Lesson 7.2: Complex SQL

CSC430/530 – DATABASE MANAGEMENT SYSTEMS

DR. ANDREY TIMOFEYEV

OUTLINE

- Multisets & set operations.
- Pattern matching & additional operators.
- Three-valued logic.
- Nested queries.
- Exists, explicit sets & renaming.
- Joined tables.
- Aggregate functions.
- Group by & Having.

MULTISETS, SETS & SET OPERATIONS

- SQL treats tables as **multisets**.
  - **Duplicated** tuples can appear in the **result** of a query.
- **DISTINCT** is used in **SELECT** clause to **eliminate** duplicates in the query result.
  - Query 1 - Retrieve the salary of every employee; retrieve all distinct salary values.
- SQL supports **set operation** commands.
  - Duplicates are **eliminated** in the results of set operation queries.
  - **UNION**, **INTERSECT**, **EXCEPT** (*set difference*).
    - Query 2 - Make a list of all project numbers for projects that involve an employee whose last name is ‘Smith’, either as a worker or as a manager of the department that controls the project.
- Adding **ALL** to **set operations** turn them into **multiset** operations.
  - Duplicates are **not eliminated**.
  - **UNION ALL**, **INTERSECT ALL**, **EXCEPT ALL**.

PATTERN MATCHING & ADDITIONAL OPERATORS

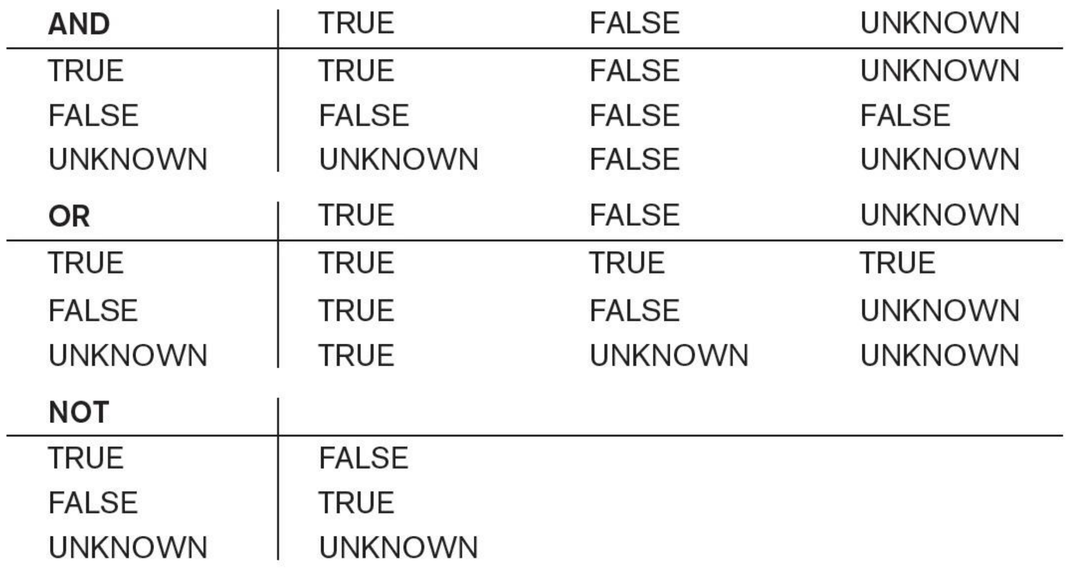
- **LIKE** is used for substring **pattern matching**.
  - "_" replaces a **single** character, "%" replaces an **arbitrary number** of characters.
    - Query 3.1 - Retrieve all employees whose address is in Houston, Texas.
    - Query 3.2 - Find all employees who were born during the 1970s.

- **Additional** operators can be used in a query.
  - Arithmetic operators + - * /
    - Query 4 - Show the resulting salaries if every employee working on the 'Product' project is given a 10% raise.
  - **BETWEEN** operator (equivalent to <= AND >=).
    - Query 5 - Retrieve all employees in department 5 whose salary is between $30,000 and $40,000.

- **ORDER BY** is used to order tuples in a query result.
  - **DESC, ASC**.
    - Query 6 - Retrieve a list of employees and the projects they are working on, ordered by department and, within each department, ordered alphabetically by last name, then first name.

THREE-VALUED LOGIC

- **NULL** is used in SQL to represent **missing values**.
  - Unknown, unavailable, or not applicable.
- When comparing attributes with **NULL** value, the **result** is considered to be ***UNKNOWN***
  - Could be ***TRUE*** or ***FALSE***.
- SQL uses **three-valued logic**: **TRUE**, **FALSE**, **UNKNOWN**.
  - Only combinations of tuples that evaluate to **TRUE** in **WHERE** clause condition are selected.



AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOWN	FALSE	UNKNOWN

OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN

NOT	
TRUE	FALSE
FALSE	TRUE
UNKNOWN	UNKNOWN

- **IS** & **IS NOT** are used to check if an attribute value is (not) equal to **NULL**.
  - Query 7 - Retrieve the names of all employees who do not have supervisors.

NESTED QUERIES

- Queries can be **nested inside SELECT**, **FROM** or **WHERE** clauses of other **queries**.
  - **Inner** (*nested*) query, **outer** query.
    - Query 8 - Make a list of all project numbers for projects that involve an employee whose last name is ‘Smith’, either as a worker or as a manager of the department that controls the project.
- **Correlated nested queries** – condition in the **WHERE** clause of **inner query** references some attribute of a relation declared in the **outer query**.
  - **Inner** (*nested*) query is evaluated once for each tuple (or combination of tuples) in the **outer query**.
  - **Aliasing** is recommended to use in nested queries.
    - Query 9 - Retrieve the name of each employee who has a dependent with the same first name and is the same sex as the employee.
- **Comparison operators** can be used with **ANY** or **ALL** quantifiers.
  - Query 10 - List the names of employees whose salary is greater than the salary of all the employees in department 5.

EXISTS

- **EXISTS** is a Boolean function that is used in **WHERE** clause.
  - Typically used in conjunction with a **correlated nested query**.
- **EXISTS** checks whether the **result** of a **nested query** is **empty** (contains no tuples) or **not**.
  - **TRUE** if the nested query result contains at least **one tuple**.
  - **FALSE** if the nested query result contains **no tuples**.
    - Query 11 - Alternative to Query 9, with EXISTS.
    - Query 12 - Retrieve the names of employees who have no dependents.
    - Query 13 - List the names of managers who have at least one dependent.

EXPLICIT SETS & RENAMING

- **Explicit sets** of values can be used in **WHERE** clause.
  - Enclosed in **parenthesis**.
    - Query 14 – Retrieve Social Security numbers of all employees who work on project numbers 1, 2, or 3.
- **AS** command is used to **rename** (*alias*) **relations** or **attributes** in appropriate part of a query.
  - Query 15 – Retrieve last name of each employee and his/her supervisor while renaming the resulting attribute names as Employee_name and Supervisor_name.

JOINED TABLES

- **Tables** can be **joined** explicitly in **FROM** clause.
- **JOIN … ON**.
  - Query 16 - Retrieve the name and address of every employee who works for the ‘Research’ department.
- **NATURAL JOIN**.
  - Query 17 - Rename department number attribute name of DEPARTMENT relation and use NATURAL JOIN.
- **Multiway JOIN … ON**.
  - Query 18 - For every project located in ‘Stafford’, list the project number, the controlling department number, and the department manager’s last name, address, and birth date.

AGGREGATE FUNCTIONS

- **Aggregate functions** are used to **summarize** information from **multiple** tuples into a **single-tuple** summary.
  - Normally used in **SELECT** or **HAVING** clause.
- **COUNT**, **SUM**, **MAX**, **MIN**, **AVG**.
  - **COUNT** function returns the number of tuples or values as specified in a query.
  - **SUM, MIN**, **MAX,** and **AVG** are used with attributes.
- **NULL** values are discarded when **aggregate functions** are applied to a particular attribute.
  - Exception is **COUNT(*)** since it counts tuples instead of values.
- Query 19.1 - Find the sum of the salaries of all employees, the maximum salary, the minimum salary, and the average salary without attribute renaming.
- Query 19.2 - Find the sum of the salaries of all employees, the maximum salary, the minimum salary, and the average salary with attribute renaming.
- Query 20 - Find the sum of the salaries of all employees of the ‘Research’ department, as well as the maximum salary, the minimum salary, and the average salary in this department.
- Query 21 - Count the number of distinct salary values in the database.
- Query 22 - Retrieve the names of all employees who have two or more dependents.

GROUP BY & HAVING

- **GROUP BY** is used to **partition** relation into **non-overlapping subsets**.
  - Each **partition** (*group*) consists of the tuples that have **same value** for some **grouping attribute(s)**.
- **GROUP BY** specifies the **grouping attribute(s)**, which also appear(s) in **SELECT** clause.
  - Query 23 - For each department, retrieve the department number, the number of employees in the department, and their average salary.
- **GROUP BY** is applied **after joining** tables if used in conjunction with **join condition**.
  - Query 24 - For each project, retrieve the project number, the project name, and the number of employees who work on that project.
- **HAVING** is used with **GROUP BY** to retrieve only those groups that **satisfy** certain **condition**.
  - Query 25 - For each project on which more than two employees work, retrieve the project number, the project name, and the number of employees who work on the project.
- **HAVING** is applied **after** the **condition** is **evaluated** in **WHERE** clause.
  - **WHERE** clause is executed **first**, to select **individual tuples** or **joined tuples**;
  - **HAVING** clause is applied **later**, to select **individual groups** of **tuples**.

SUMMARY OF SQL QUERIES

- **General structure** of **retrieval** SQL query.
  - **SELECT** <*attribute and function list*>
    FROM <*table list*>
    [**WHERE** <*condition*>]
    [**GROUP BY** <*grouping attribute(s)*>]
    [**HAVING** <*group condition*>]
    [**ORDER BY** <*attribute list*>];
    - **SELECT** lists the attributes or functions to be retrieved.
    - **FROM** specifies all relations (tables) needed in the query, including joined relations.
    - **WHERE** specifies the conditions for selecting the tuples from these relations, including join conditions.
    - **GROUP BY** specifies grouping attributes.
    - **HAVING** specifies a condition on the groups being selected rather than on the individual tuples.
    - Aggregate functions (**COUNT**, **SUM**, **MIN**, **MAX**, **AVG**) are used in conjunction with grouping, or applied to all the selected tuples in a query without a GROUP BY clause.
    - **ORDER BY** specifies an order for displaying the result of a query.