, optimiser chooses, the plan with the cheapest execution cost, and passed on that execution plan to the next step (execution stage).



RMIT UNIVERSITY

22

22

Execution Plan – Example

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			3	6
MERGE JOIN			3	6
TABLE ACCESS	DIRECTOR	BY INDEX ROWID	1	2
Filter Predicates				
D.DIRNAME='Allen, Woody'				
INDEX	SYS_C00655649	FULL SCAN	8	1
SORT		JOIN	24	4
Access Predicates				
M.DIRNUMB=D.DIRNUMB				
Filter Predicates				
M.DIRNUMB=D.DIRNUMB				
TABLE ACCESS	MOVIE	FULL	24	3

RMIT UNIVERSITY

23

23

Query Optimiser optimise different parts of a query

Some internal workings of the optimiser

RMIT UNIVERSITY

24

WHERE Condition

Some internal workings of the optimiser

25

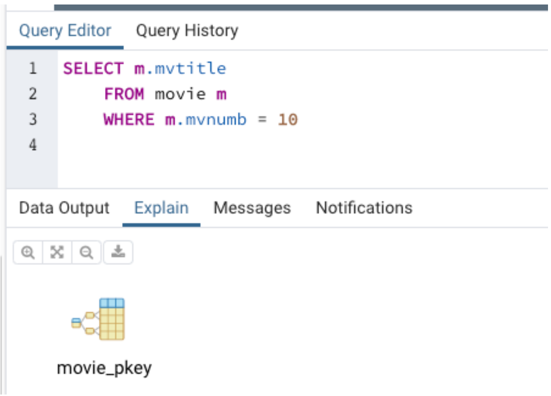
WHERE condition

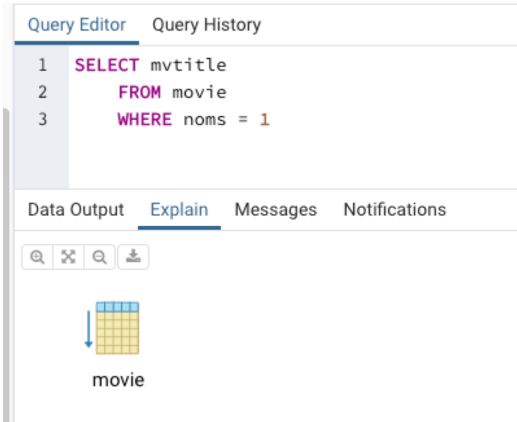
- Query planner/optimiser looks at all access paths available for a given attribute
  - Full Table Scan
  - Index Scan
  - Index-only Scan (depends on what's on other parts of the query)
- For WHERE conditions:
  - If select predicate is an equality test and an index is available for that attribute, can use an index scan
  - Can also use index scan for comparison/range tests if an ordered index (such as B+ Tree) is available for the attribute
- For more complicated tests, or if no index is available for attributes being used:
  - Use Full Table Scan.

26

26

WHERE condition





RMIT UNIVERSITY

27

27

Sorting – ORDER BY clause

Some internal workings of the optimiser

RMIT UNIVERSITY

28

Sorting – ORDER BY clause

- Sorting is expensive
  - Tables being sorted may be much larger than memory!
- For tables that fit in memory, traditional sorting techniques are used (e.g. quick-sort)
- For tables that are larger than memory, must use an external-memory sorting technique
  - Table is divided into runs to be sorted in memory
  - Each run is sorted, then written to a temporary file
  - All runs are merged using an N-way merge sort
- In general, sorting should be applied as late as possible.

RMIT

https://www.youtube.com/watch?v=ATK74YSzwxg.

29

JOINs

Some internal workings of the optimiser

RMIT UNIVERSITY

30

JOIN Operations

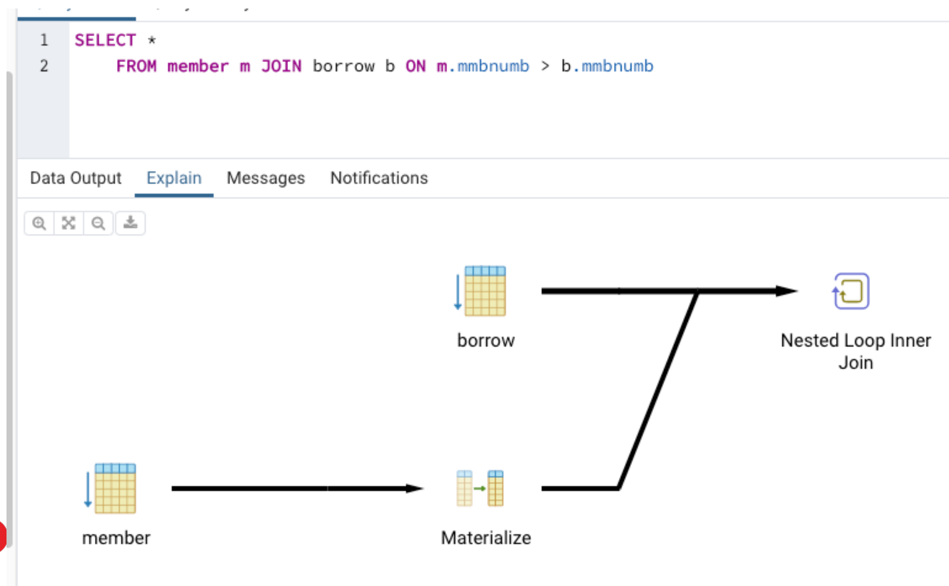
- JOINs are very common in SQL.
- Could also potentially be a very costly operation!
- A simple JOIN method is Nested-loop Join.
- This is one of the least-efficient join algorithms.
- Involves the designation of one table as the driving table (also called the **outer table**) in the join loop. The other table in the join is called the **inner table**.

```
for each row ro in outer_table do begin
    for each row ri in inner_table do begin
        if ro and ri satisfy JOIN condition then
            add ro · ri to result
    end
end
```

31

31

JOIN Operations – Nested-loop Join



32

32

JOIN Operations – Hash Join

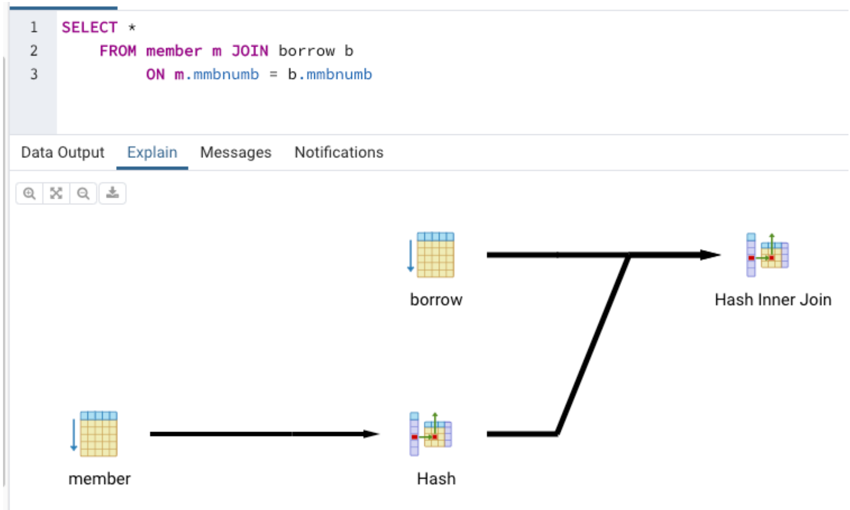
- This method works with equijoins
  - E.g. `WHERE movie m JOIN director d`
    `ON m.dirnumb = d.dirnumb`

- When you join two tables, query optimiser uses the smaller table to build a hash table on the join key.
- Use same hash function on both tables *t1* and *t2*, of course
- Partitions (say t11, t12, ... And t21, t22, ...) are saved to disk as they are generated
- Rows in partition t11 will be joined in rows in t21, etc.

- Very fast and efficient equijoin strategy

RMIT UNIVERSITY

JOIN Operations – Hash Join



RMIT UNIVERSITY

JOIN Operations – Sort-Merge Join

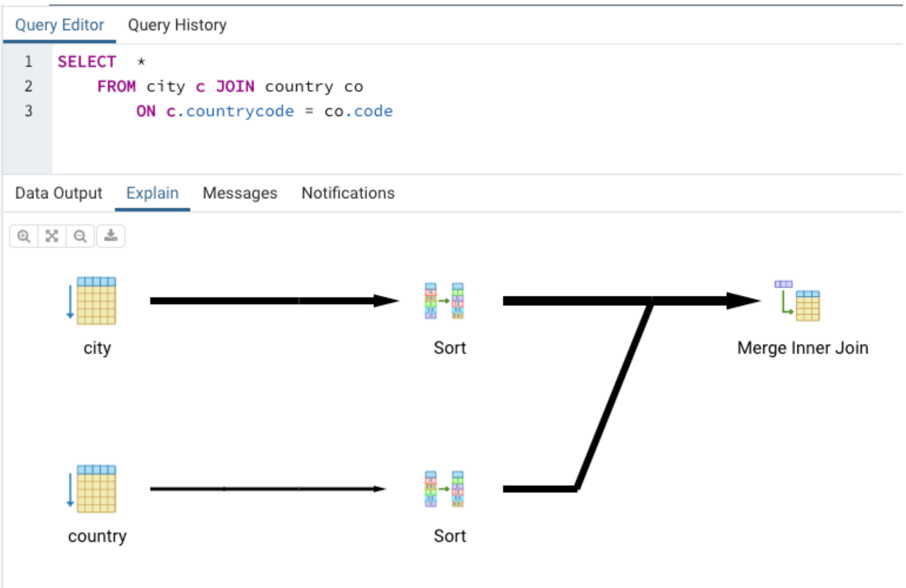
- If tables are already ordered by join attributes, can use a merge-sort technique
- Much better performance than nested-loop join
  - Dramatically reduces disk accesses
  - Unfortunately, relations aren't usually ordered
- Can also enhance sort-merge joins when at least one relation has an index on the join attributes
  - e.g. one relation is sorted, and the unsorted relation has an index on the join attributes
- When the query optimiser thinks the hash table needed for a hash join will exceed memory, it will typically use a sort-merge join instead.

RMIT UNIVERSITY

35

35

JOIN Operations – Sort-Merge Join



36

36

What can you do to help with query optimiser?

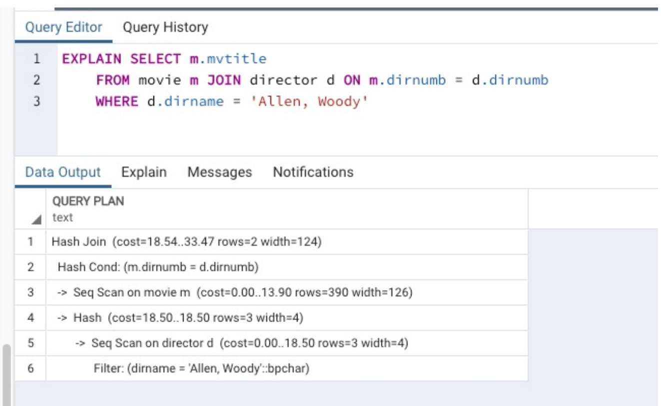
Help your query optimiser, so it can help you run your queries efficiently.

RMIT UNIVERSITY

37

Optimising Query Performance

- To improve query performance, you must know how the database actually runs your query
- Use “EXPLAIN statement”, before running the query.
  - Runs planner and optimizer on your query, then outputs the plan and corresponding cost estimates.



RMIT UNIVE

38

Optimising Query Performance

- Using this information, you can:
  - Create indexes on tables, where appropriate
  - Identify bottlenecks – the steps taking most of the time and find alternative ways/ re-write the query (use of sub-queries, nested queries, etc)
  - Use optimiser hints, if that's available.

```
PostgreSQL:
SET enable_hashjoin = off;
```

RMIT UNIVERSITY

Cost Estimation

- Query planner/optimizer must make *estimates* about the cost of each stage

- Database maintains statistics for each table, to facilitate planning and optimization

- Different levels of detail:
  - Some DBs only track min/max/count of values in each column. Estimates are very basic.
  - Some DBs generate and store histograms of values in important columns. Estimates are much more accurate.

RMIT UNIVERSITY

Cost Estimation

- Recall last week, two very similar queries – one used the index, the other didn't.
- Query optimiser had statistics on spread of data
- Databases also frequently provide a command to compute table statistics.
- Make sure statistics are up-to-date, so that planner has best chance of generating a good plan

PostgreSQL:
VACUUM ANALYZE;
To update statistics on all tables in database
VACUUM ANALYZE tablename;
To update statistics on a specific table

RMIT UNIVERSITY

34

41

Demo Time!

Let's try out a query.

World Database:
Find the other languages spoken in the countries where "Swahili" is spoken.
This requires a self-join.

RMIT UNIVERSITY

42

Demo Time!

Let's try out a query.

```
EXPLAIN SELECT cl2.countrycode, cl2.language
    FROM countrylanguage cl1 JOIN countryLanguage cl2
                ON cl1.CountryCode = cl2.countrycode
    WHERE cl1.language = 'Swahili' AND
                  cl2.language <> 'Swahili'
```

RMIT UNIVERSITY

43

Demo Time!

Let's try out a query.

World Database:
Find the other countries that has common languages as in South Africa.

RMIT UNIVERSITY

44

Demo Time!

```
EXPLAIN SELECT c2.name, cl2.language
    FROM ((country c1 JOIN countrylanguage cl1
                ON c1.code = cl1.countrycode)
                JOIN countrylanguage cl2
                          ON cl1.language = cl2.language)
                         JOIN country c2
                                   ON cl2.countrycode = c2.code
    WHERE c1.name ='South Africa' AND
                 c2.name <>'South Africa'
```

45

Summary

- Discussed general details of how most databases evaluate SQL queries
- Some SQL queries have several ways to execute them
- Query planner/ optimiser evaluates the cost of each query plan and chooses the optimal query plan and builds an execution plan on it.
- Database maintains statistics for each table, to facilitate planning and optimization, such as choice of access methods, join methods, etc
- Make sure statistics are up-to-date, so that planner has best chance of generating a good plan

RMIT UNIVERSITY

46

46

Menti-time!

Menti.com
Code: 10 84 112

RMIT UNIVERSITY

47

Questions?

RMIT UNIVERSITY

48