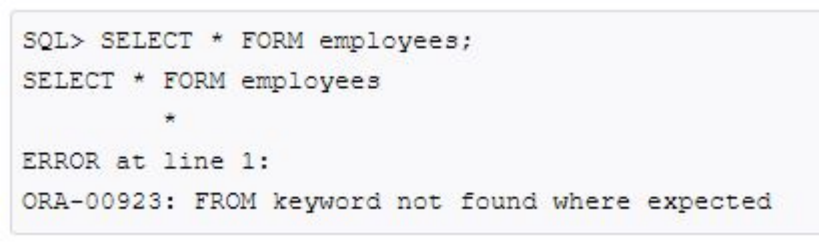
SQL Queries

Leon Lam, Rachel Yang

How is a query processed?

It goes through **pre-optimization** (parsing, binding, other checks), **optimization** (creating and comparing query plans), and finally **execution**.

Pre-optimization

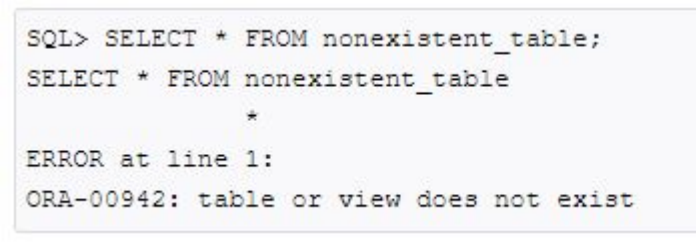


```
SQL> SELECT * FORM employees;
SELECT * FORM employees
         *
ERROR at line 1:
ORA-00923: FROM keyword not found where expected
```

Parsing

- Is the query written correctly?
- Checks syntax:
  - Spelling words wrong
  - Misusing reserved words
  - Forgetting semicolons
- Outputs:
  - Parse / sequence tree with logical execution steps

Pre-optimization



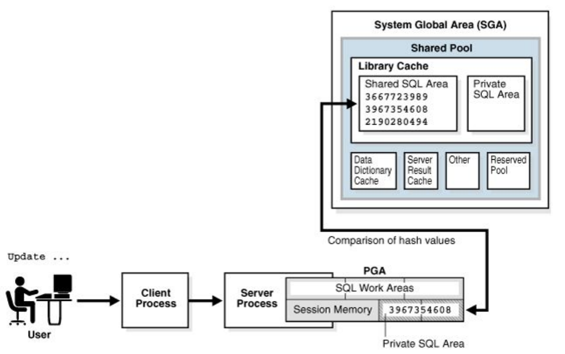
```
SQL> SELECT * FROM nonexistent_table;
SELECT * FROM nonexistent_table
              *
ERROR at line 1:
ORA-00942: table or view does not exist
```

Binding - performed by algebrizer

- Is the query statement meaningful — are the objects valid and referenced correctly?
- Checks semantics:
  - Existence of objects
- Outputs:
  - Query processor tree

Pre-optimization

Figure 3-2 Shared Pool Check

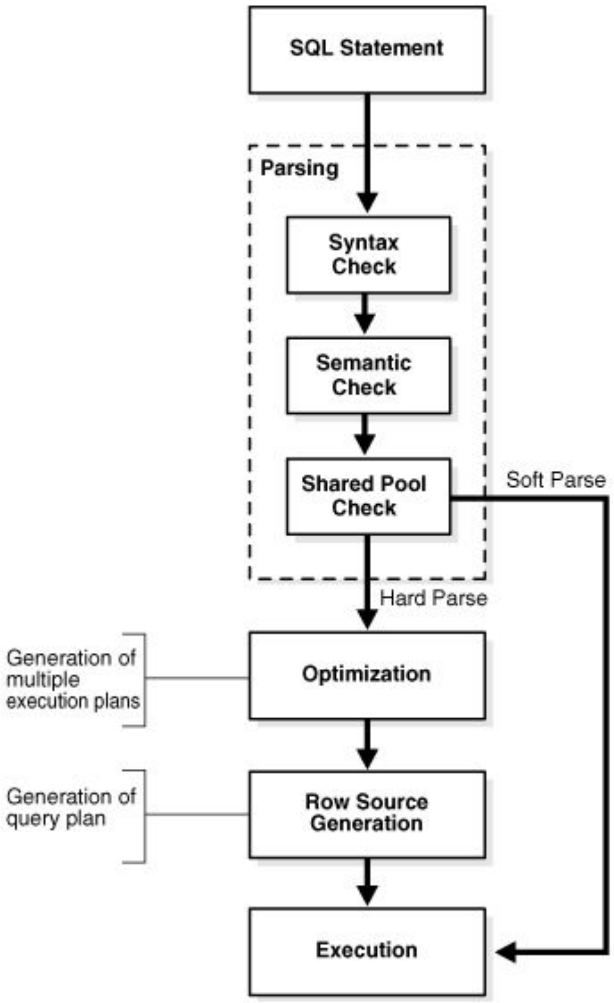


Other Checks - i.e. Shared Pool Checks

- Can the DB skip resource-intensive processing steps?
- Checks:
  - Has this query been encountered before? *By using hashing algorithm*
- Outputs:
  - Result of either soft or hard parse

Shared Pool Checks (expanded)

Figure 3-1 Stages of SQL Processing



What is query optimization?

Finding the **most efficient** way to execute a given query for a given database, by comparing possible query execution plans.

most efficient = least costly

fastest
execution time

# of steps
required

and
more

How is query optimization done?

There’s a few pre- and post-optimization processes, but ultimately it **looks at all (or most) possible orders** of operations, **builds trees** for each execution plan, and **compares these trees by cost**.

Optimization

Create execution plans

- How does the software know what execution plan gives us the desired result?
- Compute different permutations of operations
  - Different types of access paths
  - Different relational table join techniques / join order
  - Reconsidering indexes
- Output: Node tree where each node is a separate operation required to execute a query
  - Flows from bottom to top — child nodes output to parent nodes

Optimization

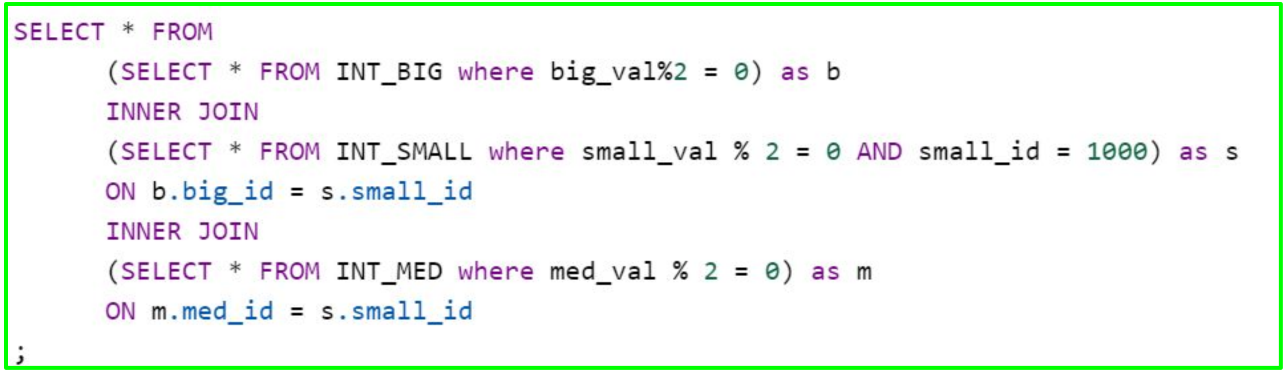
Compare execution plans

- How to compare plans across multiple cost metrics (execution time, monetary cost)?
- In each plan, each step is assigned an estimated cost value
    - Cardinality flowing through each edge in a query plan affects number of operations
    - These cardinality estimates depend on estimates of the selection factor of predicates in the query
    - * If predicates are combined, the accuracy of estimation decreases because of high correlation between predicates
- Each plans' total estimated cost is the sum of its steps' estimated costs
- Creating and comparing plans is also costly

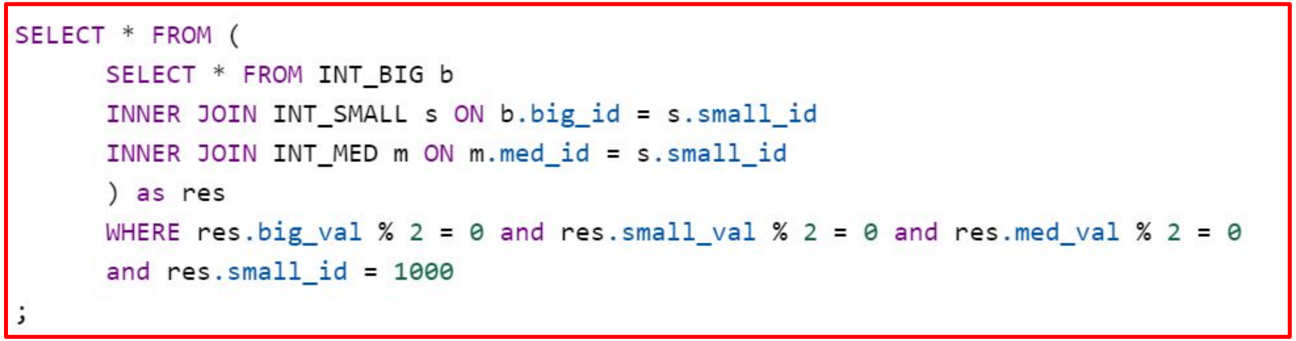
Why is query optimization important?

Although they arrive at the same result, an inefficient query can be many times **more expensive** than an efficient one.

Example queries

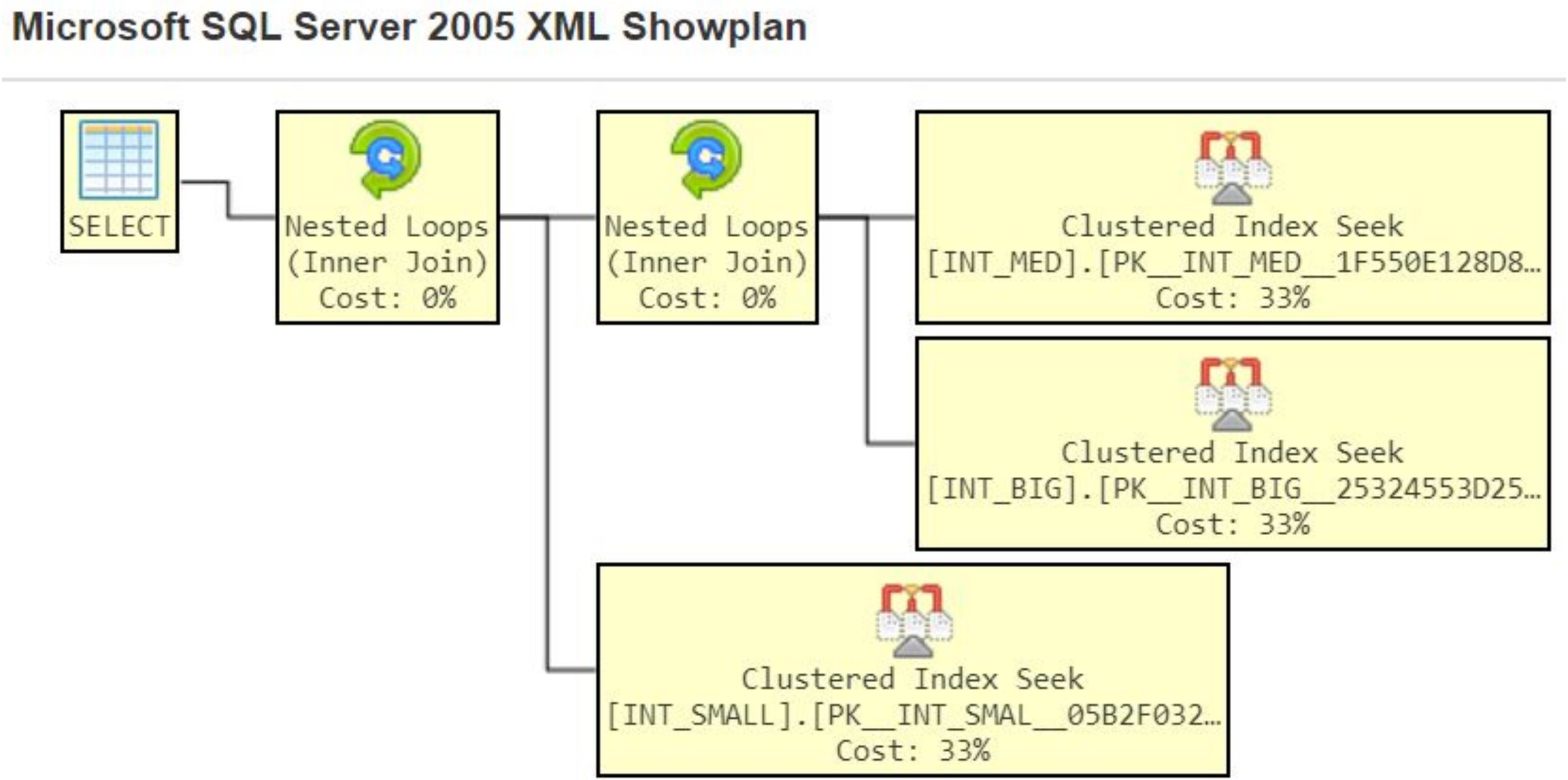


```
SELECT * FROM
      (SELECT * FROM INT_BIG where big_val%2 = 0) as b
      INNER JOIN
      (SELECT * FROM INT_SMALL where small_val % 2 = 0 AND small_id = 1000) as s
      ON b.big_id = s.small_id
      INNER JOIN
      (SELECT * FROM INT_MED where med_val % 2 = 0) as m
      ON m.med_id = s.small_id
;
```



```
SELECT * FROM (
      SELECT * FROM INT_BIG b
      INNER JOIN INT_SMALL s ON b.big_id = s.small_id
      INNER JOIN INT_MED m ON m.med_id = s.small_id
      ) as res
      WHERE res.big_val % 2 = 0 and res.small_val % 2 = 0 and res.med_val % 2 = 0
      and res.small_id = 1000
;
```

- **Green** was written to filter the individual tables before joining
- **Red** was written to join first and then filter afterward
- If parsed sequentially, we would expect **Green** to be much faster than **Red**



Results: Both queries were optimized to same query plan

How does the exact optimization math work?

We have no real idea. **Each database is different** in terms of what the data looks like and which cost metric is more important.

Post-optimization

Plan Storage

- Once a plan is selected as “best,” it’s stored in a plan cache in memory
    - UNLESS an identical plan already exists in the cache
- These plans are accessed by SQL Server and then executed