Database Systems

Luiz Jonatã Pires de Araújo
l.araujo@innopolis.university

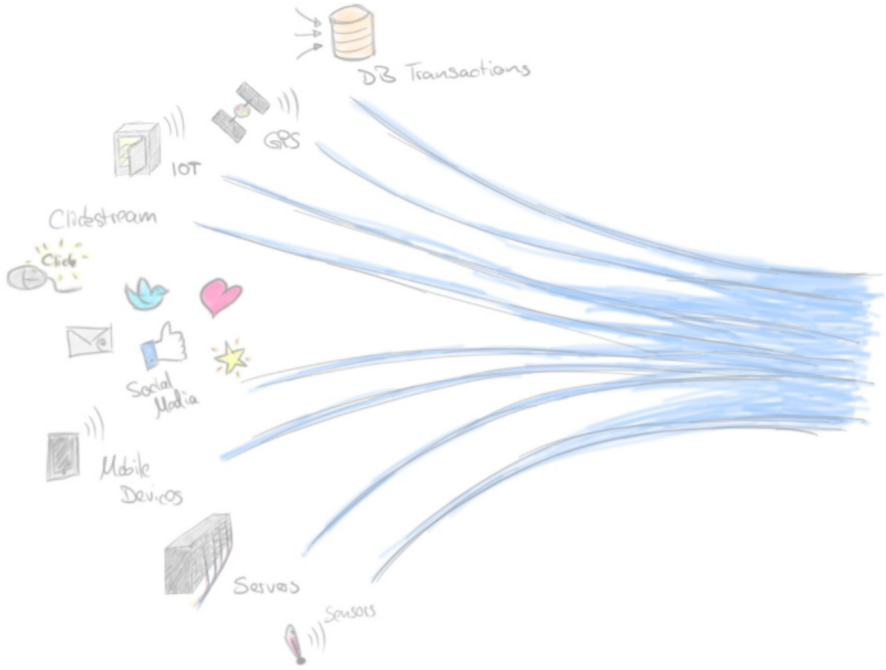
2. Relational Algebra; Advanced SQL

Giving credit where it's due:

- This material is based on the 7th edition of 'Fundamentals of Database Systems" by Elmasri and Navathe
- https://www.pearson.com/us/higher-education/program/Elmasri-Fundamentals-of-Database-Systems-7th-Edition/PGM189052.html

Outline

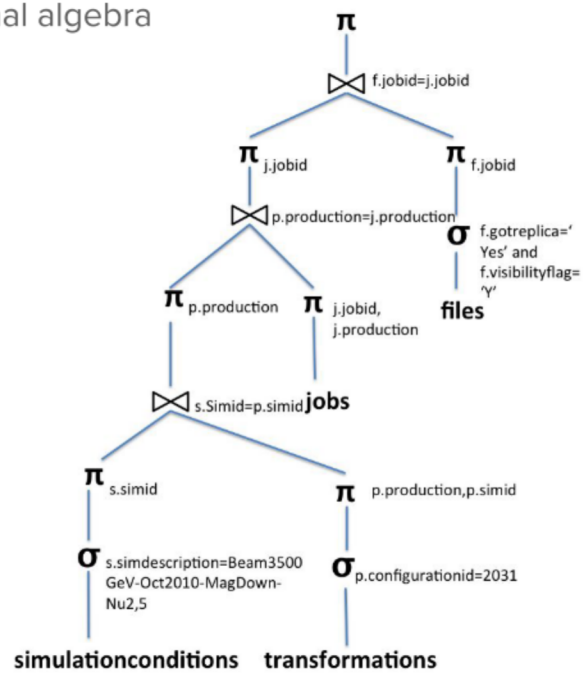
- Relational Algebra
- Advanced SQL



Relational Algebra

Concepts

- You should be now familiar with the structures and constraints of the formal relational model
- The basic set of operations for the formal relational model is the relational algebra
- These operations enable a user to specify basic retrieval requests as relational algebra expressions
- The result of a retrieval query is a new relation
- The algebra operations thus produce new relations, which can be further manipulated using operations of the same algebra
- A sequence of relational algebra operations forms a relational algebra expression, whose result will also be a relation that represents the result of a database query (or retrieval request)



Types of RA Operators

- RA operations can be divided into two groups
  - Operations from mathematical set theory: UNION, INTERSECTION, SET DIFFERENCE, and CARTESIAN PRODUCT (also known as CROSS PRODUCT)
  - Operations developed specifically for relational databases: SELECT, PROJECT, and JOIN
- Some common database requests cannot be performed with the original relational algebra operations, so additional operations were created to express these requests (aggregate functions, OUTER JOINs and OUTER UNIONs, etc)

Importance of RA

1. RA provides a formal foundation for relational model operations
2. It is used as a basis for implementing and optimizing queries in the query processing and optimization modules that are integral parts of relational database management systems (RDBMSs)
3. Core operations and functions in the internal modules of most relational systems are based on relational algebra operations

SELECT Operation

- The SELECT operation is used to choose a subset of the tuples from a relation that satisfies a condition
- For example, to select the EMPLOYEE tuples whose department is 4, or those whose salary is greater than $30,000, we can individually specify each of these two conditions with a SELECT operation as follows:
  - $\sigma_{Dno=4}(EMPLOYEE)$
  - $\sigma_{Salary>30000}(EMPLOYEE)$
- In general, the SELECT operation is denoted by:
  - $\sigma_{<selection\ condition>}(R)$
- Where the symbol σ (sigma) is used to denote the SELECT operator and the selection condition is a Boolean expression (condition) specified on the attributes of relation R

SELECT Operation

$\sigma_{(Dno=4\ \textbf{AND}\ Salary>25000)\ OR\ (Dno=5\ \textbf{AND}\ Salary>30000)}$ (EMPLOYEE)

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5

Corresponding SQL query:

```
SELECT *
FROM EMPLOYEE
WHERE Dno=4 AND Salary>25000;
```

PROJECT Operation

- The PROJECT operation selects certain columns from the table and discards the other columns
- Example: to list each employee's first and last name and salary, we can use the PROJECT operation as follows:
  - $\pi_{Lname, Fname, Salary}(EMPLOYEE)$
- The general form of the PROJECT operation is
  - $\pi_{<attribute\ list>}(R)$

Where π (pi) is the symbol used to represent the PROJECT operation, and <attribute list> is the desired sublist of attributes from the attributes of relation R

ASSIGNMENT Operation

- We can explicitly show the sequence of operations, giving a name to each intermediate relation, and using the assignment operation, denoted by ← (left arrow), as follows:

$$DEP5\_EMPS \leftarrow \sigma_{Dno=5}(EMPLOYEE)$$

$$RESULT \leftarrow \pi_{Fname,\ Lname,\ Salary}(DEP5\_EMPS)$$

```
mysql> select number,sex from (select emp_no as number, gender as sex from employees limit 5) as T;
+--------+-----+
| number | sex |
+--------+-----+
|  10001 | M   |
|  10002 | F   |
|  10003 | M   |
|  10004 | M   |
|  10005 | M   |
+--------+-----+
5 rows in set (0.06 sec)

mysql>
```

RENAME Operation

- We can also use this technique to rename the attributes in the intermediate and result relations
- To rename the attributes in a relation, we simply list the new attribute names in parentheses, as in the following example:

$$TEMP \leftarrow \sigma_{Dno=5}(EMPLOYEE)$$

$$R(First\_name, Last\_name, Salary) \leftarrow \pi_{Fname, Lname, Salary}(TEMP)$$

- The general RENAME operation when applied to a relation R of degree n is denoted by any of the following forms:

$$\rho_{S(B1, B2, \ldots, Bn)}(R) \text{ or } \rho_{(B1, B2, \ldots, Bn)}(R)$$

- where the symbol ρ (rho) is used to denote the RENAME operator, S is the new relation name, and B1, B2, ... , Bn are the new attribute names

Set Operations

- Example of UNION operation:

  DEP5_EMPS ← σDno=5(EMPLOYEE)

  RESULT1 ← πSsn(DEP5_EMPS)

  RESULT2(Ssn) ← πSuper_ssn(DEP5_EMPS)

  RESULT ← RESULT1 ∪ RESULT2

- UNION: The result of this operation, denoted by R ∪ S, is a relation that includes all tuples that are either in R or in S or in both R and S. Duplicate tuples are eliminated.
- INTERSECTION (R ∩ S): a relation that includes all tuples that are in both R and S
- SET DIFFERENCE (or MINUS): The result of this operation, denoted by R – S, is a relation that includes all tuples that are in R but not in S.

Set Operations

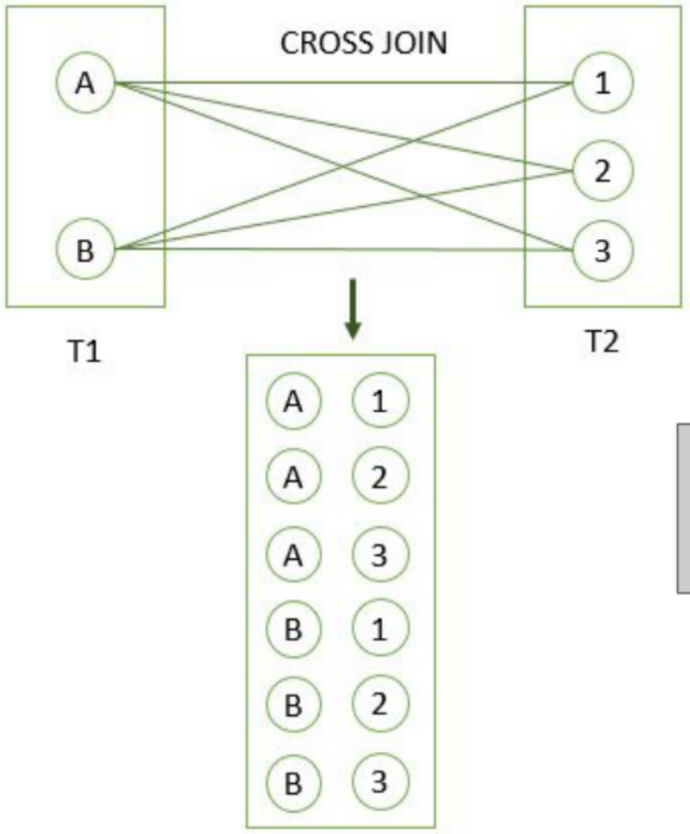
```
SELECT * FROM top_rated_films
UNION ALL -- UNION ALL keeps the duplicate records
SELECT * FROM most_popular_films;
```

```
SELECT * FROM top_rated_films
EXCEPT
SELECT * FROM most_popular_films;
```

```
SELECT * FROM most_popular_films
INTERSECT
SELECT * FROM top_rated_films;
```

CARTESIAN PRODUCT (CROSS PRODUCT) Operation

- Denoted by ×
- This binary set operation produces a new element by combining every member (tuple) from one relation (set) with every member (tuple) from the other relation (set)
- The result of R(A1, A2, ... , An) × S(B1, B2, ... , Bm) is a relation Q:
  - Q has degree n + m attributes Q(A1, A2, ... , An, B1, B2, ... , Bm)
  - R × S will have $n_R * n_S$ tuples




```
SELECT *
FROM T1
CROSS JOIN T2;
```

JOIN Operation

- The JOIN operation, denoted by ⋈, is used to combine related tuples from two relations into single “longer” tuples
- This operation is very important for any relational database with more than a single relation because it allows us to process relationships among relations
- The most common use of JOIN involves join conditions with equality comparisons only. Such a JOIN, where the only comparison operator used is =, is called an EQUIJOIN
- Example:

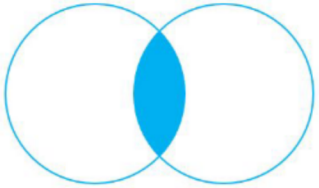
$$DEPT\_MGR \leftarrow DEPARTMENT \bowtie_{Mgr\_ssn=Ssn} EMPLOYEE$$

$$RESULT \leftarrow \pi_{Dname,\ Lname,\ Fname}(DEPT\_MGR)$$

NATURAL JOIN Operation

- NATURAL JOIN operation, denoted by *, is an EQUIJOIN and related columns have the same name
- If columns have different names, a renaming operation can be applied first. Example:

$$PROJ\_DEPT \leftarrow PROJECT * \rho_{(Dname,\ Dnum,\ Mgr\_ssn,\ Mgr\_start\_date)}(DEPARTMENT)$$

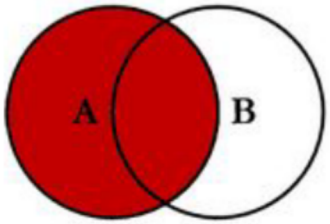


INNER JOIN

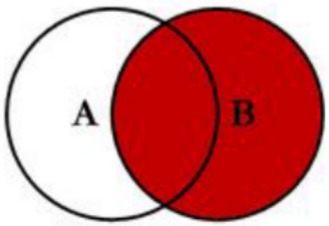
```
SELECT * FROM products
NATURAL JOIN categories;
```

```
SELECT * FROM products
INNER JOIN categories USING (category_id);
```

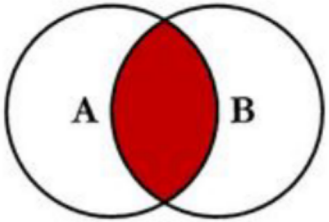
SQL JOINS



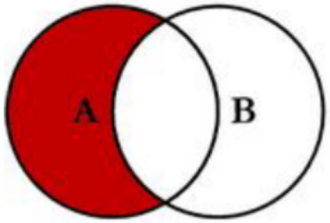

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



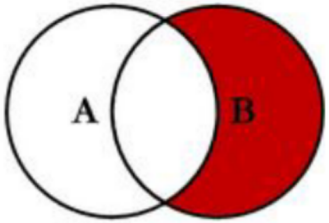

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



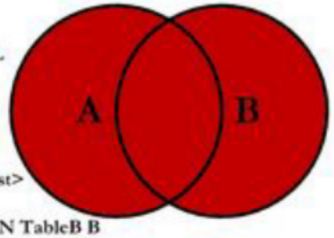

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



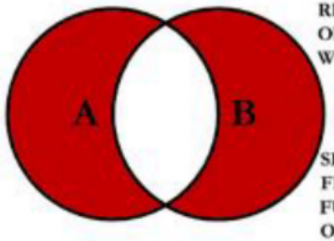

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

© C.L. Moffatt, 2008

Aggregate Functions and Grouping

- A type of request that cannot be expressed in the basic relational algebra is to specify mathematical aggregate functions on collections of values from the database
- It involves grouping the tuples in a relation by the value of some of their attributes and then applying an aggregate function independently to each group (e.g. SUM, MAX, MIN, COUNT)
- Notation:

  $<\text{grouping attributes}>\ \mathcal{I}\ <\text{function list}>\ (R)$

- Example

  $\rho_{R(\text{Dno, No\_of\_employees, Average\_sal})}\ (\ \text{Dno}\ \mathcal{I}\ \text{COUNT Ssn, AVERAGE Salary}\ (\text{EMPLOYEE})\ )$

```
SELECT      SUM (Salary), MAX (Salary), MIN (Salary), AVG (Salary)
FROM        (EMPLOYEE JOIN DEPARTMENT ON Dno = Dnumber)
WHERE       Dname = 'Research';
```

```
SELECT
     customer_id,
     SUM (amount)
FROM
     payment
GROUP BY
     customer_id
HAVING
     SUM (amount) > 200;
```

```
SELECT      COUNT (DISTINCT Salary)
FROM        EMPLOYEE;
```

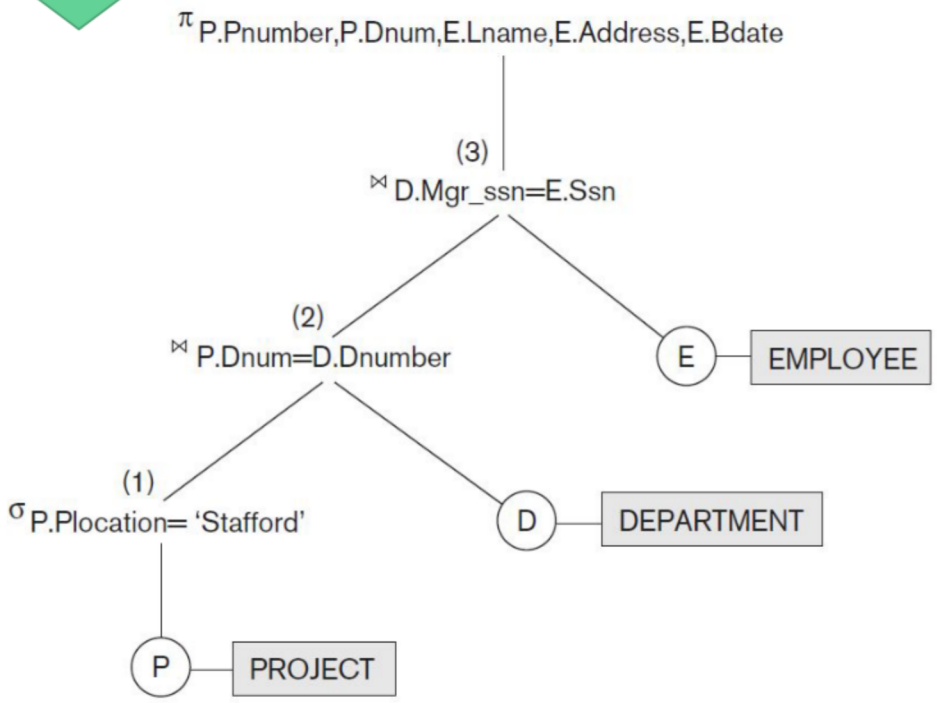
```
SELECT      Lname, Fname
FROM        EMPLOYEE
WHERE       ( SELECT      COUNT (*)
              FROM        DEPENDENT
              WHERE       Ssn = Essn ) >= 2;
```

Notation for Query Trees

Query Tree

- Query tree, also known sometimes as query evaluation tree or query execution tree, includes the relational algebra operations being executed and is used as a data structure for the internal representation of the query in an RDBMS.
- It represents the input relations of the query as leaf nodes of the tree, and represents the relational algebra operations as internal nodes
- An execution of the query tree consists of executing an internal node operation whenever its operands (represented by its child nodes) are available, and then replacing that internal node by the relation that results from executing the operation
- The execution terminates when the root node is executed and produces the result relation for the query

$$\pi_{\text{Pnumber, Dnum, Lname, Address, Bdate}}(((\sigma_{\text{Plocation='Stafford'}}(\text{PROJECT})) \bowtie_{\text{Dnum=Dnumber}}(\text{DEPARTMENT})) \bowtie_{\text{Mgr\_ssn=Ssn}}(\text{EMPLOYEE}))$$



Advanced SQL

Nested Queries

```sql
SELECT
    employeeid, firstname, lastname
FROM
    employees
WHERE
    employeeid IN (
        SELECT DISTINCT
            reportsto
        FROM
            employees);
```

outer query

subquery

```sql
SELECT  Lname, Fname
FROM    EMPLOYEE
WHERE   Salary > ALL ( SELECT  Salary
                       FROM    EMPLOYEE
                       WHERE   Dno = 5 );
```

```sql
-- SQL NESTED QUERY EXAMPLE
USE [SQL Tutorial]
GO
SELECT sub.Occupation,
       SUM(sub.YearlyIncome) AS Income,
       SUM(sub.Sales) AS TotalSale
FROM (
    SELECT [Id]
          ,[FirstName]
          ,[LastName]
          ,[Education]
          ,[Occupation]
          ,[YearlyIncome]
          ,[Sales]
    FROM [Employee Table]
    WHERE [Sales] > 500
    ) AS sub
GROUP BY sub.Occupation
```

100 %

©tutorialgateway.org

Results | Messages

	Occupation	Income	TotalSale
1	Management	310000	12924.06
2	Professional	330000	11516.52
3	Skilled Manual	110000	3019.0882

```sql
SELECT ProductID,
       Name,
       ListPrice
FROM   production.Product
WHERE  ListPrice > (SELECT AVG(ListPrice)
                    FROM   Production.Product)
```

subquery

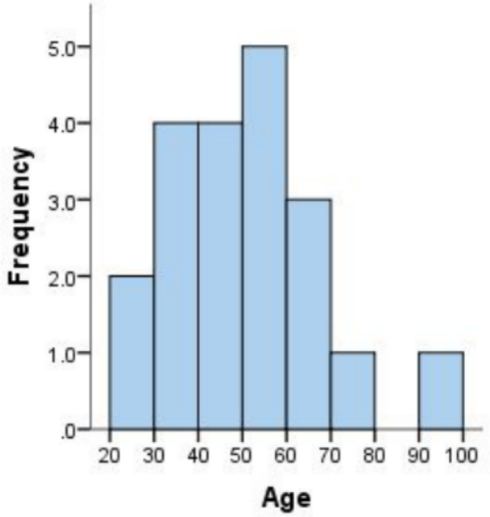
EXISTS

```
SELECT   E.Fname, E.Lname
FROM     EMPLOYEE AS E
WHERE    EXISTS ( SELECT   *
                  FROM     DEPENDENT AS D
                  WHERE    E.Ssn = D.Essn AND E.Sex = D.Sex
                           AND E.Fname = D.Dependent_name);
```

EXISTS and NOT EXISTS are typically used in conjunction with a correlated nested query :(

CASE

```
SELECT title,
       length,
       CASE
           WHEN length> 0 AND length <= 50 THEN 'Short'
           WHEN length > 50 AND length <= 120 THEN 'Medium'
           WHEN length> 120 THEN 'Long'
       END duration
FROM film
ORDER BY title;
```



Views (Virtual Tables)

- A view does not necessarily exist in physical form; it is considered to be a virtual table
- Base tables, on the other hands, tuples are always physically stored in the database

```
CREATE VIEW     WORKS_ON1
AS SELECT       Fname, Lname, Pname, Hours
    FROM        EMPLOYEE, PROJECT, WORKS_ON
    WHERE       Ssn = Essn AND Pno = Pnumber;

CREATE VIEW     DEPT_INFO(Dept_name, No_of_emps, Total_sal)
AS SELECT       Dname, COUNT (*), SUM (Salary)
    FROM        DEPARTMENT, EMPLOYEE
    WHERE       Dnumber = Dno
    GROUP BY    Dname;
```

```
CREATE VIEW comedies AS
    SELECT *
    FROM films
    WHERE kind = 'Comedy';
```

Triggers

- In many cases it is convenient to specify the type of action to be taken when certain events occur and when certain conditions are satisfied
- For example, it may be useful to specify a condition that, if violated, causes some user to be informed of the violation
- A manager may want to be informed if an employee's travel expenses exceed a certain limit by receiving a message whenever this occurs
- The action that the DBMS must take in this case is to send an appropriate message to that user
- The condition is thus used to monitor the database

```
CREATE TRIGGER SALARY_VIOLATION
BEFORE INSERT OR UPDATE OF SALARY, SUPERVISOR_SSN
   ON EMPLOYEE
FOR EACH ROW
   WHEN ( NEW.SALARY > ( SELECT SALARY FROM EMPLOYEE
                         WHERE SSN = NEW.SUPERVISOR_SSN ) )
                         INFORM_SUPERVISOR(NEW.Supervisor_ssn,
                         NEW.Ssn );
```

See: Documentation: 9.2: Trigger Procedures