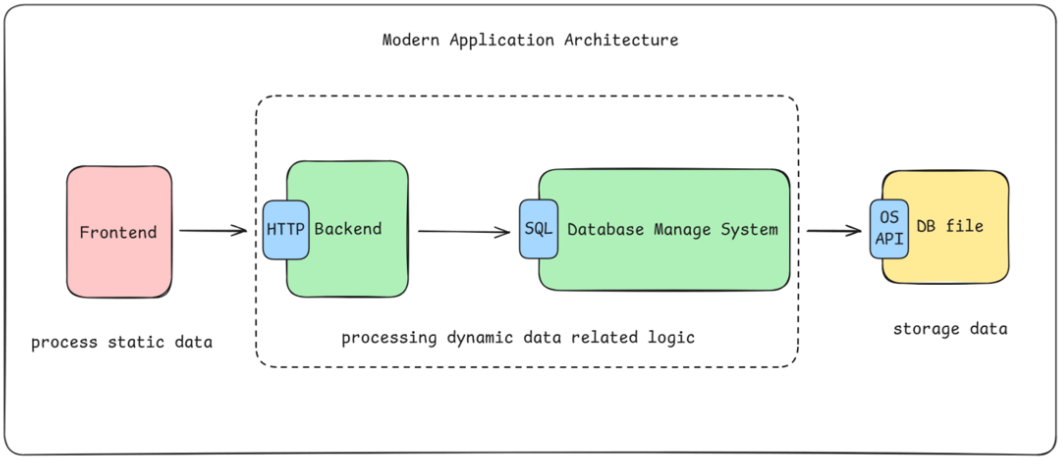
SQL

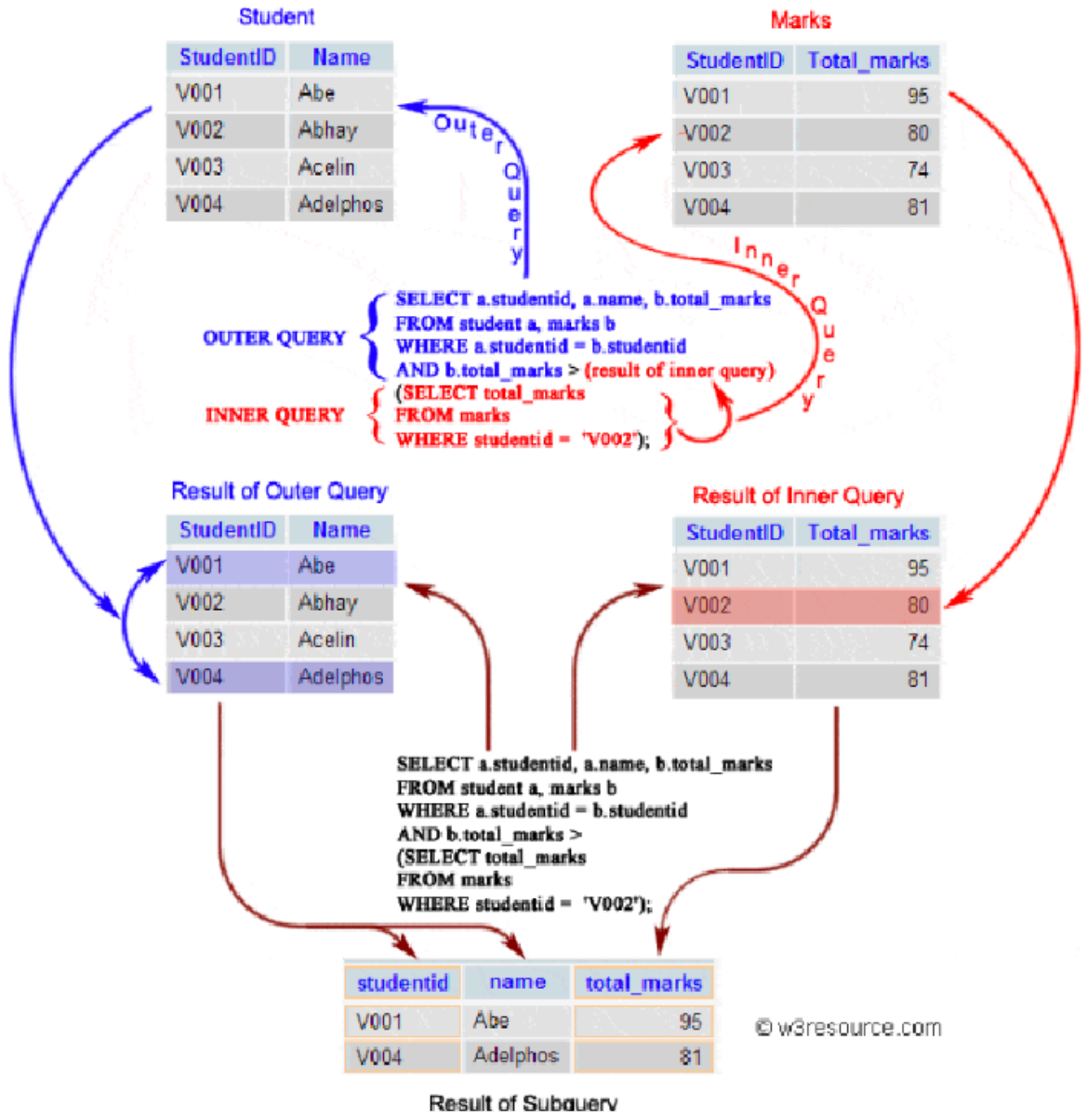
我简单理了一份关于SQL的notes，但是并不是很全，建议大家还是去读一下我找到的优质notes

- SQL Part 1 - Basic Queries - Database Systems 186的前两小节
- SQL 语言介绍 - HobbitQia的笔记本 学长笔记中的SQL语言介绍，中级SQL两小节，高级SQL基本不考
- SQL语法基础知识总结 | JavaGuide 面经文档，里面有一些练习题可以做一下

SQL是用户为了通过数据库管理系统来操作数据库文件而产生的语言，如下图是基础的前后端应用的逻辑构图



在SQL的视角下，数据库是由一系列表table组成的，例如下图，这些表之间一般存在一定的联系



1. SQL的类别

SQL中的语句分为以下三种

- DDL 数据定义语言：表的定义和定义的修改，数据完整性的设置 create, alter, drop，同时还包括定义视图和索引
- DML 数据操作语言：表数据的增删查改 select, insert, delete, update
- DCL 数据控制语言：现代数据库管理系统支持多用户，不同用户对表由不同的操作权限，DCL是关于数据权限的修改 Grant, revoke

在考试之中，重要性是 DML > DDL > DCL

2. DDL

考试中考察DDL这块的核心内容就是表相关操作，主要是创建表

2.1 表的增删查改

创建表

表 table 的概念是数据库的核心，关系型数据库的主体内容是由一系列的表组成，而DDL语句就是根据我们的需求，创建一个个表，例如

```sql
create table `book` (
    `book_id` int not null auto_increment,
    `category` varchar(63) not null,
    `title` varchar(63) not null,
    `press` varchar(63) not null,
    `publish_year` int not null,
    `author` varchar(63) not null,
    `price` decimal(7, 2) not null default 0.00,
    `stock` int not null default 0,
    primary key (`book_id`),
    unique (`category`, `press`, `author`, `title`, `publish_year`)
)
```

我们在创建表的时候要考虑到以下内容

- 这个表的名字 book
- 这个表拥有的属性 book_id, category, title ...
- 这个表的各个属性的类别 int, varchar ...
- 这个表的各个属性的完成性约束（**Integrity Constraints**）not null, auto_increment

由于考试中是可以携带A4的，因此这里的各个属性和完整性约束的具体含义我就不描述了，这里只讲宏观的理解。

大家应该都学过数据结构或者面向对象程序设计，因此我们要设计一个数据对象的时候，除了描述它里面应该有什么属性以外，例如存储图书需要有图书的名字和图书的价格之类的，还需要有属性的类型，其实不论是**类型还是完整性约束**，都是对于这个属性的一种约束，描述了设计者对于该属性的需求，例如：

- 书本的存量是 int，因为不能存在 1.5 本书这个概念
- 书本的名称是 not null，因为我需要根据书本的名称来定位到这本书

删除表

- `drop table <table_name>` 删除整个表
- `delete from <table_name>` 删除整个表里面的所有内容，表依然存在，只是里面的记录为空

比较简单不做赘述

修改表

- `ALTER TABLE <table_name> ADD <column_name> <data_type> [constraints];` 新增一条属性
- `ALTER TABLE <table_name> MODIFY [COLUMN] <column_name> <new_data_type> [new_constraints];` 修改一条属性
- `ALTER TABLE <table_name> DROP [COLUMN] <column_name>;` 删除一条属性

比较简单不做赘述，考试也不怎么考，记几个例子到A4上就行

2.2 视图的增删查改

视图（View）是一个虚拟表，其是基于现有的表存在的，例如一个表存储用户信息，我公司只有部分核心员工可以访问完整这个表，其他员工可能只能访问不带用户密码的用户信息，因此我们可以设计一个视图，让非核心员工只能访问该视图。

> 需要注意的事，视图本身不存储数据，新增一个视图并不会让我们把表的数据复制一份。

例如

```
CREATE VIEW safe_user_access AS
    SELECT id, name, email
    FROM users
```

视图基于一个SELECT语句存在，简单来说就是把这个SELECT语句保存下来了，部分人访问的时候直接经过这个SELECT

删除视图

```
DROP VIEW safe_user_access;
```

视图的操作同样考察不多

2.3 索引的增删查改

索引的目的在于快速访问，其会基于表的某个或几个属性，构建一个索引，以后基于该属性的访问就会非常快，例如

```
CREATE INDEX idx_name ON users (name);
```

以后如果我们需要通过name来获取该user的密码，就会非常快速

删除索引

```
DROP INDEX idx_name ON users;
```

3. DML

DML是考试重点和难点，考试中一般会有一道大题，给你一个数据库，里面有一些表，让你根据特定的需求编写SQL语句。但是就经验而言，只有最后两分的小题会涉及较为复杂的语句，大头都是很基础的。并且由于SQL本身比较灵活，达到同一个的目的的写法比较多样，不需要强求一定要和标准答案写的一模一样。

其中在DML里面的SELECT是最重要的

3.1 select

关系型数据库由表（relations/tables）组成，每个表包含行（tuples）和列（attributes）。例如，一个Person表可能如下：

name	age	num_dogs
Ace	20	4
Ada	18	3

name	age	num_dogs
Ben	7	2
Cho	27	3

SQL通过DML语句操作这些数据，SELECT是其中最重要的命令，用于检索数据。

基本查询

SELECT语句用于从表中选择特定列，语法为：`SELECT <columns> FROM <table>;`

例如，从Person表中选择name和num_dogs：

```
SELECT name, num_dogs FROM Person;
```

结果为：

name	num_dogs
Ace	4
Ada	3
Ben	2
Cho	3

可以使用DISTINCT去除重复行：`SELECT DISTINCT name, num_dogs FROM Person;`

条件查询

WHERE子句用于根据条件筛选行，语法为：`SELECT <columns> FROM <table> WHERE <predicate>;`

例如，仅选择年龄大于等于18的记录：

```
SELECT name, num_dogs FROM Person WHERE age >= 18;
```

结果为：

name	num_dogs
Ace	4

name	num_dogs
Ada	3
Cho	3

布尔运算符如NOT、AND、OR可用于更复杂的条件：`SELECT name, num_dogs FROM Person WHERE age >= 18 AND num_dogs > 3;`

聚合分组

聚合函数用于汇总数据，包括SUM、AVG、MAX、MIN和COUNT。例如，计算平均年龄：

```
SELECT AVG(age) FROM Person;
```

GROUP BY用于按列分组后，其会自动将这个属性相同的行放在一起，然后应用聚合函数，例如按部门计算平均薪资：

```
SELECT department, AVG(salary) FROM Employees GROUP BY department;
```

连接

连接用于合并多个表的数据，很多时候我们要基于多个表获取查询结果，连接的方式有很多种

- 交叉连接（Cross Join）交叉连接生成两个表的笛卡尔积，例如：

```
SELECT * FROM courses, enrollment;
```

- 内连接（Inner Join）内连接仅返回匹配的行，语法为：

```
SELECT * FROM courses INNER JOIN enrollment ON courses.num =
enrollment.c_num;
```

- 外连接（Outer Joins）外连接确保保留一个表的所有行，未匹配的行用NULL填充。

```
SELECT * FROM courses LEFT OUTER JOIN enrollment ON courses.num = enrollment.c_num;  左外连接
SELECT * FROM enrollment RIGHT OUTER JOIN courses ON courses.num = enrollment.c_num; 右外连接
SELECT * FROM courses FULL OUTER JOIN enrollment ON courses.num = enrollment.c_num;  全外连接
```

子查询

有时我们无法在一个查询语句里面表述清楚逻辑，因此我需要在里面嵌套一个子查询，子查询可用于WHERE子句。例如，查找学生数高于平均值的课程：

```
SELECT num FROM enrollment WHERE students >= (SELECT AVG(students) FROM enrollment);
```

同时为了防止多层的嵌套使得可读性严重下降，我们可以用WITH语句给子查询一个名称，with语句可以同时定义多个

```
WITH courseEnrollment(num1, name, students) AS (
    SELECT c.num AS num1, c.name, e.num AS num2, e.students
    FROM courses AS c INNER JOIN enrollment AS e
    ON c.num = e.num;
)
SELECT num1, name, students
FROM courseEnrollment
WHERE students > 700;

```

集合操作

- union 并集，就是将两个属性相同的表加在一块 union all 保持重复组

```
SELECT student_id, name
FROM students_a
UNION
SELECT student_id, name
FROM students_b;
```

- except 删除集合中的反例，代表关系代数中的“-”在取反操作中很常见

```
SELECT student_id, name
FROM students_a
WHERE student_id NOT IN (
    SELECT student_id
    FROM students_b
);
```

- intersect 交集，取相同的部分

```
SELECT student_id, name
FROM students_a
INTERSECT
SELECT student_id, name
FROM students_b;
```

some

some 是指某个的意思，指代后续集合中的任意一个值

```
select *
from section
where id > some (select id ...)
```

这里只要我们的id大于后面这个子查询产生的id列中的任意一个值即可

all

与some相对应，做法完全相同，要求id大于所有子查询中的id

exists 和 not exists

```
select course_id
from section AS S    注意，这里的S可以在子查询里使用，与子查询里的section区分
where semester='Falls' and year = 2017 and exists(子查询)
```

注意子查询使用的S已经经过前面的筛选，子查询不为空即可

总结

SQL语句中有大量细节语法，但是我们并不需要对每个语法都做到灵活应用，在考试中，学会基础的join，筛选，子查询，已经可以解决大部分问题了

3.2 insert

基础的插入操作

```
INSERT INTO users (name, email, age)
VALUES ('张三', 'zhangsan@example.com', 25);
```

3.3 delete

基础的删除操作

```
delete from  r
where p
```

例如删除工资在1300-1500之内的教师

```
delete from instuctor
where salary between 1300 and 1500;
```

3.4 update

基础的更新操作

```
UPDATE users
SET name = '李四光', age = 35
WHERE id = 2;
```

case语句

case语句可以构成分段的操作，适用于update语句，例如

```sql
update works
set salary = case
    when salary <= 100000 then salary * 1.1
    else salary * 1.03
    end
where company_name = "First Bank Cororation" and ID = some (select
ID from manages)
```

3.5 特殊情形

关于NULL

SQL中的NULL表示未知或缺失值，需特别注意。任何与NULL的操作结果为NULL，例如x + NULL为NULL。检查NULL使用`IS NULL`或`IS NOT NULL`，例如：

```sql
SELECT name, num_dogs FROM Person WHERE age IS NOT NULL;
```

4. DCL

DCL在考试中基本没有出现过，而且不同的数据库管理系统在这块做的差异很大，因此基本不会考，知道有这么个东西即可

- `GRANT SELECT ON mydb.users TO 'app_user'@'localhost';` 赋予权限给某个用户
- `REVOKE SELECT ON mydb.users FROM 'app_user'@'localhost';` 撤销权限给某个用户