数据库：SQL考试范围

- ## 初级
  - 数据定义 Data Definition
    - 数据种类
      - char(n)
      - varchar(n)
      - *nvarchar*
      - int
      - smallint
      - numeric(p,d)
      - real
      - float(n)
      - null
    - 基本架构 Basic Schema Definition
      - create table
        - *integrity constraints（完整性约束）
          - Primary key
          - Foreign key
          - NOT NULL
        - * integrity constraints（完整性约束）两种表达方式
          - 列级约束
          - 表级约束
      - 增
        - INSERT INTO 表（属性） VALUE(属性值...)
        - ALTER TABLE 表 ADD COLUMN 属性 类型
      - 删
        - Drop Table 表
        - Delete 表 WHERE 条件
        - ALTER TABLE 表 DROP COLUMN 属性
  - **查询基本结构 Basic structure of SQL Queries**
    - SQL -> 关系代数 -> 查询优化 -> 查询评价
    - 单一关系查询
    - 重命名

    - SELECT 属性 AS 命名 FROM 表
    - SELECT * FROM employees AS staff;
  - 字符串操作 **String Operations**
    - SELECT **CONCAT**('Hello', ' ', 'World');
    - SELECT **SUBSTRING**('Hello World', 7);
    - SELECT **LENGTH**('Hello World');
    - SELECT **LOWER**('Hello World');
    - SELECT **UPPER**('Hello World');
    - SELECT * FROM employees WHERE **name LIKE '%son%'**;
  - 排列 Ordering the Display of Tuples
    - SELECT * FROM employees ORDER BY salary **DESC**, name **ASC**;
  - Where子句
    - SELECT * FROM employees WHERE ***HireDate > '2019-01-01';***
    - SELECT * FROM employees WHERE *Salary* ***between*** *50000* ***and*** *55000;*
  - set操作
    - (SELECT * FROM table1 WHERE Department = 'Sales') **UNION** (SELECT * FROM table2 WHERE Department = 'Engineering');
    - (SELECT Department FROM table1) **INTERSECT** (SELECT Department FROM table2);
    - (SELECT Department FROM table1) **EXCEPT ALL** (SELECT Department FROM table2);
  - 关于NULL值
    - SELECT * FROM mytable WHERE **age IS NULL;**
    - SELECT * FROM mytable WHERE age IS **UNKNOWN;**
    - DISTINCT 处理NULL值很棘手
- **聚合函数 Aggregate Functions**
  - 基本 Basic Aggregation
    - SELECT name, **AVG**(Score) as avgscore FROM Students WHERE ...;
    - SELECT COUNT(DISTINCT ID) FROM Students WHERE XXX;

      > SQL不允许DISTINCT with COUNT(*)
  - 分组聚合 Aggregation with Grouping
    - SELECT product_id , SUM(sale_amount) as total_sales FROM sales **GROUP BY** product_id;

      > 每种产品的销售额
    - SELECT customer_id , COUNT(*) as num_sales FROM sales **GROUP BY customer_id**;
  - Having 子句
    - SELECT CustomerID , SUM(Total) AS OrderTotal FROM orders 3 GROUP BY CustomerID **HAVING** OrderTotal > 100
  - 有空值和布尔值的聚合 Aggregation with Null and Boolean Values

        - 聚合通常忽略NULL
        - SELECT COUNT(column1) FROM table

          > 查询所有值的数量，包括空值

        - SELECT AVG(column1) FROM table

          > NULL & unknown 在AVG时被忽略

    - 聚合和关系代数
- **Nested Subqueries 嵌套子查询**
    - Set Membership
        - 查询特定值是否存在 SELECT * FROM orders WHERE CustomerID IN (SELECT CustomerID FROM customers);

          > NOT IN也可以

    - Set Comparison
        - SELECT DISTINCT T.name FROM instructor AS T, instructor AS S WHERE T.salary > S.salary AND S.depLname = 'Biology'
        - SELECT name FROM instructor AS T WHERE salary > **SOME** (SELECT salary FROM instructor WHERE depLname = 'Biology';
        - SELECT name FROM instructor AS T WHERE salary > **ALL** (SELECT salary FROM instructor WHERE depLname = 'Biology';
        - SELECT name FROM instructor AS T WHERE salary > **NOT IN** (SELECT salary FROM instructor WHERE depLname = 'Biology';
    - 提取 Extract
        - 得到日期、时间的一部分，比如年月日小时
        - SELECT EXTRACT(YEAR FROM '2023-01-01') AS ExtractedYear;
    - Empty Relations测试
        - SELECT customer_id , name FROM customers **WHERE NOT EXISTS** (SELECT * FROM orders WHERE orders.customer_id = customers.customer_id);
        - SELECT customer_id , name FROM customers **WHERE EXISTS** (SELECT * FROM orders WHERE orders.customer_id = customers.customer_id);
    - From Clause
        - 查找所有薪资高于所有部门平均预算的教授：
        - SELECT I.NAME , I.SALARY **FROM (SELECT avg(BUDGET) AS avgbudget FROM DEPARTMENT) AS BUDGET**, **Instructor** AS I WHERE I.SALARY > BUDGET.avgbudget;
    - WITH子句
        - 允许定义一个可以在主查询中引用的临时结果集。
        - WITH alias AS (SELECT ...
    - 标量子查询Scaler Subqueries
- **数据库修改 Modification of the Database**

        - **INSERT INTO** employees (id, name , salary) VALUES (5678, 'John Doe', 45000);
        - **INSERT INTO** employees (*id, first_name*) **SELECT** *id, first_name* FROM people WHERE age > 30;
        - **DELETE** FROM employees WHERE salary < 40000;
        - **ALTER TABLE** orders **ADD FOREIGN KEY** (customer_id) REFERENCES customers(id) **ON DELETE CASCADE;**
        - **UPDATE** employees **SET** salary = 60000 WHERE id = 1234;
        - UPDATE employees SET salary = **CASE WHEN** department = 'Sales' **THEN** salary * 1.1 **WHEN** department = 'Marketing' **THEN** salary * 1.2 **ELSE** salary **END** WHERE id IN (1, 2, 3);
- 中级
- 高级
- 查询优化
    - **Measures of Query Cost**
    - **Selection Operation**
        - Selections Using File Scans and Indices
    - **Join Operation**
        - Nested-Loop Join 嵌套循环JOIN
        - Block Nested-Loop Join 块嵌套循环JOIN
        - Merge Join 合并JOIN
        - Complex Join 复杂JOIN
    - **Evaluation of Expressions**
        - Materialized
        - **Pipelining**
    - **Transformation of Relational Expressions**
        - 等价
            - 关系表达式
            - SQL
        - 等价规则
            - 联合选择操作可以被分解（分解）为一系列单独的选择。

              $$\sigma_{\theta_1 \wedge \theta_2}(E) = \sigma_{\theta_1}(\sigma_{\theta_2}(E))$$

            - 选择操作是可交换的

              $$\sigma_{\theta_2}(\sigma_{\theta_1}(E)) = \sigma_{\theta_1}(\sigma_{\theta_2}(E))$$

            - 只需要最后一个投影操作序列，其他操作可以省略

>> > $$\Pi_{L_1}(\Pi_{L_2}(\cdots\Pi_{L_n}(E)\cdots)) = \Pi_{L_1}(E)$$
>> >
>> - 选择可以与笛卡尔积和theta联接结合使用
>>
>> > $\sigma_\theta(E_1 \times E_2) = E_1 \bowtie_\theta E_2$, $\sigma_{\theta_1}(E_1 \bowtie_{\theta_2} E_2) = E_1 \bowtie_{\theta_1 \wedge \theta_2} E_2$
>>
>> - 还有很多
>
> - 尽早执行选择，投影可以减少要连接的关系的大小。