**2025-3-6 COMPSCI751 Summary**

**英文原文 & 中文翻译**

1. Background: Relational Algebra and Its Importance

• Relational Algebra provides the foundational theory for modern relational databases. Its operations include:

• Selection (σ)

• Projection (π)

• Union (∪)

• Difference (−)

• Cartesian Product (×)

• Natural Join (⋈)

• Theta Join (⋈ with conditions)

• Rename (ρ)

• Assignment (←)

• Expression Trees for query representation

These operations paved the way for SQL, a high-level declarative language for managing and querying relational databases.

2. Introduction to SQL

SQL (Structured Query Language) is a standard query language used to communicate with relational databases. It originated from Edgar F. Codd’s relational model, proposed in the 1970s, and has evolved into multiple versions (SQL-86, SQL-89, SQL-92, etc.).

2.1 Two Main Parts of SQL

1. Data Definition Language (DDL)

• Used to define database schemas (e.g., creating tables, specifying constraints such as primary keys and foreign keys).

• Common statements: CREATE TABLE, ALTER TABLE, DROP TABLE.

2. Data Manipulation Language (DML)

• Used to query and update data (e.g., inserting, deleting, updating records, or retrieving results with SELECT-FROM-WHERE).

• Common statements: SELECT, INSERT, UPDATE, DELETE.

3. Fundamental SQL Query Structure

A basic SQL query follows:

SELECT A1, A2, ..., An

FROM R1, R2, ..., Rm

WHERE condition;

• SELECT clause: Attributes (columns) to be retrieved.

• FROM clause: Relation(s) or table(s) from which data is taken.

• WHERE clause: Conditions that filter tuples.

In standard practice:

• If multiple tables are listed in the FROM clause, it implies a Cartesian product or some form of join.

• The result of a SELECT query is always another relation (table).

4. Frequently Used Clauses and Features

1. Removing Duplicates:

• By default, SQL uses a multiset (allows duplicates).

• To remove duplicates in the result, add DISTINCT in the SELECT clause.

SELECT DISTINCT department\_name

FROM instructor;

2. Alias (Renaming):

• The keyword AS can rename tables or columns for readability.

SELECT T.name AS teacher\_name

FROM instructor AS T;

3. String Matching with LIKE:

• % matches a sequence of characters.

• \_ matches a single character.

SELECT name

FROM instructor

WHERE name LIKE '%in%';

4. Ordering Results:

• ORDER BY <column> [ASC|DESC] sorts the output.

SELECT name

FROM instructor

ORDER BY name DESC;

5. Range Queries (BETWEEN):

• BETWEEN x AND y is inclusive of x and y.

SELECT name

FROM instructor

WHERE salary BETWEEN 90000 AND 100000;

6. Set Operations:

• UNION, INTERSECT, EXCEPT combine or compare result sets.

• By default, set operations remove duplicates. To retain duplicates, use UNION ALL, etc.

SELECT course\_id FROM section

WHERE semester = 'Fall'

UNION

SELECT course\_id FROM section

WHERE semester = 'Spring';

7. Join Conditions:

• Natural Join automatically matches common attributes.

• Theta Join (or just a condition in the WHERE clause) explicitly matches attributes:

SELECT \*

FROM instructor, teaches

WHERE instructor.ID = teaches.ID;

5. Briefly Touched-On Knowledge Points (Expanded)

The instructor mentioned several ideas briefly. Below are expansions on those topics:

1. Primary Key vs. Foreign Key

• A Primary Key is a column (or combination of columns) that uniquely identifies a row in a table. Each table can have only one primary key.

• A Foreign Key is a column (or combination of columns) in one table that refers to the primary key of another table (or sometimes the same table for self-reference). This enforces referential integrity: the foreign key values must match existing primary key values in the referenced table.

2. Referential Integrity

• Ensures that relationships between tables remain valid.

• If a row in a “child” table (with a foreign key) references a row in the “parent” table, that parent row must exist before insertion or remain valid before updates/deletion.

• SQL can specify cascading actions (ON UPDATE CASCADE, ON DELETE CASCADE) so that changes in the parent reflect in the child, or restrict certain deletions/updates to maintain consistency.

3. Relational Model Foundation

• Edgar F. Codd’s relational model introduced the concept of organizing data into relations (tables) with tuples (rows).

• The relational model abstracts physical details and presents data in a logical, tabular format, allowing for powerful, declarative queries.

4. NULL and Three-Valued Logic

• If a column’s value is unknown or not applicable, it can store NULL.

• NULL is neither equal nor not equal to other values, which can affect conditions in WHERE.

• Comparisons involving NULL often result in UNKNOWN, requiring special handling (like IS NULL, IS NOT NULL).

5. Subqueries, SOME/ANY, and ALL (if the system supports them)

• You can nest queries inside a WHERE clause:

SELECT name

FROM instructor

WHERE salary > SOME(

SELECT salary

FROM instructor

WHERE department\_name = 'Comp. Sci.'

);

• SOME (or ANY in some SQL dialects) compares the given value to each value in the subquery, returning TRUE if the condition holds for at least one row.

• ALL requires the condition to hold for every row returned by the subquery.

6. Basic Code Templates

Below are some template examples for quick reference.

1. Table Creation (DDL)

CREATE TABLE instructor (

ID CHAR(5),

name VARCHAR(20) NOT NULL,

department\_name VARCHAR(20),

salary DECIMAL(8,2),

PRIMARY KEY (ID),

FOREIGN KEY (department\_name) REFERENCES department(dept\_name)

);

2. Inserting Records

INSERT INTO instructor (ID, name, department\_name, salary)

VALUES ('22222', 'Smith', 'History', 45000);

3. Updating Records

UPDATE instructor

SET salary = salary + 1000

WHERE department\_name = 'Comp. Sci.';

4. Deleting Records

DELETE FROM instructor

WHERE ID = '22222';

5. Selecting and Joining

SELECT i.ID, i.name, t.course\_id

FROM instructor AS i

JOIN teaches AS t

ON i.ID = t.ID

WHERE i.department\_name = 'Comp. Sci.';

6. Using Subqueries (if supported)

SELECT name

FROM instructor

WHERE salary > (SELECT AVG(salary) FROM instructor);

7. Final Tips

• Remember the difference between declarative (SQL) and procedural styles. SQL focuses on what you want, not how it is retrieved.

• Always check your DB system’s support for features like SOME, ANY, or advanced string functions, as each SQL implementation may vary.

• Good practices in relational database design—like normalization—start from the relational algebra principles and follow referential integrity rules.

中文版本（重点知识总结报告）

1. 背景：关系代数及其重要性

• 关系代数为现代关系型数据库奠定了理论基础，主要操作包括：

• 选择 (σ)

• 投影 (π)

• 并 (∪)

• 差 (−)

• 笛卡儿积 (×)

• 自然连接 (⋈)

• θ 连接（带条件的连接）

• 重命名 (ρ)

• 赋值 (←)

• 表达式树 用于表示查询结构

这些运算为后来的 SQL（高级声明式语言）奠定了理论基础，使我们能高效地管理和查询关系数据库。

2. SQL 简介

SQL（Structured Query Language）是一种标准的查询语言，用于与关系数据库交互。它源于 Edgar F. Codd 在 20 世纪 70 年代提出的关系模型，并演进为多个版本（SQL-86、SQL-89、SQL-92 等）。

2.1 SQL 的两大部分

1. 数据定义语言（DDL）

• 用于定义数据库模式（如创建表、指定主键和外键等）。

• 常用语句：CREATE TABLE、ALTER TABLE、DROP TABLE。

2. 数据操作语言（DML）

• 用于查询和更新数据（插入、删除、更新记录或使用 SELECT-FROM-WHERE 检索结果）。

• 常用语句：SELECT、INSERT、UPDATE、DELETE。

3. 基本 SQL 查询结构

一个基本的 SQL 查询格式如下：

SELECT A1, A2, ..., An

FROM R1, R2, ..., Rm

WHERE condition;

• SELECT 子句：要检索的属性（列）。

• FROM 子句：数据来源表（可同时列出多个表）。

• WHERE 子句：筛选元组的条件。

在实践中：

• 如果在 FROM 中列出多个表，默认是做笛卡儿积或使用连接。

• SELECT 查询的结果始终是一个新的关系（表）。

4. 常用子句和功能

1. 去重 (DISTINCT)

• SQL 默认是多重集（允许重复）。

• 在 SELECT 后加 DISTINCT 可以去重：

SELECT DISTINCT department\_name

FROM instructor;

2. 别名（重命名）

• 关键字 AS 用来给表或列重命名，便于阅读：

SELECT T.name AS teacher\_name

FROM instructor AS T;

3. 字符串匹配 (LIKE)

• % 匹配任意长度的字符序列。

• \_ 匹配单个任意字符。

SELECT name

FROM instructor

WHERE name LIKE '%in%';

4. 结果排序 (ORDER BY)

• ORDER BY <column> [ASC|DESC] 根据字段进行排序。

SELECT name

FROM instructor

ORDER BY name DESC;

5. 范围查询 (BETWEEN)

• BETWEEN x AND y 包含端点 x 和 y：

SELECT name

FROM instructor

WHERE salary BETWEEN 90000 AND 100000;

6. 集合操作

• UNION、INTERSECT、EXCEPT 用于集合合并或比较。

• 缺省会去重，想保留重复可用 UNION ALL 等。

SELECT course\_id FROM section

WHERE semester = 'Fall'

UNION

SELECT course\_id FROM section

WHERE semester = 'Spring';

7. 连接条件

• 自然连接（NATURAL JOIN）自动匹配共同属性。

• θ 连接（或在 WHERE 中加条件）手动匹配属性：

SELECT \*

FROM instructor, teaches

WHERE instructor.ID = teaches.ID;

5. 老师简单带过的一些知识点（扩展）

1. 主键与外键

• 主键（Primary Key）：在一张表中唯一标识一个元组的列或列组合。每张表只能有一个主键。

• 外键（Foreign Key）：在一张表中引用另一张表（或自身表）主键的列或列组合，用于保证引用完整性：外键的值必须先在被引用的主键表里存在，才可插入。

2. 参照完整性

• 用于确保多表间的引用关系有效。

• 当子表（有外键的一方）引用的父表中对应记录发生更新/删除时，需要根据设置的级联规则（ON UPDATE CASCADE、ON DELETE CASCADE）或限制规则，来决定子表数据是否同步或是否允许父表记录被删除。

3. 关系模型基础

• Edgar F. Codd 提出的关系模型，将数据组织成关系（表），每条记录是一个元组（行）。

• 这种模型屏蔽了物理细节，以逻辑上二维表的形式呈现数据，使得声明式查询十分强大。

4. NULL 与三值逻辑

• 如果某个字段的值未知或不适用，可以使用 NULL。

• NULL 既不等于某个值，也不不等于某个值，可能导致条件运算为 UNKNOWN，需要特殊判断（如 IS NULL, IS NOT NULL）。

• 在 WHERE 子句中包含 NULL 会影响过滤逻辑，需要注意。

5. 子查询、SOME/ANY 与 ALL（如果系统支持）

• 可以在 WHERE 中嵌套另一个查询：

SELECT name

FROM instructor

WHERE salary > SOME(

SELECT salary

FROM instructor

WHERE department\_name = 'Comp. Sci.'

);

• SOME 或 ANY（不同 SQL 方言会有所区别）指只要满足条件集合中任意一条记录就返回 TRUE。

• ALL 要求对子查询返回的所有记录都满足条件才返回 TRUE。

6. 基础代码示例

下面给出一些常用的示例模板供参考。

1. 创建表（DDL）

CREATE TABLE instructor (

ID CHAR(5),

name VARCHAR(20) NOT NULL,

department\_name VARCHAR(20),

salary DECIMAL(8,2),

PRIMARY KEY (ID),

FOREIGN KEY (department\_name) REFERENCES department(dept\_name)

);

2. 插入记录

INSERT INTO instructor (ID, name, department\_name, salary)

VALUES ('22222', 'Smith', 'History', 45000);

3. 更新记录

UPDATE instructor

SET salary = salary + 1000

WHERE department\_name = 'Comp. Sci.';

4. 删除记录

DELETE FROM instructor

WHERE ID = '22222';

5. 连接查询

SELECT i.ID, i.name, t.course\_id

FROM instructor AS i

JOIN teaches AS t

ON i.ID = t.ID

WHERE i.department\_name = 'Comp. Sci.';

6. 使用子查询（若数据库支持）

SELECT name

FROM instructor

WHERE salary > (SELECT AVG(salary) FROM instructor);