

M146 Database Systems Spring 2021

Georgia Koutrika



About me

Georgia Koutrika

- PhD (di.uoa.gr)
- Stanford, IBM, HP Labs, ...
- Today: Research Director at ATHENA Research Center



Research interests

in the intersection of databases, information retrieval, machine learning:

- data exploration, including natural language interfaces,
- recommendation systems,
- big data analytics
- large-scale information extraction, entity resolution and information integration

What about you?

Logistics

When: Wednesdays 5 – 8

Gotomeeting URL: <https://global.gotomeeting.com/join/337207853>

LECTURES

- Instructor lectures
- **DBTalks**
- Student presentations

Logistics

PROJECTS

- First Project: Paper + Oral presentation
- Second Project: Report on a new idea on the paper you read
- Final Project: Implementation of your idea and comparison to the paper

Logistics

First Project: 1 paper + Oral presentation

Sources: <https://dblp.uni-trier.de/>

Conference: EDBT, SIGMOD, PVLDB (2021, 2020),

PURPOSE:

The purpose of this assignment is to pick one (1) paper, study it in depth, and be in position to give a detailed lecture and answering any questions.

Logistics

First Project: 1 paper + Oral presentation

Example Topics: database internals, data exploration, recommendations, ...

- *Explaining Differences Between Unaligned Table Snapshots*
- *Dynamic Query Refinement for Interactive Data Exploration*
- *Improved Cardinality Estimation by Learning Queries Containment Rates*
- *Optimal Histograms with Outliers*
- *Distributed Similarity Joins over Top-K Rankings*
- *Cost Models for Big Data Query Processing: Learning, Retrofitting, and Our Findings*
- *Learning a Partitioning Advisor for Cloud Databases*
- *Qd-tree: Learning Data Layouts for Big Data Analytics*
- ...
- *Duoquest: A Dual-Specification System for Expressive SQL Queries*
- *DeepDB: Learn from Data, not from Queries*
- *Data-Driven Domain Discovery for Structured Datasets*

Logistics

First Project: 1 paper + Oral presentation

DELIVERABLE: A presentation (slides) that make a thorough and in-depth explanation of the paper. Points that you should (at least) cover (depending on the paper some may not be applicable):

- What is the problem studied in the paper? Why is it important?
- What are the specific challenges to address?
- What are the contributions of this work?
- What has been done in the past (related works)
- Problem Formulation, Concepts, Critical Definitions
- System architecture
- Algorithms : What they do, how they work, provide the basic intuition as well as a more detailed explanation
- Experiments: datasets used, what was measured and how, what did the experiments showed, etc
- Evaluation of the work (as you see it): advantages, disadvantages, possible improvements for the solution, what is missing from the experiments, what we learned from this work.

THERE IS NO MIN OR MAX NUMBER OF SLIDES: you should be able to cover everything in sufficient detail. Copying/pasting blocks of text from the paper will not work. Using slides found on the web will not do.

Logistics

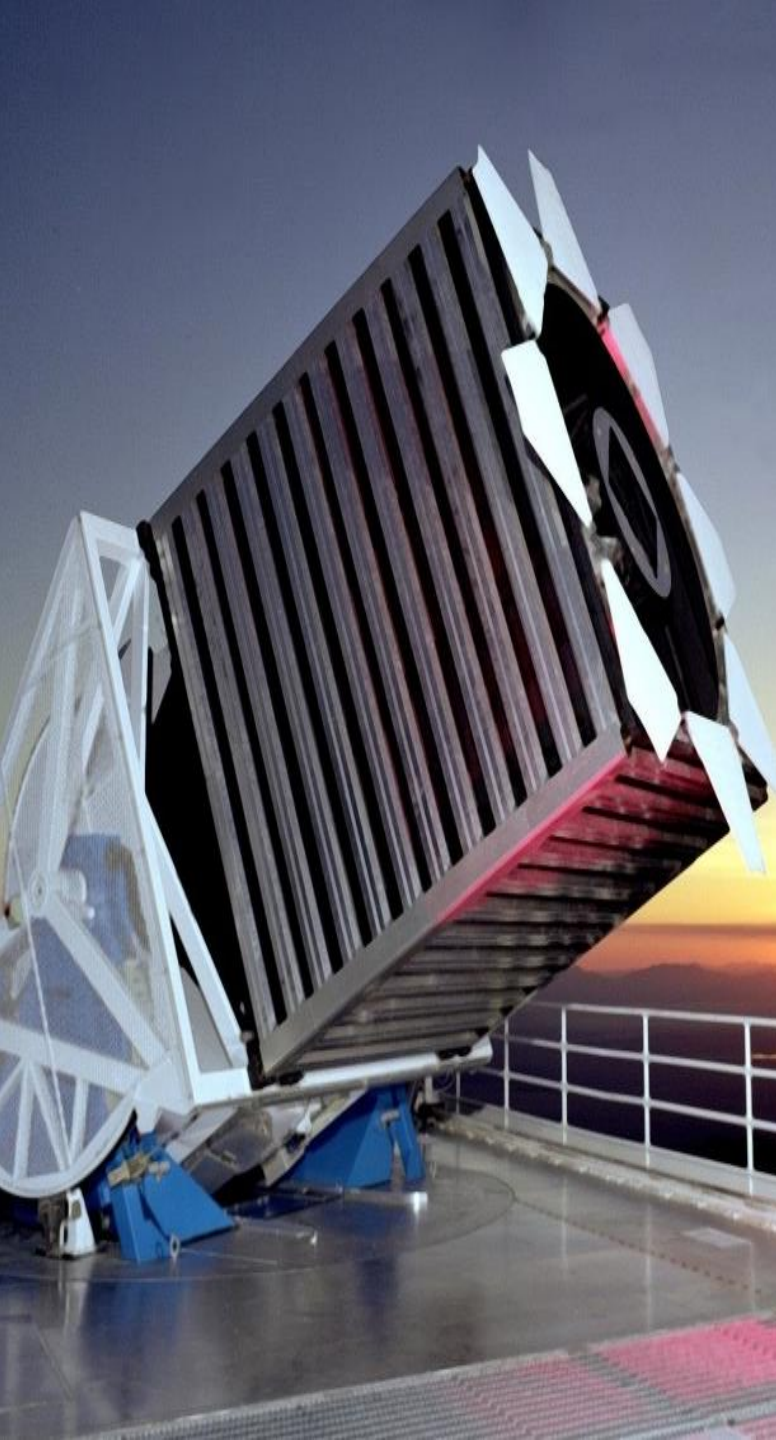
First Project: 1 paper + Oral presentation

DEADLINE: March 30, 2021. Send it through e-class.

PRESENTATIONS: We will schedule presentations after that date (date to be determined).

About Database Systems





How Much Data Do We Create? (1)

Sloan Digital Sky Survey or SDSS

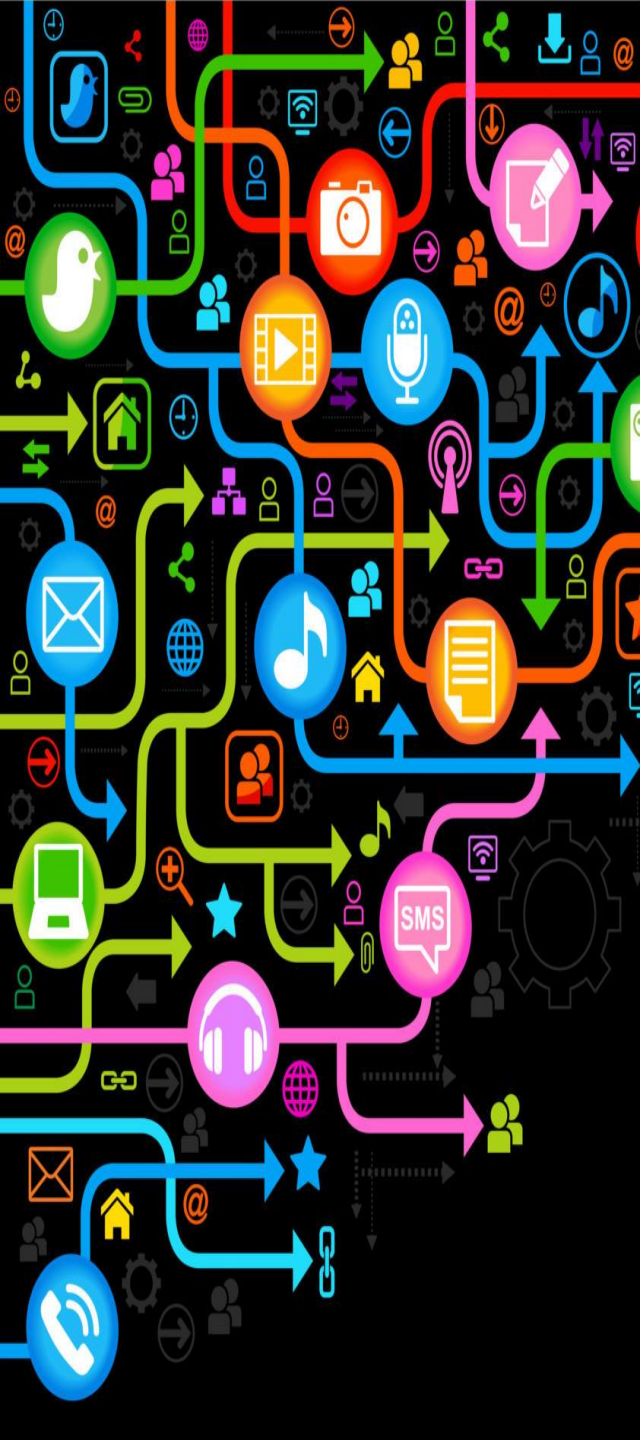
- SDSS is a major multi-spectral imaging and spectroscopic redshift survey using a dedicated 2.5-m wide-angle optical telescope at Apache Point Observatory in New Mexico, United States.
- Data collection began in 2000.
- Every night the telescope produces > 200 GB of data



How Much Data Do We Create? (2)

VISA

- Nearly 15,000 banks, processors and other third parties connect to Visa's network
- **Q4 2016 (US): 9.96 million transactions**
- The company's flagship data center, dubbed Operations Center East, or OCE, is a 140,000-square-foot facility.
- 376 servers, 277 switches, 85 routers and 42 firewalls--all connected by 3,000 miles of cable



How Much Data Do We Create? (3)

Social Data

- More than 3.7 billion humans use the internet
- More than 300 million photos get uploaded per day on Facebook
- 510,000 comments posted and 293,000 statuses updated per minute
- 2.5 quintillion bytes of data created each day

The world is increasingly driven by data...

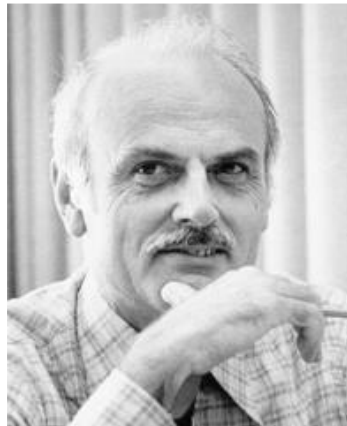
DBMSs help us organize, store, index, and query efficiently our data

Some definitions

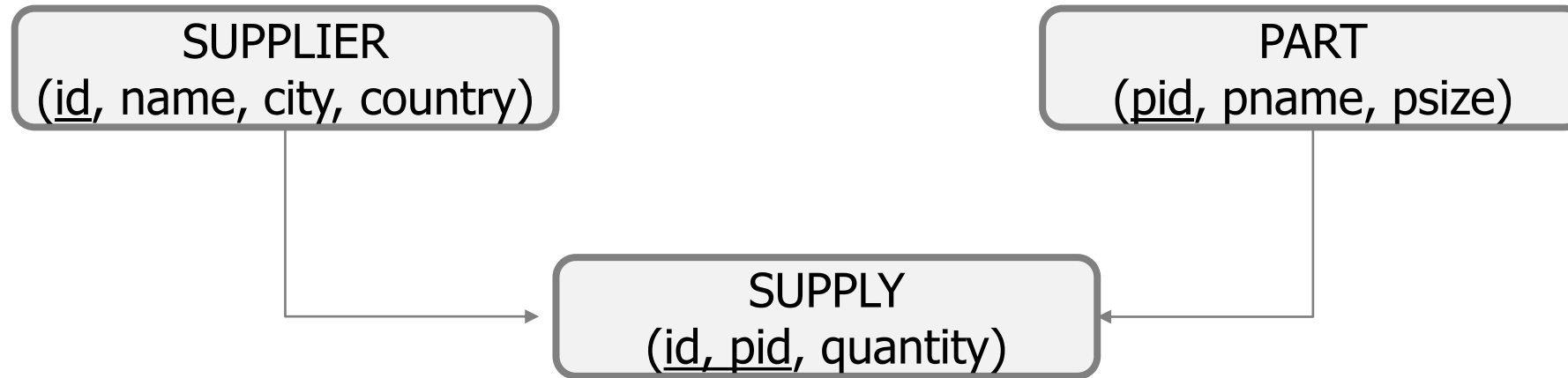
- A **database** is an organized collection of structured information, or data, stored in a computer system.
- A database is usually controlled by a **database management system (DBMS)**.
- Together, the data and the DBMS are referred to as a database system, often shortened to just **database**.

History of RDBMS

- 1970: The term "relational database" was invented by E. F. Codd at IBM
 - Present the data to the user as relations (a presentation in tabular form, i.e. as a collection of tables with each table consisting of a set of rows and columns);
 - Provide relational operators to manipulate the data in tabular form.



Relational Data Model



History of RDBMS

- **1974: IBM System R**
 - A key development of the System R project was SQL.
- **1979: Oracle**, released in 1979 by Relational Software
- Other examples: DB2, SAP Sybase ASE, and Informix



Jim Gray




Michael Stonebraker



Larry Ellison

Today ...




MORE ACM AWARDS

A.M. TURING AWARD

A.M. TURING AWARD LAUREATES BY...

ALPHABETICAL LISTING YEAR OF THE AWARD



MICHAEL STONEBRAKER
United States – 2014

CITATION
For fundamental contributions to the design and implementation of modern database systems.

BIRTH:

SHORT ANNOTATED BIBLIOGRAPHY **ACM TURING AWARD LECTURE VIDEO**



MONEY INC

What are you looking for?

BUSINESS ▼ PERSONAL FINANCE ▼ REAL ESTATE ▼ TRAVEL ▼

Home » Business » How Larry Ellison Achieved a Net Worth of \$54.5 Billion

How Larry Ellison Achieved a Net Worth of \$54.5 Billion

Nat Berman 1 Year Ago





MORE ACM AWARDS

A.M. TURING AWARD

A.M. TURING AWARD LAUREATES BY...

ALPHABETICAL LISTING YEAR OF THE AWARD RESEARCH SUBJECT



JAMES ("JIM") NICHOLAS GRAY 

United States – 1998

CITATION
For seminal contributions to database and transaction processing research and technical leadership in system implementation.

SHORT ANNOTATED BIBLIOGRAPHY **ACM TURING AWARD LECTURE** **RESEARCH SUBJECTS** **ADDITIONAL MATERIALS**

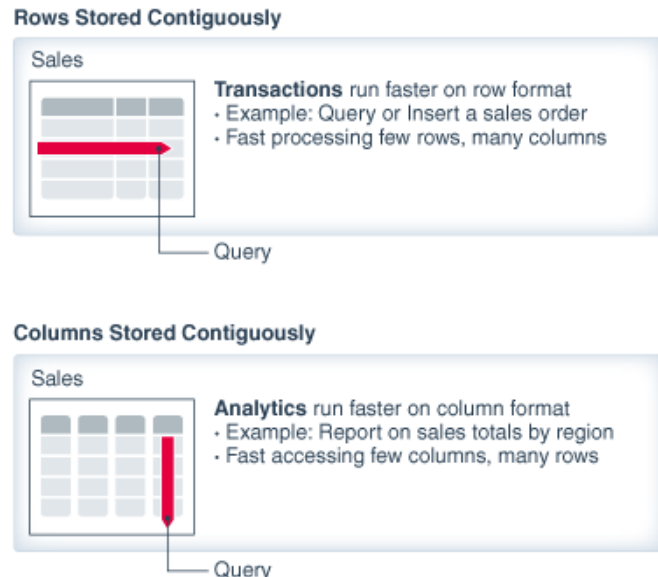
➔ As a guide for those interested in learning more about Gray and his work, links have been provided below to relevant articles from the [proceedings](#) of the Tribute held for him on May 31, 2008 at the University of California, Berkeley.



History of RDBMS

- 2000s:
 - Rise of the special purpose OLAP DBMSs.
(Online analytical processing tools enable users to analyze multidimensional data interactively from multiple perspectives.)
 - **Columnar Stores (vs Row Stores)** ([what is the difference?](#))

Columnar and Row-Based Storage

















History of RDBMS

- 2000s: NoSQL SYSTEMS

- A NoSQL, or nonrelational database, allows **unstructured and semi-structured data** to be stored and manipulated (in contrast to a relational database, which defines how all data inserted into the database must be composed).
- NoSQL databases grew popular as web applications became more common and more complex.

- Schemaless
- Non relational data models (document, key/value, etc)
- No ACID transactions
- Custom APIs instead of SQL
- Focus on high availability & high scalability

Document Database	Graph Databases
  	 
Wide Column Stores	Key-Value Databases
   	    

History of RDBMS

- 2010s: NewSQL

NewSQL is a class of relational database management systems that provide same performance for OLTP workloads as NoSQL DBMSs without giving up ACID:

- Relational / SQL
- Distributed



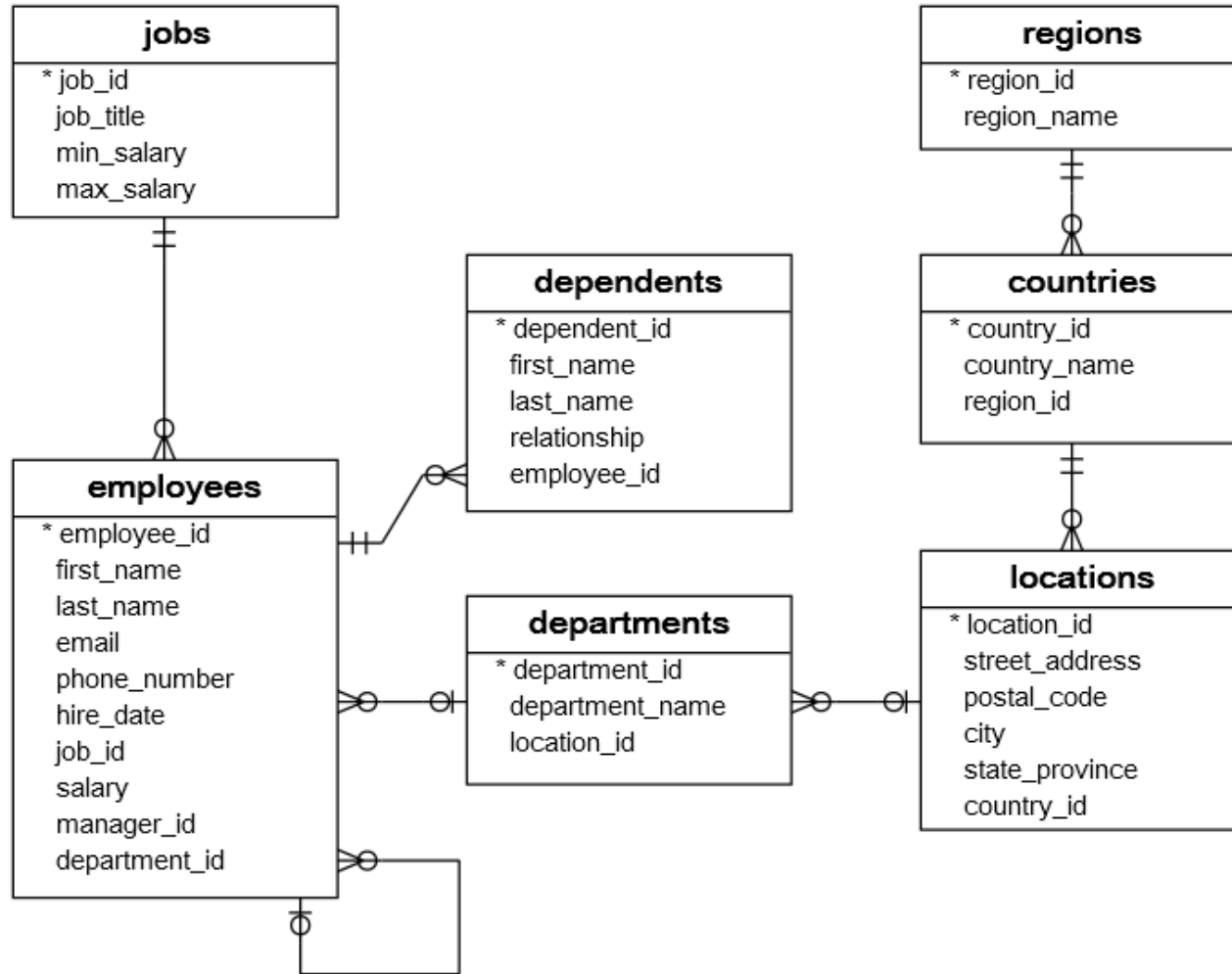
So why do I need to know about RDBMS's?

- Can't live without them: We need DBMS's to store, organize and query our data
- Lots of companies, lots of different types of platforms and infrastructures
- We need to understand how they work in order to choose the right one for our applications
- It is good to think like a database person!

Key concepts across different systems

System	Logical Data Model	Physical Storage	API	Other Features
Relational databases	Relations (i.e. tables)	B-trees, column stores, indexes, ...	SQL, ODBC	Durability, transactions, query planning, migrations, ...
TensorFlow	Tensors	NCHW, NHWC, sparse arrays, ...	Python DAG construction	query planning, distribution, specialized HW
Apache Kafka	Streams of opaque records	Partitions, compaction	Publish, subscribe	Durability, rescaling
Apache Spark RDDs	Collections of Java objects	Read external systems, cache	Functional API, SQL	Distribution, query planning, transactions*

An example relational database



An SQL Query

Query

```
SELECT e.last_name, j.job_title, d.department_name  
FROM   hr.employees e, hr.departments d, hr.jobs j  
WHERE  e.department_id = d.department_id  
AND    e.job_id = j.job_id  
AND    e.last_name LIKE 'A%';
```

Projections: attributes we want to see

Relations from where the data is coming

Joins: how one row in one relation connects to a row in another relation

Selections: criteria to filter the rows we want to see

Course Outline

Part 1: Internals of an RDBMS

- How does an RDBMS execute a SQL query [Query optimization]
- How does an RDBMS allow multiple queries to run concurrently without breaking everything up [Transactions, Concurrency & Locking]
- How does an RDBMS recover from a disaster [Recovery]

- 
- Part 1: Query Processing - Query Optimization

What happens when an app issues a SQL statement

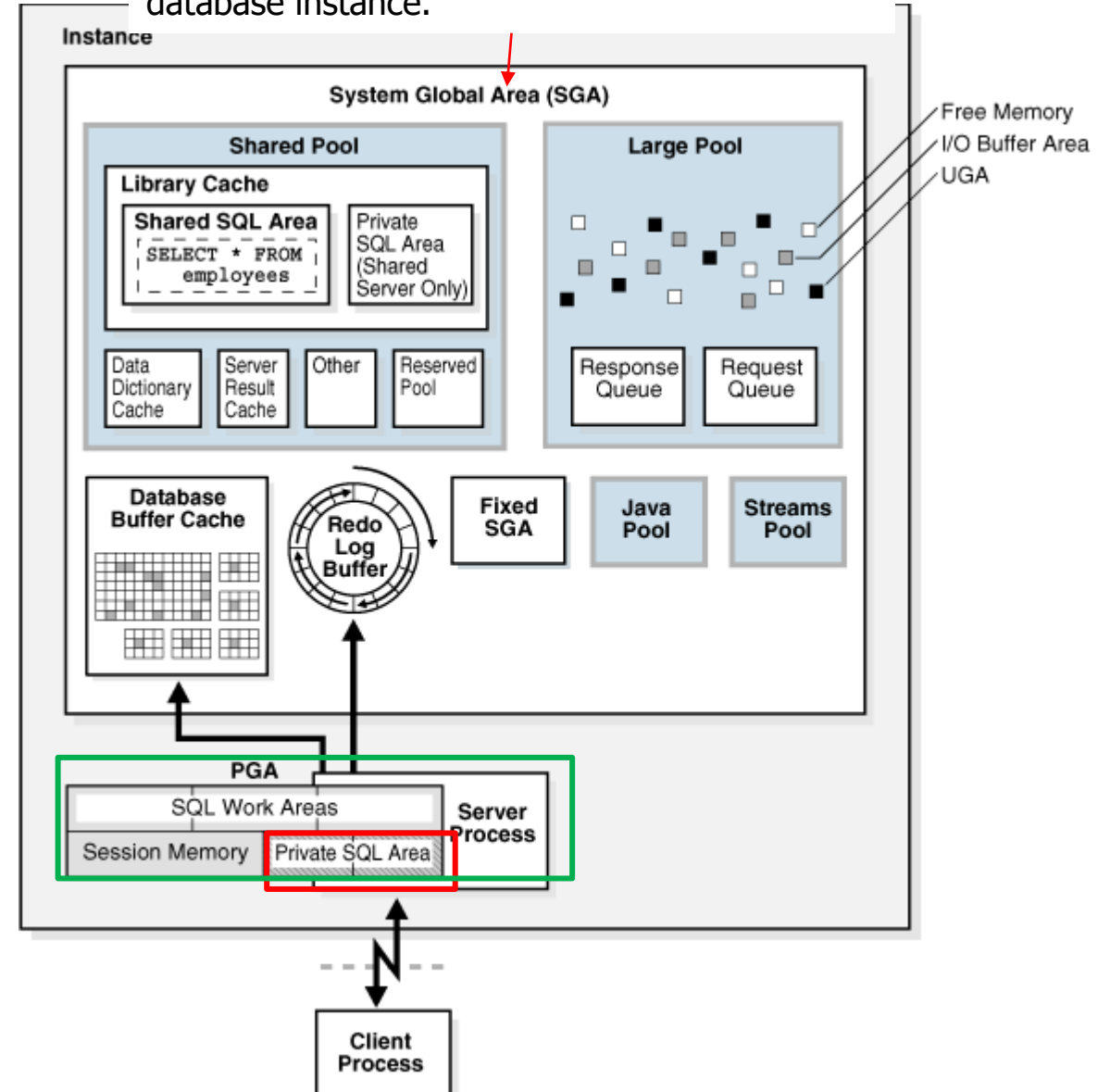
```
import java.sql.*;
class MysqlCon{
public static void main(String args[]){
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con=DriverManager.getConnection(
"jdbc:mysql://localhost:3306/sonoo","root","root");
//here sonoo is database name, root is username and password
Statement stmt=con.createStatement();
ResultSet rs=stmt.executeQuery("select * from emp");
while(rs.next())
System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));
con.close();
}catch(Exception e){ System.out.println(e);}
}
}
```

SQL Processing

When an application issues a SQL statement:

- The application makes a **parse call** to the database to prepare the statement for execution.
- The parse call opens or creates a **cursor**, which is a handle for the session-specific **private SQL area** that holds a parsed SQL statement and other processing information.
- The cursor and private SQL area are in the: **PGA (program global area)** shared memory that contains data and control information for one database instance

SGA: group of shared memory structures that contain data and control information for one database instance.



SQL Processing

Shared SQL area (cursor cache)

An area in the shared pool that contains the parse tree and [execution plan](#) for a SQL statement.

Data dictionary

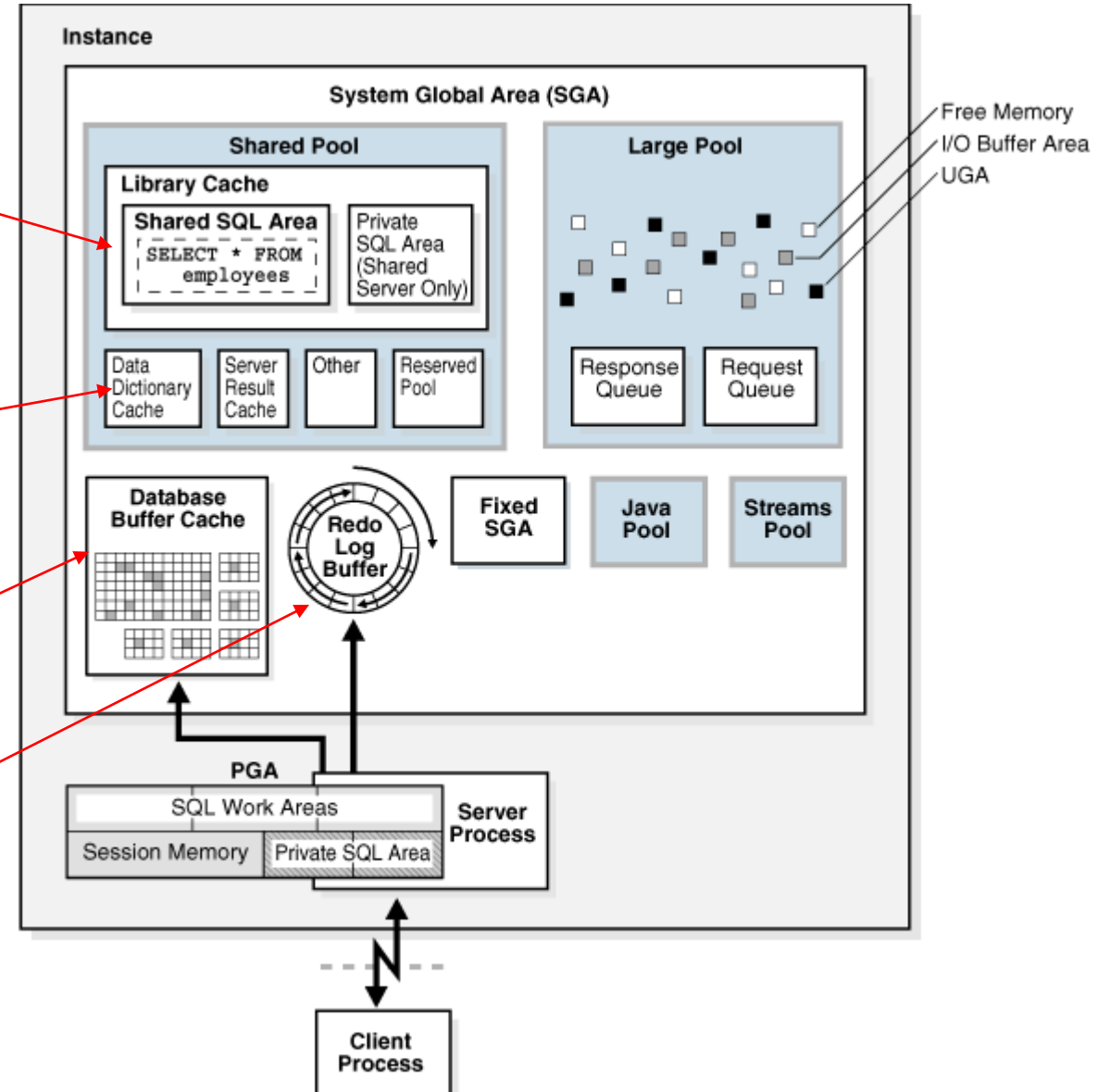
Information about the tables, attributes, indexes, statistics, etc.

Buffer

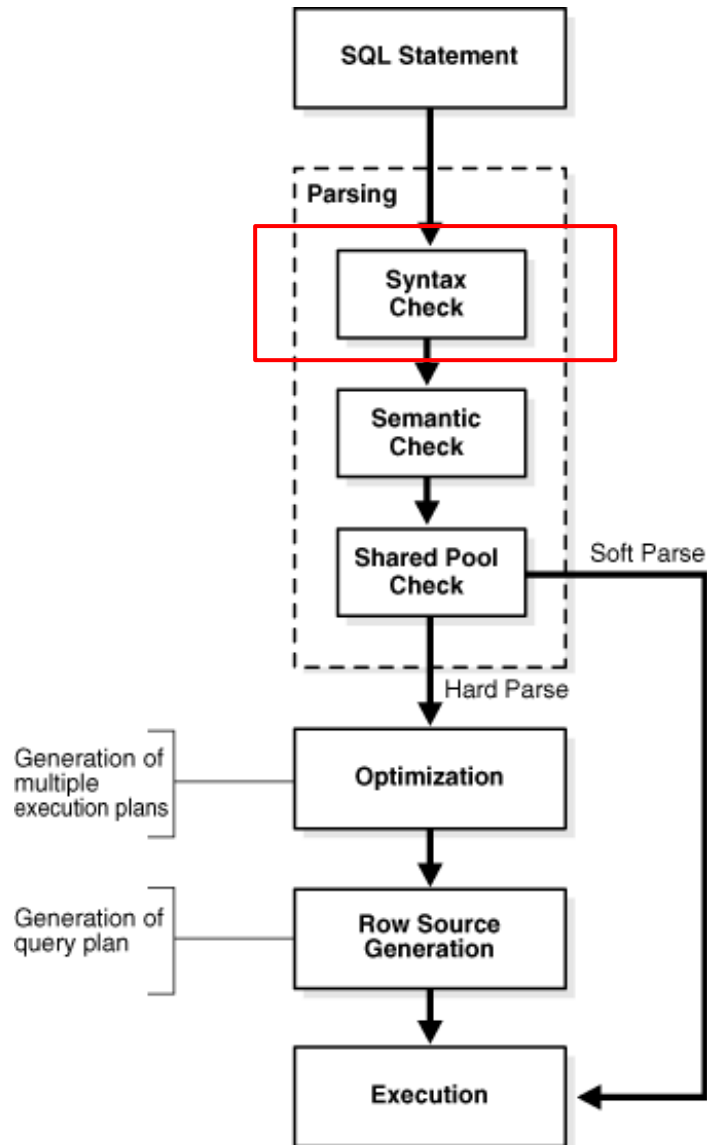
Keeps data from the disk

Redo Log Buffer

Holds information about changes made to the database



SQL Processing



Syntax Check

```
SQL> SELECT * FORM employees;
```

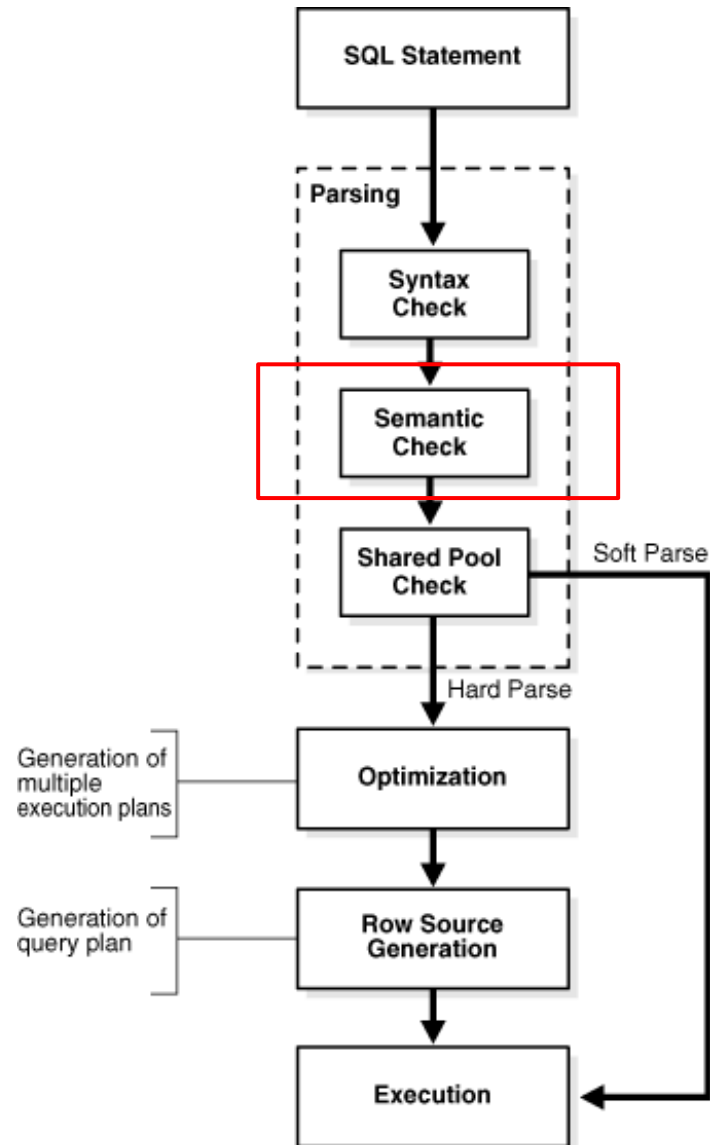
```
SELECT * FORM employees
```

```
      *
```

ERROR at line 1:

ORA-00923: FROM keyword not found where expected

SQL Processing



Semantic Check

```
SQL> SELECT * FROM nonexistent_table;
```

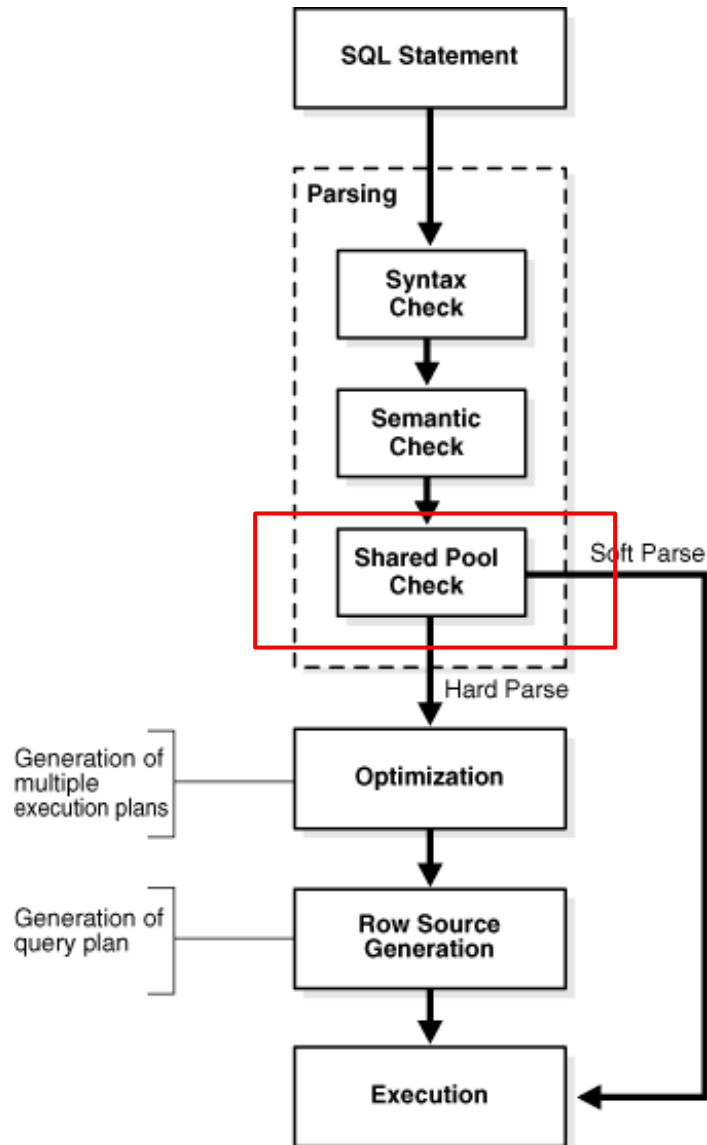
```
SELECT * FROM nonexistent_table
```

*

ERROR at line 1:

ORA-00942: table or view does not exist

SQL Processing

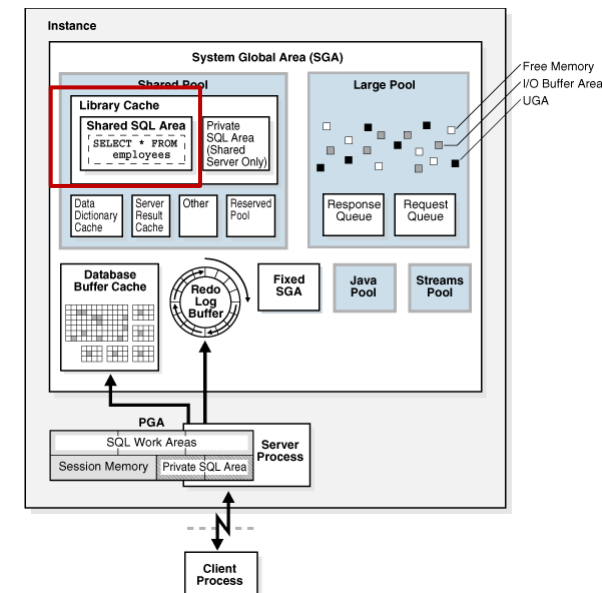


Shared Pool Check

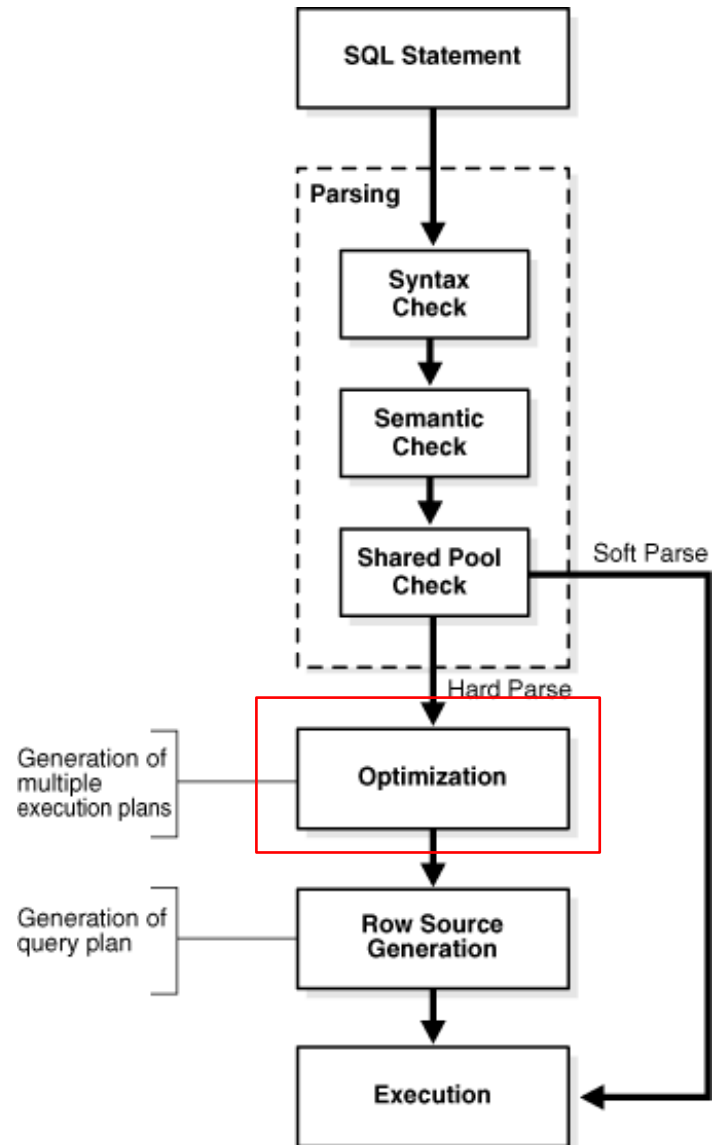
When a user submits a SQL statement, the database searches the [shared SQL area](#) to see if an existing parsed statement exists.

A SQL statement can have multiple plans in the shared pool.

In this way, the database obtains possible memory addresses of the statement.



SQL Processing



Generates an execution plan for the query

SQL Processing

Query

```
SELECT e.last_name, j.job_title, d.department_name
FROM   hr.employees e, hr.departments d, hr.jobs j
WHERE  e.department_id = d.department_id
AND    e.job_id = j.job_id
AND    e.last_name LIKE 'A%';
```

SQL Processing

Query

```
SELECT e.last_name, j.job_title, d.department_name
FROM   hr.employees e, hr.departments d, hr.jobs j
WHERE  e.department_id = d.department_id
AND    e.job_id = j.job_id
AND    e.last_name LIKE 'A%';
```

Execution Plan

Plan hash value: 975837011

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		3	189	7 (15)	00:00:01
*1	HASH JOIN		3	189	7 (15)	00:00:01
*2	HASH JOIN		3	141	5 (20)	00:00:01
3	TABLE ACCESS BY INDEX ROWID	EMPLOYEES	3	60	2 (0)	00:00:01
*4	INDEX RANGE SCAN	EMP_NAME_IX	3		1 (0)	00:00:01
5	TABLE ACCESS FULL	JOBS	19	513	2 (0)	00:00:01
6	TABLE ACCESS FULL	DEPARTMENTS	27	432	2 (0)	00:00:01

Predicate Information (identified by operation id):

```
1 - access("E"."DEPARTMENT_ID"="D"."DEPARTMENT_ID")
2 - access("E"."JOB_ID"="J"."JOB_ID")
4 - access("E"."LAST_NAME" LIKE 'A%')
   filter("E"."LAST_NAME" LIKE 'A%')
```

- Each step in an execution plan has an ID number.
- Initial spaces in the Operation column indicate hierarchical relationships.
 - E.g., if the name of an operation is preceded by two spaces, then this operation is a child of an operation preceded by one space.
 - Operations preceded by one space are children of the SELECT statement itself.

SQL Processing

SQL Execution

```
SELECT e.last_name, j.job_title, d.department_name
FROM   hr.employees e, hr.departments d, hr.jobs j
WHERE  e.department_id = d.department_id
AND    e.job_id = j.job_id
AND    e.last_name LIKE 'A%';
```

Execution Plan

Plan hash value: 975837011

Id	Operation	Name	Rows	By
0	SELECT STATEMENT		3	189 7(15) 00:00:01
*1	HASH JOIN		3	189 7(15) 00:00:01
*2	HASH JOIN		3	141 5(20) 00:00:01
3	TABLE ACCESS BY INDEX ROWID	EMPLOYEES	3	60 2 (0) 00:00:01
*4	INDEX RANGE SCAN	EMP_NAME_IX	3	1 (0) 00:00:01
5	TABLE ACCESS FULL	JOBS	19	513 2 (0) 00:00:01
6	TABLE ACCESS FULL	DEPARTMENTS	27	432 2 (0) 00:00:01

Predicate Information (identified by operation id):

```
1 - access("E"."DEPARTMENT_ID"="D"."DEPARTMENT_ID")
2 - access("E"."JOB_ID"="J"."JOB_ID")
4 - access("E"."LAST_NAME" LIKE 'A%')
   filter("E"."LAST_NAME" LIKE 'A%')
```

Join methods: techniques to execute a join of two tables

Access paths: techniques for retrieving data from the database.

SQL Processing

SQL Execution

```
SELECT e.last_name, j.job_title, d.department_name
FROM   hr.employees e, hr.departments d, hr.jobs j
WHERE  e.department_id = d.department_id
AND    e.job_id = j.job_id
AND    e.last_name LIKE 'A%';
```

Execution Plan

Plan hash value: 975837011

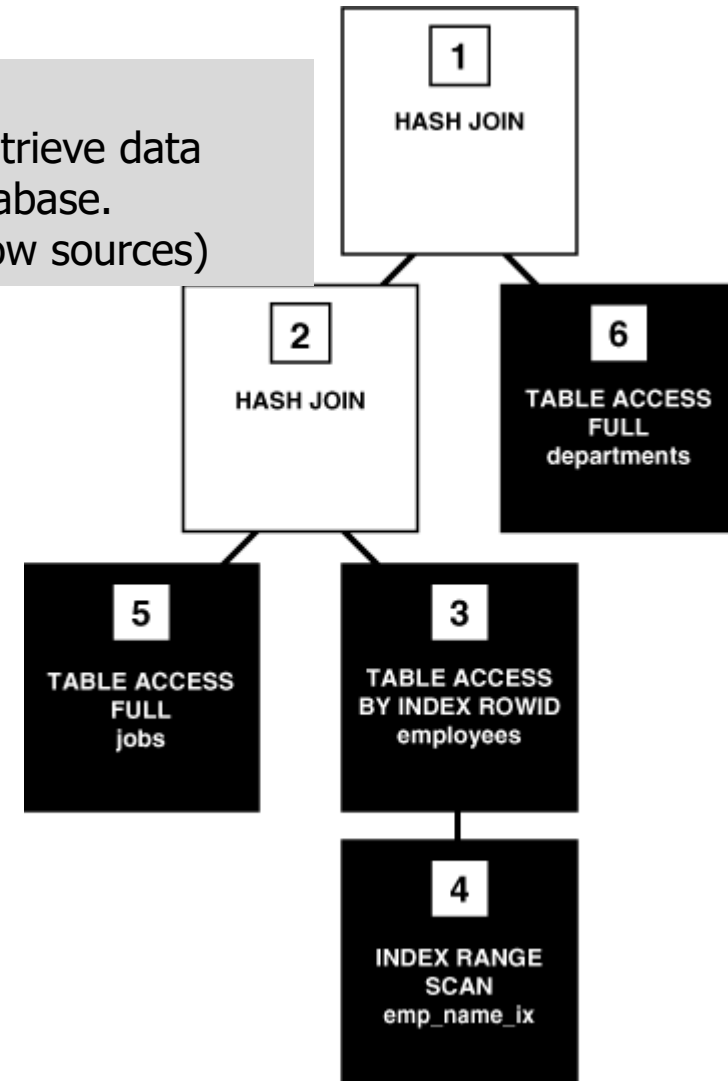
Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		3	189	7 (15)	00:00:01
*1	HASH JOIN		3	189	7 (15)	00:00:01
*2	HASH JOIN		3	141	5 (20)	00:00:01
3	TABLE ACCESS BY INDEX ROWID	EMPLOYEES	3	60	2 (0)	00:00:01
*4	INDEX RANGE SCAN	EMP_NAME_IX	3		1 (0)	00:00:01
5	TABLE ACCESS FULL	JOBS	19	513	2 (0)	00:00:01
6	TABLE ACCESS FULL	DEPARTMENTS	27	432	2 (0)	00:00:01

Predicate Information (identified by operation id):

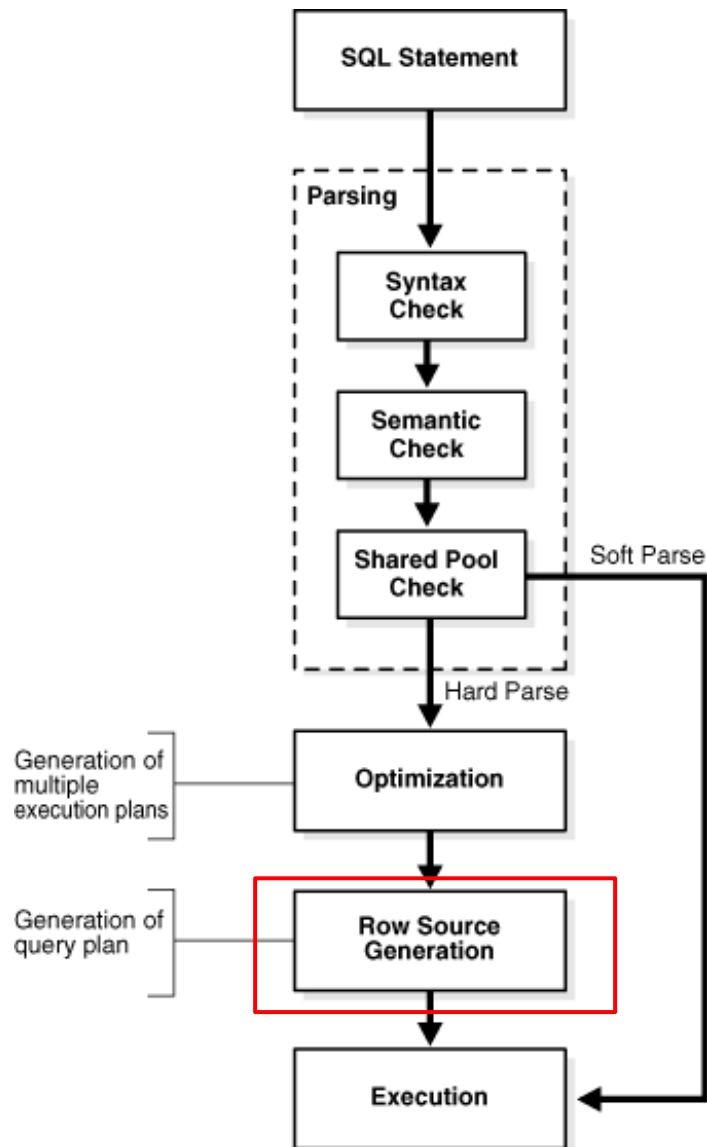
```
1 - access("E"."DEPARTMENT_ID"="D"."DEPARTMENT_ID")
2 - access("E"."JOB_ID"="J"."JOB_ID")
4 - access("E"."LAST_NAME" LIKE 'A%')
   filter("E"."LAST_NAME" LIKE 'A%')
```

QUERY TREE

(Black boxes physically retrieve data from an object in the database.
Clear boxes operate on row sources)



SQL Processing



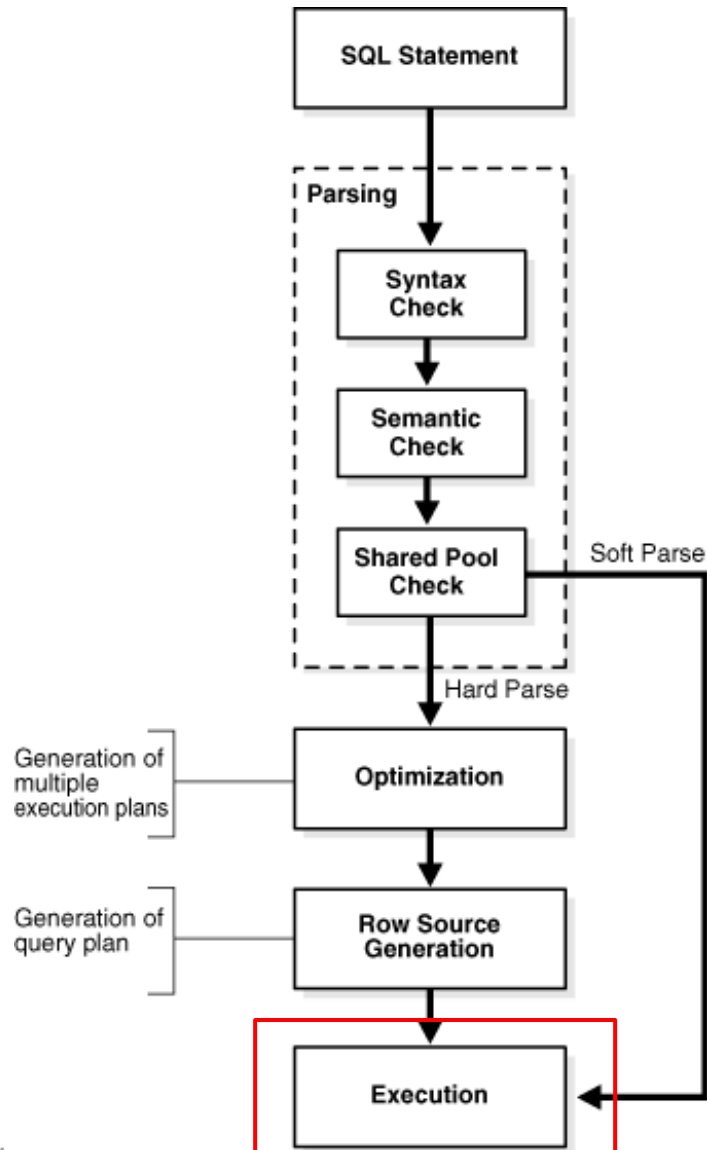
Row Source generator

The row source generator is software that receives the optimal execution plan from the optimizer and **produces an iterative execution plan** that is usable by the rest of the database.

The iterative plan is a **binary program** that, when executed by the SQL engine, **produces the result set**. The plan takes the form of a combination of steps. **Each step returns a row set**.

The next step either uses the rows in this set, or the last step returns the rows to the application issuing the SQL statement.

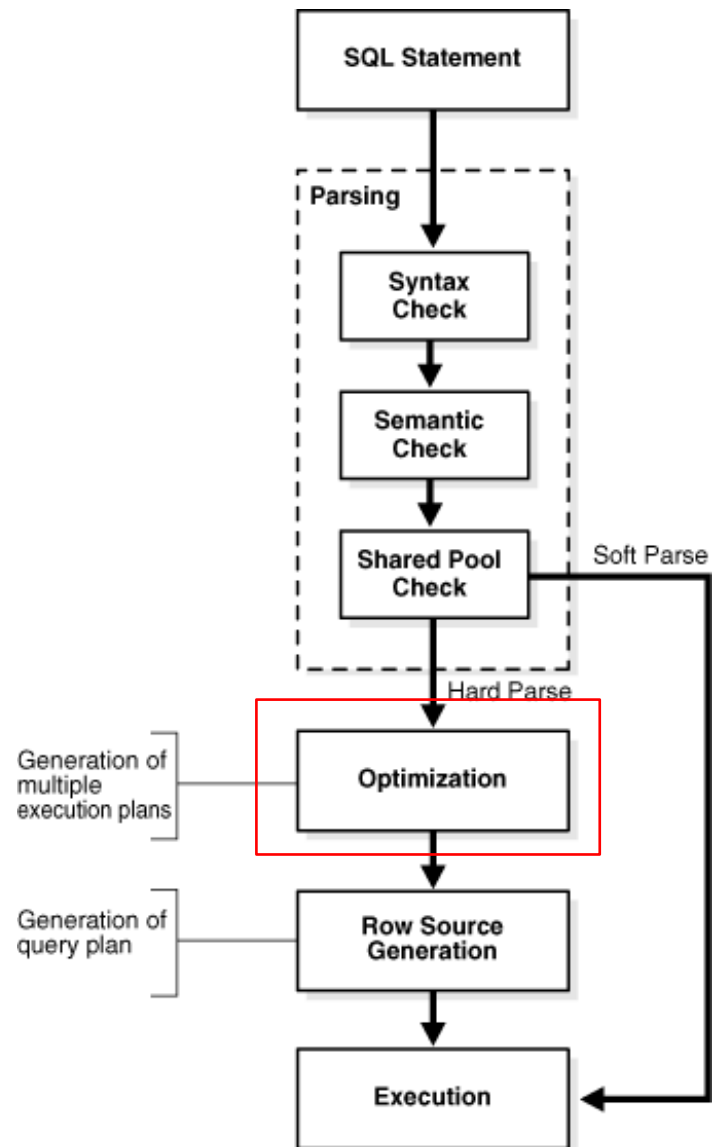
SQL Processing



Execution

- The database **reads the data** from disk into memory if the data is not in memory.
- The database also takes out any **locks** necessary to ensure data integrity
- The database **logs** any changes made during the SQL execution.
- The final stage of processing a SQL statement is **closing the cursor**.

A little bit more on the Query Optimizer



Query Optimizer

The **query optimizer** determines the most efficient method for a SQL statement to access requested data: i.e., the **execution plan with the lowest cost** among all considered candidate plans.

Example: a query might request information about employees who are managers.

- If the optimizer statistics indicate that 80% of employees are managers, then the optimizer may decide that a full table scan is most efficient.
- If statistics indicate that very few employees are managers, then reading an index followed by a table access by rowid may be more efficient than a full table scan.

Query Optimizer

The **query optimizer** determines the most efficient method for a SQL statement to access requested data: i.e., the **execution plan with the lowest cost** among all considered candidate plans.

It 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**.

It examines:

- **multiple access methods**, such as full table scan or index scans,
- different **join methods** such as nested loops and hash joins,
- different **join orders**, and
- different **possible query transformations**

Cost-based optimizer (CBO)

Query Blocks

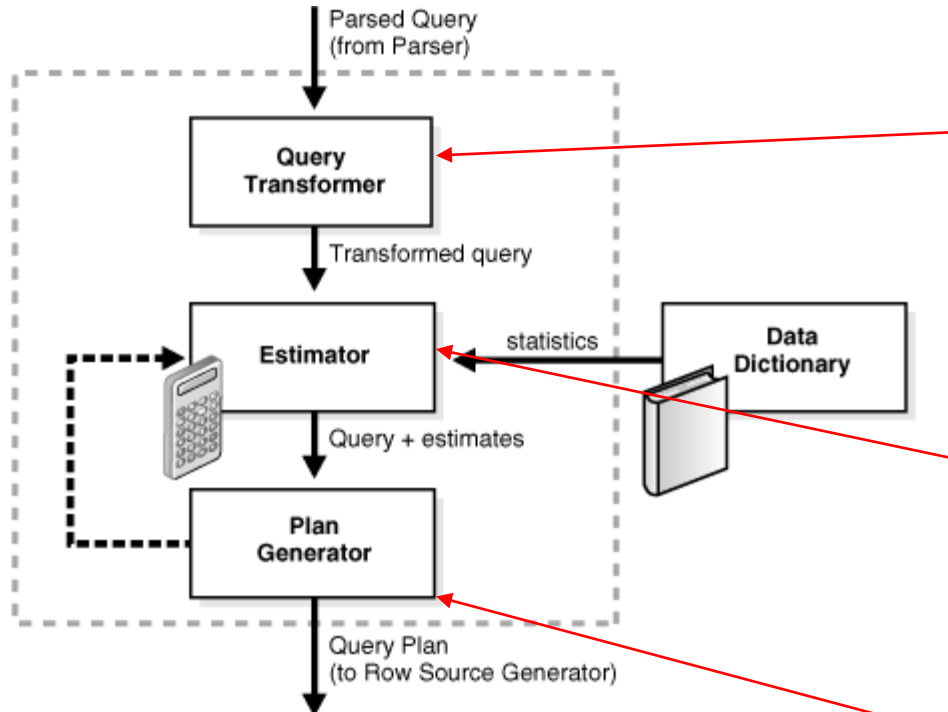
```
SELECT first_name, last_name
FROM   hr.employees
WHERE  department_id
IN      (SELECT department_id
        FROM    hr.departments
        WHERE   location_id = 1800);
```

For each query block, the optimizer generates a query subplan.

The database optimizes query blocks separately from the **bottom up**. Thus, the database optimizes the innermost query block first and generates a subplan for it, and then generates the outer query block representing the entire query.

The number of possible plans for a query block is **proportional to the number of objects in the FROM clause**. This number rises exponentially with the number of objects.

Query Optimizer Components



Query transformer

determines whether it is helpful to change the form of the query so that the optimizer can generate a better execution plan.

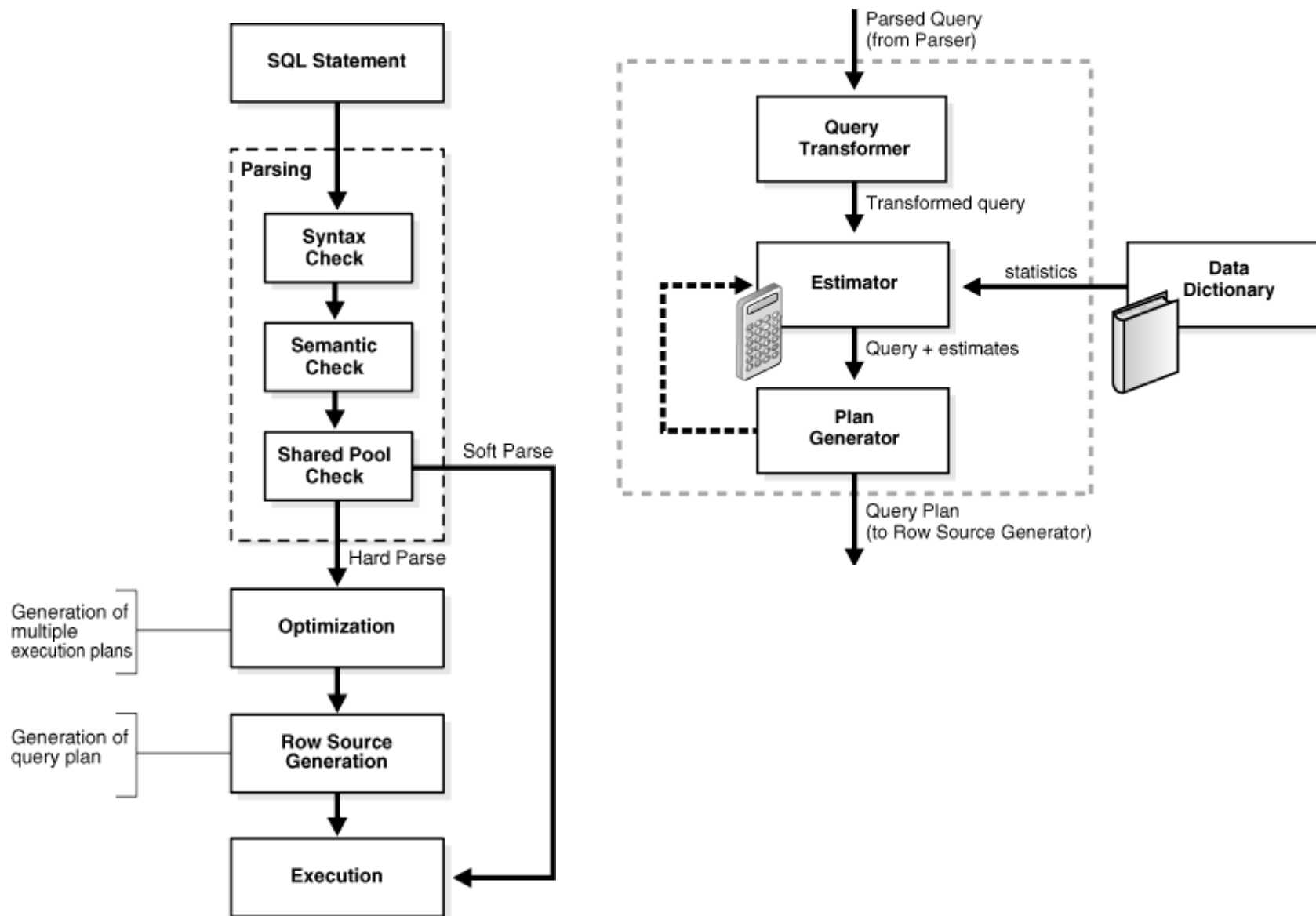
Estimator

estimates the cost of each plan based on statistics in the data dictionary.

Plan Generator

compares the costs of plans and chooses the lowest-cost plan, known as the execution plan, to pass to the row source generator.

What we have seen so far



What we are going to talk about next ...

Execution Plan

```
SELECT e.last_name, j.job_title, d.department_name
FROM   hr.employees e, hr.departments d, hr.jobs j
WHERE  e.department_id = d.department_id
AND    e.job_id = j.job_id
AND    e.last_name LIKE 'A%';
```

Execution Plan

Plan hash value: 975837011

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		3	189	7 (15)	00:00:01
*1	HASH JOIN		3	189	7 (15)	00:00:01
*2	HASH JOIN		3	141	5 (20)	00:00:01
3	TABLE ACCESS BY INDEX ROWID	EMPLOYEES	3	60	2 (0)	00:00:01
*4	INDEX RANGE SCAN	EMP_NAME_IX	3		1 (0)	00:00:01
5	TABLE ACCESS FULL	JOBS	19	513	2 (0)	00:00:01
6	TABLE ACCESS FULL	DEPARTMENTS	27	432	2 (0)	00:00:01

Predicate Information (identified by operation id):

```
1 - access("E"."DEPARTMENT_ID"="D"."DEPARTMENT_ID")
2 - access("E"."JOB_ID"="J"."JOB_ID")
4 - access("E"."LAST_NAME" LIKE 'A%')
   filter("E"."LAST_NAME" LIKE 'A%')
```

- Buffer
- Access paths
- Join methods
- Statistics
- Plan generation
- Histograms