**2025-3-5 COMPSCI751 Summary**

**英文原文 & 中文翻译**

1. Relational Model & Basic Concepts

English Version

• Primary Key: A unique identifier for each tuple in a relation (e.g., an ID attribute in the employee relation).

• Relations: Structured as tables with rows (tuples) and columns (attributes).

• Schema: The set of attributes (column names) defining a relation. Matching schemas is crucial for set operations (e.g., UNION).

• Tuple: A single row in the relation; each tuple has a value for every attribute.

Additional Notes/Extension

• In a well-designed database, each relation typically has at least one key (candidate key). One of these is chosen as the primary key.

• Other constraints often include foreign keys (to reference other tables) and unique constraints.

中文版本

• 主键（Primary Key）：在一个关系中用来唯一标识每一条元组的字段（例如，在 employee 表中用 ID 来作为主键）。

• 关系（Relations）：以二维表形式表现，行对应元组（tuple），列对应属性（attribute）。

• 模式（Schema）：定义关系的属性集合（即列的名称和数据类型），在做集合操作（如 UNION）时必须匹配模式才能执行。

• 元组（Tuple）：关系中的一行，包含该行在每个属性上的具体取值。

知识扩展

• 在一个设计良好的数据库中，每个关系通常至少有一个候选键（Candidate Key），其中一个被选作主键。

• 除了主键，数据库中还可能存在外键（Foreign Key）以及唯一约束等其它完整性约束，用于保证数据一致性和关联性。

2. Core Relational Algebra Operations

English Version

Relational Algebra includes several fundamental operations that allow us to query and manipulate relations in a formal way.

1. Selection (σ)

• Filters tuples in a relation based on a given condition.

• Example: σsalary>10000(Works) selects all tuples in Works where salary > 10000.

2. Projection (π)

• Extracts specific attributes (columns) from a relation.

• Example: πID, Name(Employee) returns only the ID and Name columns from the Employee relation.

3. Cross Product (× or ⨯)

• Combines every tuple of one relation with every tuple of another relation.

• Resulting relation’s schema is the union of the two relations’ attributes (with potential attribute renaming if conflicts occur).

• Number of resulting tuples = (# of tuples in R1) × (# of tuples in R2).

4. Natural Join (⋈)

• An equijoin that automatically matches tuples based on all attributes of the same name, and merges these common attributes into one.

• Example: Employee ⋈ Works, if both have an attribute ID, the resulting relation merges the ID column.

5. Theta-Join (θ-Join)

• A more general form of join where you specify a condition (θ) explicitly (e.g., “Employee.ID = Works.ID AND Works.salary > 9000”).

• Often written as σcondition(R1 × R2) or using the special join notation R1 ⋈condition R2.

Additional Notes/Extension

• Join vs. Cartesian Product: A join is essentially a Cartesian product followed by a selection condition; the natural join is a special case where the selection is based on matching all identically named attributes.

• Push-Down Selections: In query optimization, selections are often “pushed down” to reduce the size of intermediate results before performing expensive joins.

中文版本

关系代数提供了一系列用于查询与操作关系（表）的基础运算。

1. 选择（Selection，σ）

• 根据给定的条件从一个关系中筛选出符合条件的元组。

• 例子：σsalary>10000(Works) 表示从 Works 中选出薪水大于10000的所有元组。

2. 投影（Projection，π）

• 从一个关系中选取特定的属性（列）。

• 例子：πID, Name(Employee) 表示从 Employee 表中只选出 ID 和 Name 两列。

3. 笛卡儿积（Cross Product，× 或 ⨯）

• 将一个关系中的每条元组与另一个关系中的每条元组合并。

• 笛卡儿积结果的模式是两个关系属性的并集（若有属性名冲突，需要适当重命名）。

• 结果元组数 = R1的元组数 × R2的元组数。

4. 自然连接（Natural Join，⋈）

• 一种特殊的等值连接，会根据公共属性自动匹配，并将公共属性合并成一列。

• 例如：Employee ⋈ Works，当二者都含有 ID 属性时，结果只保留一个 ID 列。

5. θ连接（Theta-Join，θ-Join）

• 在连接条件（θ）中可以显式指定连接方式（例如：“Employee.ID = Works.ID AND Works.salary > 9000”）。

• 常写作 σ条件(R1 × R2) 或 R1 ⋈条件 R2。

知识扩展

• Join 与笛卡儿积：连接实际上是先做笛卡儿积，再用选择条件进行过滤；自然连接则是根据所有同名属性进行匹配的一种特殊连接。

• 选择下推（Push-Down Selections）：在查询优化中，为减少中间结果规模，通常会在进行昂贵的连接操作之前先下推选择操作。

3. Set Operations

English Version

• Union (∪): Combines tuples from two relations with the same schema, removing duplicates in standard set semantics.

• Intersection (∩): Returns tuples common to both relations. Often can be expressed using difference operators.

• Difference (−): Finds tuples in one relation that are not present in the other (with the same schema).

Additional Notes/Extension

• Schema Matching: For set operations, the relations must have the same set of attributes and compatible data types.

• Intersection via Difference: A ∩ B ≡ A − (A − B), which shows intersection is not strictly necessary as an operator (it can be derived).

中文版本

• 并（Union，∪）：把两个模式相同的关系中的元组合并起来，去重后得到结果。

• 交（Intersection，∩）：返回同时存在于两个关系中的元组。通常也可以通过差集来表示。

• 差（Difference，−）：在前一个关系出现而在后一个关系中没有出现的元组（前提是二者模式相同）。

知识扩展

• 模式匹配（Schema Matching）：进行集合操作时，两个关系必须具有相同的属性集，且属性类型兼容。

• 用差集表示交集：A ∩ B 可以写成 A − (A − B)，说明了在关系代数中，交集运算并非不可或缺。

4. Rename Operation (ρ)

English Version

• Purpose: To change the relation name or attribute names in a relation.

• Usage: ρnewRelationName(newAttr1, newAttr2, …)(OldRelation).

• Self-Join: Often used to distinguish two copies of the same relation in a self-join scenario.

中文版本

• 作用：给关系或关系中的属性重命名。

• 用法：ρ新关系名(新属性1, 新属性2, …)(旧关系)。

• 自连接（Self-Join）：在对同一个关系进行多次连接时，常用rename为不同的关系别名，方便区分。

5. Self-Join & Advanced Constraints

English Version

• Self-Join: A situation where the same relation is joined with itself.

• Example: Find pairs of employees in the same department. You need to rename the relation into two aliases (e.g., R1 and R2) and then match on the department attribute.

• Avoid duplicates (like pairing (Einstein, Einstein) or (Einstein, Gould) plus (Gould, Einstein)) by applying conditions such as R1.ID < R2.ID.

• Complex Constraints:

• Check constraints: e.g., to ensure a salary must be > 0.

• Referential Integrity: e.g., foreign key references must exist in the parent table.

中文版本

• 自连接（Self-Join）：当一个关系与自己进行连接时，就需要用到自连接。

• 例如，查找同一部门下的所有员工对：先将同一张表分别重命名为R1、R2，然后根据部门号或系名相同的条件进行连接。

• 可以用“R1.ID < R2.ID”来避免重复配对（如Einstein-Gould和Gould-Einstein）。

• 高级约束：

• 检查约束（Check constraints）：例如要求薪水必须大于0等。

• 参照完整性（Referential Integrity）：例如外键在父表中必须存在对应记录。

6. Assignment Operation

English Version

• Definition: Allows you to store the result of a relational algebra expression in a temporary relation.

• Usage: Temp1 := σ<sub>dept="Physics"</sub>(Instructor);

• Benefit: Makes complex queries more readable by breaking them down into smaller intermediate steps.

中文版本

• 定义：将某个关系代数表达式的结果赋给一个临时关系。

• 用法：Temp1 := σ<sub>dept="Physics"</sub>(Instructor);

• 好处：将复杂查询分解成多个简单步骤，代码或表达式更易读。

7. Expression Trees

English Version

• Concept: Visual representation of a relational algebra expression in a tree structure.

• Leaves: Relations (base tables).

• Internal Nodes: Operators (selection, join, projection, etc.).

• Pushdown Strategy: By examining the tree, we can move selection operators down the tree to reduce intermediate relation sizes.

中文版本

• 概念：将关系代数表达式用树形结构呈现的方式。

• 叶子节点：关系（基础表）。

• 内部节点：各种操作符（选择、连接、投影等）。

• 下推策略（Pushdown Strategy）：通过查看表达式树，可以将选择操作往叶子方向移动，以减少中间结果的规模。

8. SQL Implementation and Basic Syntax

English Version

Though this session focused on relational algebra, here is a brief introduction to equivalent SQL constructs:

1. Selection → WHERE clause

2. Projection → SELECT clause

3. Cross Product & Join → FROM table1, table2 or explicit JOIN ... ON ... syntax

4. Set Operations → UNION, INTERSECT, EXCEPT in SQL (note that SQL allows duplicates unless UNION DISTINCT, etc.)

SQL Example Template

-- Selecting employees with salary > 10000

SELECT ID, Name

FROM Works

WHERE Salary > 10000;

-- Natural Join equivalent (explicitly listing common attributes)

SELECT e.ID, e.Name, w.CompanyName

FROM Employee e

JOIN Works w ON e.ID = w.ID

WHERE w.CompanyName = 'BigBank';

中文版本

虽然本次课程主要内容是关系代数，但在SQL中可以找到与之对应的实现方式：

1. 选择(Selection) → WHERE 子句

2. 投影(Projection) → SELECT 子句

3. 笛卡儿积和连接 → FROM table1, table2 或者使用显式的 JOIN ... ON ... 语法

4. 集合操作 → SQL 中的 UNION, INTERSECT, EXCEPT（注意标准SQL中UNION默认去重，不同数据库对于是否去重有不同实现）

SQL 示例模板

-- 查找薪水大于10000的员工（Works表）

SELECT ID, Name

FROM Works

WHERE Salary > 10000;

-- 自然连接等价写法（显式指定公共属性）

SELECT e.ID, e.Name, w.CompanyName

FROM Employee e

JOIN Works w ON e.ID = w.ID

WHERE w.CompanyName = 'BigBank';

9. Additional Points & Best Practices

English Version

• Data Integrity and Constraints: Always ensure your design includes primary keys, foreign keys, and necessary constraints to maintain data consistency.

• Indexing: Large tables often require indexing (e.g., on the primary key, or frequently queried attributes) to speed up queries.

• Normalization: A fundamental design principle to reduce data redundancy (e.g., moving repeated attributes to different tables).

• Query Optimization: The RDBMS often transforms the SQL into an internal relational algebra expression tree and optimizes the order of joins, selections, and projections.

中文版本

• 数据完整性与约束：务必要确保数据库设计中包含主键、外键以及必要的约束，以维护数据的一致性。

• 索引（Indexing）：面对大型表时，通常需要在主键或常用查询字段上建立索引，以提高查询效率。

• 范式化（Normalization）：数据库设计的重要原则，用于减少数据冗余，例如将重复字段拆分到不同的表中。

• 查询优化（Query Optimization）：数据库系统通常会将SQL转换为内部的关系代数表达式树，再通过选择顺序、投影下推、连接顺序等策略进行优化。

10. Summary and Practice Advice

English Version

• Conceptual Mastery: Understand the formal foundations (relational algebra) behind SQL.

• Practice: Attempt to write both relational algebra queries and their equivalent SQL versions for better understanding.

• Exercise:

• Write a self-join to find pairs of employees in the same city.

• Practice set operations using UNION, INTERSECT, and EXCEPT.

• Attempt “pushdown” by rearranging your conditions to reduce result sizes early.

中文版本

• 概念掌握：理解SQL背后的理论基础（关系代数），这有助于深层次掌握数据库查询原理。

• 多加练习：尝试编写关系代数表达式并在SQL里实现对应的查询，可以加深理解。

• 练习示例：

• 写一个自连接（self-join），找出住在同一城市的员工对。

• 练习使用 UNION, INTERSECT, EXCEPT 等集合操作。

• 尝试将选择操作下推以减少连接前的结果规模。

Sample Relational Algebra & SQL Comparison

English Version

1. Relational Algebra

-- Rename for self-join

R1 := ρ(Employee1(ID1, Name1, City1, Salary1))(Employee);

R2 := ρ(Employee2(ID2, Name2, City2, Salary2))(Employee);

-- Find employee pairs in the same city

Result := σ(City1 = City2 AND ID1 < ID2)(R1 × R2);

2. SQL Equivalent

-- Self-join with aliases

SELECT e1.ID AS ID1, e1.Name AS Name1,

e2.ID AS ID2, e2.Name AS Name2

FROM Employee e1

JOIN Employee e2

ON e1.City = e2.City

AND e1.ID < e2.ID;

中文版本

1. 关系代数

-- 自连接重命名

R1 := ρ(Employee1(ID1, Name1, City1, Salary1))(Employee);

R2 := ρ(Employee2(ID2, Name2, City2, Salary2))(Employee);

-- 找出居住在同一城市的员工对（并避免重复自己）

Result := σ(City1 = City2 AND ID1 < ID2)(R1 × R2);

2. SQL 对应写法

-- 使用别名进行自连接

SELECT e1.ID AS ID1, e1.Name AS Name1,

e2.ID AS ID2, e2.Name AS Name2

FROM Employee e1

JOIN Employee e2

ON e1.City = e2.City

AND e1.ID < e2.ID;