数据库复习总结

过程考核成绩:93(过程考核:93) 期末成绩:74 总评84 时间2024年

感觉多扣分了，是为数不多分数极低的科目

数据库考完不久，是HE老师出的题,全英文，可请求老师翻译。题量对于我来说很大，我没答完题以下是我复习所用资料。

题型范围如下

事务，安全性，完整性，视图，索引，关系代数，sql查询，视图查询，建立触发器，嵌入式sql，最小函数依赖集，建立er图

本次考试（2024年冬）未涉及事务处理

本次考试题目较多，仍然是全英文（可以举手询问老师翻译）

关系代数共5题，SQL语言共15题

嵌入式SQL，存储过程，ER图，最小函数依赖集（包括候选键判断）一个大题，还有个题是题目给出表格，画出经过关系运算后的表格。

往届题目

Answer the following questions, (20 = 4 Scores * 5)

1. What does integrity mean? And what kinds of integrity are there in a database?
2. What is the concept of a data model? What does it consist of?
3. What is a database transaction? What are the properties of a database transaction?
4. What are the characteristics of an index? Compare the difference between an index and a cluster.
5. Which methods can be used to solve the problem of data transferring between embedded SQL and host languages?

回答以下问题，（20 = 4 分 * 5）

1. 什么是完整性？数据库中有哪些完整性类型？
2. 数据模型的概念是什么？它由哪些部分组成？

3. 什么是数据库事务？数据库事务有哪些特性？
4. 索引有哪些特点？比较索引和聚簇的区别。
5. 哪些方法可以用来解决嵌入式 SQL 与宿主语言之间的数据传输问题？

Answer the following questions, (21 = 3 Scores * 6)

The CAP database is described as follows:

C: CUSTOMERS, A: AGENTS, P: PRODUCTS, O: ORDERS

Table CUSTOMERS (cid, cname, city, discnt)

Table PRODUCTS (pid, pname, city, quant, price)

Table AGENTS (aid, name, city, percent)

Table ORDERS (ordno, month, cid, aid, pid, qo, dollars)

The following queries involve the CAP database. You are required to solve the queries using relational algebra:

1. Find all (customer name, product name) pairs for orders of quantity equal to or greater than 650.
2. Find all the names of customers in Dallas who placed orders for product p15.
3. Get the product names ordered by at least one customer through an agent based in York.
4. Find the set of products ordered by c002 but not ordered by c006.
5. Get the names of customers who ordered at least one product costing less than $2.
6. Find all the cids of customers who have placed orders for all products priced at 20.

回答以下问题，（21 = 3 分 * 6）

CAP 数据库描述如下：

C：CUSTOMERS，A：AGENTS，P：PRODUCTS，O：ORDERS

表 CUSTOMERS (cid, cname, city, discnt)

表 PRODUCTS (pid, pname, city, quant, price)

表 AGENTS (aid, name, city, percent)

表 ORDERS (ordno, month, cid, aid, pid, qo, dollars)

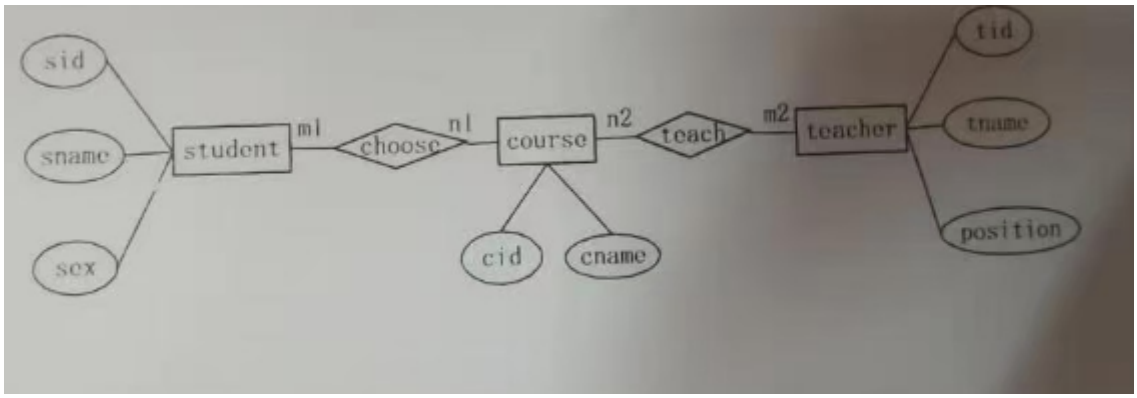
以下查询涉及 CAP 数据库，要求使用关系代数解决查询问题：

1. 找出所有订单数量大于等于 650 的 (客户名, 产品名) 对。
2. 找出在 Dallas 下订单购买产品 p15 的所有客户的名字。
3. 获取通过位于 York 的代理下单的至少一位客户所订购的产品名称。
4. 找出 c002 订购但 c006 未订购的产品集合。
5. 找出订购至少一件价格小于 $2 产品的客户姓名。
6. 找出下单购买了所有价格等于 20 产品的客户 cid。

4. (10 Score)

The E-R diagram describes the Course-Choosing process in a school. Please design tables based on the Entities and Relationships.

1. Discuss the outlines of your design, including Tables, Attributes, and Candidate Keys.
2. Transform the E-R model into tables and use Foreign Keys to maintain referential integrity.
3. (10 分)



E-R 图描述了学校中的选课场景。请根据实体和关系设计表格。

1. 讨论设计概要，包括表格、属性和候选键。
2. 将 E-R 模型转换为表格，并使用外键维护引用完整性。

5. (15 Scores)

Given the SQL statement to do the following tasks:

1. Create a table called `Students` with the following three columns:
   - `sno` (student number, type `INT`)
   - `sname` (student name, type `VARCHAR(13)`)
   - `ssex` (sex of the student, type `CHAR(2)`)
   - `roomno` (student dormitory number, type `CHAR(3)`)
2. Insert your own information as a row into this table.
3. Create a view from the `Students` table that includes the `sno`, `ssex`, and `roomno` for students whose name begins with "Na".
4. Grant privileges on the `Students` table to the user "Susan", allowing her to select and delete data, and only update the `roomno` column.
5. Calculate the number of students in each dormitory.
6. (15 分)

给定 SQL 语句，完成以下任务：

1. 创建一个名为 `Students` 的表，包含以下三个列：
   - `sno` (学生编号，类型为 `INT`)
   - `sname` (学生姓名，类型为 `VARCHAR(13)`)
   - `ssex` (学生性别，类型为 `CHAR(2)`)
   - `roomno` (学生宿舍编号，类型为 `CHAR(3)`)
2. 将你自己的信息作为一行插入到这个表中。
3. 从 `Students` 表中创建一个视图，包含学生编号、性别和宿舍编号，条件是学生姓名以 "Na" 开头。
4. 给用户 "Susan" 授予对 `Students` 表的权限，允许她选择和删除数据，并且只能更新 `roomno` 列。
5. 计算每个宿舍的学生人数。

(10 Scores)

Given a relation schema Head() = A, B, C, D, and a functional dependency set G:

G = {AB → C, C → D, D → A}

1. Find the minimum cover set G.
2. Find all the candidate keys for the table.

(10 分)

给定一个关系模式头为 Head() = A, B, C, D，且有一个函数依赖集 G：

G = {AB → C, C → D, D → A}

1. 找到最小覆盖集 G。
2. 找出该表的所有候选键。

(10 Scores)

In the CAP database, perform the following task: Input the pname of a product in a loop, then update its price based on user input.

1. Provide a flowchart for the processing steps of this task.
2. Write a C program with embedded SQL to implement this task.

(10 分)

在 CAP 数据库中，完成以下任务：在循环中输入产品的 pname，然后通过用户输入更新其价格。

1. 给出此任务的流程图 (Flowchart)。
2. 编写一个包含嵌入式 SQL 的 C 程序来实现此任务。

以下是对问题的逐一解答：

1. **什么是完整性？数据库中有哪些完整性类型？**

完整性是指数据库中数据的正确性和一致性。它是数据库必须遵守的一组约束条件，用于保证数据的有效性和逻辑合理性。

数据库完整性类型：

1. **实体完整性：**
   - 确保表中的每一行（实体）都是唯一可识别的。
   - 实现方式：通过主键约束（`PRIMARY KEY`）。
2. **参照完整性：**
   - 保证表与表之间的关系数据的一致性。
   - 实现方式：通过外键约束（`FOREIGN KEY`）。
3. **域完整性：**
   - 确保表中某一列的数据取值范围是有效的。
   - 实现方式：通过数据类型、长度限制和检查约束（`CHECK`）。
4. **用户定义完整性：**
   - 特定应用程序的业务规则，用来满足特殊需求。
   - 实现方式：通过触发器或存储过程。

---

5. **数据模型的概念是什么？它由哪些部分组成？**

数据模型是对现实世界数据特性及其关系的抽象描述，是数据库系统设计和实现的基础。

组成部分：

1. **数据结构：**
   - 描述数据的组织形式（如表、记录、字段）。
2. **数据操作：**
   - 定义对数据的操作方式（如插入、删除、更新、查询）。
3. **数据完整性约束：**
   - 定义数据的约束规则（如主键、外键、唯一性）。

---

4. **什么是数据库事务？数据库事务有哪些特性？**

数据库事务是一个操作序列，这些操作作为一个整体，要么全部执行成功，要么全部失败回滚。事务用于保证数据库在并发访问和故障情况下的数据一致性。

事务的四大特性（ACID）：

1. **原子性（Atomicity）：**
   - 事务是不可分割的最小操作单元，全部成功或全部失败。
2. **一致性（Consistency）：**
   - 事务执行前后，数据库必须处于一致状态。

3. **隔离性（Isolation）：**
    - 一个事务的执行不受其他事务的干扰。
4. **持久性（Durability）：**
    - 事务一旦提交，其对数据库的更改是永久的。

---

5. **索引有哪些特点？比较索引和聚簇的区别。**

索引的特点：

1. 提高查询效率。
2. 占用存储空间。
3. 对插入、更新和删除操作会增加额外的开销。

索引和聚簇索引的区别：

1. **数据存储方式：**
    - 普通索引：索引和数据分开存储，索引仅保存数据的指针。
    - 聚簇索引：索引和数据存储在一起，数据按照索引的顺序存储。
2. **性能：**
    - 普通索引适合范围较大的查询。
    - 聚簇索引适合范围查询和排序操作。
3. **表的限制：**
    - 一张表可以有多个普通索引。
    - 一张表只能有一个聚簇索引。

---

4. **哪些方法可以用来解决嵌入式 SQL 与宿主语言之间的数据传输问题？**

解决嵌入式 SQL 与宿主语言数据传输问题的方法：

1. **主变量（Host Variable）：**
    - 使用主变量在宿主语言中定义，传递到 SQL 中。
    - 例如：`EXEC SQL BEGIN DECLARE SECTION` 用于声明变量。
2. **游标（Cursor）：**
    - 用于在宿主语言中处理 SQL 查询返回的多行数据。
3. **动态 SQL：**
    - 使用 `PREPARE` 和 `EXECUTE` 语句动态构造和执行 SQL。
4. **数据类型映射：**

- 在嵌入式 SQL 和宿主语言之间，定义数据类型的转换规则。

5. **错误处理机制：**
   - 使用 `SQLCODE` 或 `SQLSTATE` 检查操作是否成功，确保数据传输的正确性。

触发器的定义

触发器（Trigger）是一种特殊的存储过程，在对表执行 **INSERT**、**UPDATE** 或 **DELETE** 操作时自动触发执行。触发器依附于表，用于实现对数据变动的自动响应。

触发器由数据库管理系统定义，用户无法直接调用或执行，而是由数据库在特定事件发生时自动触发。

触发器的作用

1. **自动执行逻辑**
   实现对表操作的自动化处理，无需手动调用。例如，记录日志、统计变化数据等。
2. **数据完整性保障**
   确保数据的完整性和一致性，例如防止非法数据插入，或在一张表数据改变时自动更新相关表数据。
3. **业务规则实现**
   在数据库层面实现复杂的业务规则，例如某些字段更新后触发通知或计算其他字段。
4. **数据维护**
   在主表数据删除时，自动删除相关表数据（级联删除），避免孤立数据。
5. **日志记录**
   自动记录对表数据的插入、更新和删除操作，为审计和分析提供支持。

触发器的分类

1. **按触发事件分类**
   - **INSERT 触发器**：在插入新记录时触发。
   - **UPDATE 触发器**：在更新记录时触发。
   - **DELETE 触发器**：在删除记录时触发。
2. **按触发时间分类**
   - **BEFORE 触发器**：在操作执行前触发。
   - **AFTER 触发器**：在操作执行后触发

一段完整的嵌入式sql解析（来自KONG神PPT）

已知 students表，编程查询某系全体学生的信息，系名由用户在程序运行时输入，根据用户要求修改其中某些学生的年龄。

```
#define TRUE 1  // 定义常量 TRUE 为 1，用于控制无限循环
#include <stdio.h>  // 标准输入输出头文件，提供 printf 和 scanf 等函数
#include "prompt.h"  // 自定义头文件，假设包含 prompt 函数，用于用户输入
exec sql include sqlca;  // 引入嵌入式 SQL 通信区，用于捕获 SQL 语句的执行结果状态

// 开始声明嵌入式 SQL 变量
exec sql begin declare section;
    char hdept[10];  // 存储用户输入的系名
    char hsno[4];    // 存储学生学号
    char hname[10];  // 存储学生姓名
    char hsex[2];    // 存储学生性别
    int hage;        // 存储学生年龄
exec sql end declare section;  // 结束嵌入式 SQL 变量声明

int main() {
    char msg[] = "Please input the department name: ";  // 提示用户输入系名的消息

    // 定义游标 p_student，用于查询学生信息，支持对 sage 字段的更新
    exec sql declare p_student cursor for
        select sno, sname, ssex, sage, sdepartment
        from students
        where sdepartment = :hdept
        for update of sage;  // 指定游标可以更新 sage 字段

    // 设置 SQL 错误处理条件
    exec sql whenever sqlerror goto report_error;  // 如果出现 SQL 错误，则跳转到 report_error 标签处理
    exec sql whenever not found goto finish;  // 如果未找到记录，则跳转到 finish 标签处理

    // 循环处理用户输入的系名
```

```
    while (prompt(msg, 1, hdept, 10) > 0) {  // 调用 prompt 函数获取用户输入的系名，返回值大于 0 表示成功
        exec sql connect to testdb;  // 连接到名为 testdb 的数据库

        exec sql open p_student;  // 打开游标，准备查询

        while (TRUE) {  // 无限循环处理游标查询结果
            // 从游标中获取一条记录，将结果存入主机变量
            exec sql fetch p_student into :hsno, :hname, :hsex, :hage;

            if (sqlca.sqlcode == success) {  // 如果 SQL 操作成功
                printf("%s %s %s %d\n", hsno, hname, hsex, hage);  // 输出学生信息
                printf("Update the age of this record? ");  // 提示用户是否更新年龄
                char ans;
                scanf(" %c", &ans);  // 获取用户输入的选择
                if (ans == 'y') {  // 如果用户选择更新
                    printf("Input the new age: ");  // 提示输入新年龄
                    scanf("%d", &hage);  // 获取用户输入的新年龄

                    // 更新当前游标所指记录的年龄字段
                    exec sql update students
                    set sage = :hage
                    where current of p_student;  // 使用游标更新
                }
            } else {
                break;  // 如果没有更多记录，退出循环
            }
        }

    finish:  // 结束标签
        exec sql close p_student;  // 关闭游标
        exec sql commit work;  // 提交事务，保存所有更改
    }  // 结束 while 循环（用户输入）

    exec sql disconnect current;  // 断开与数据库的连接
    return 0;  // 正常退出程序

report_error:  // SQL 错误处理标签
    print_dberror();  // 调用自定义函数打印数据库错误信息
    exec sql rollback work;  // 回滚事务，撤销所有未提交的更改
    exec sql disconnect current;  // 断开与数据库的连接
    return 1;  // 返回错误状态
}

```

游标的定义和使用

游标的定义

游标（Cursor）是数据库系统中的一种机制，用于在 SQL 结果集中逐行检索数据。游标可以看作是一个指针，指向查询结果集中的某一行，便于对数据进行逐行操作（如读取、更新或删除）。

游标适用于需要逐行处理的操作，尤其是在无法通过单一 SQL 语句实现复杂逻辑时。

游标的使用步骤

声明游标

使用 DECLARE 声明游标，并指定其对应的 SQL 查询。

查询可以包含 SELECT 语句及相关条件。

打开游标

使用 OPEN 打开游标，执行查询并建立结果集。

此步骤将游标与查询结果集绑定。

提取数据

使用 FETCH 从游标中获取一行数据。

每次调用 FETCH，游标会指向结果集中的下一行。

处理数据

根据业务逻辑，对提取的数据进行操作，如打印、计算或更新等。

关闭游标

使用 CLOSE 关闭游标，释放相关资源。

游标的基本语法

以下是游标的通用操作及其 SQL 语法（以嵌入式 SQL 为例）：

声明游标

```
EXEC SQL DECLARE cursor_name CURSOR FOR
SELECT column1, column2
FROM table_name
WHERE condition;
```

打开游标

```
EXEC SQL OPEN cursor_name;
```

提取数据

```
EXEC SQL FETCH cursor_name INTO :host_variable1, :host_variable2;
```

关闭游标

```
EXEC SQL CLOSE cursor_name;
```

游标示例代码

以下是一个嵌入式 SQL 使用游标的完整示例：

```
#include <stdio.h>
#include <string.h>
exec sql include sqlca;

exec sql begin declare section;
    char department[20];  // 用户输入的系名
    char student_id[10];  // 学生学号
```

```
    char student_name[50];  // 学生姓名
    int student_age;  // 学生年龄
exec sql end declare section;

int main() {
    // 提示用户输入系名
    printf("Enter department name: ");
    scanf("%s", department);

    // 声明游标
    exec sql declare student_cursor cursor for
    select id, name, age
    from students
    where department = :department;

    // 打开游标
    exec sql open student_cursor;

    // 提取数据
    while (1) {
        exec sql fetch student_cursor into :student_id, :student_name,
:student_age;
        if (sqlca.sqlcode != 0) break;  // 如果没有更多行，退出循环

        // 打印学生信息
        printf("ID: %s, Name: %s, Age: %d\n", student_id, student_name,
student_age);
    }

    // 关闭游标
    exec sql close student_cursor;

    return 0;
}
```

游标的优缺点

优点:

逐行处理：支持对查询结果集的逐行操作，适合复杂的业务逻辑。

动态更新：可以结合 `FOR UPDATE` 和 `WHERE CURRENT OF` 进行动态更新或删除。

缺点:

性能较低：相比批量操作，逐行处理速度较慢。

资源消耗高：需要额外的内存和资源来维护游标的状态。

动态游标

动态游标允许在程序运行时动态改变游标的查询条件，从而灵活处理不同的数据集。结合输入参数或条件，动态游标通常通过编程实现灵活性。

自己总结的涵盖了所有SQL语句的知识点

下面是我自己总结的涵盖了所有SQL语句的知识点，其实语句并不多，半小时就能全部背过，可惜考试的时候题目太多了，又有些英文我看不懂，感觉全卷没有难度过高的题目，直至最后也没写完题目，有部分错误（语义错误，无语法错误）懒得改了，其中不难发现，很多人以为很难的嵌入式SQL可以套模板

建表

```
Creat table Orders(
    oid char(5) primer key,
    :
    // primer key( oid , zid),
    :
    foreign key (cid) references cro(cid)
);
```

删表

```
drop table [cascade];
```

改表

```
alter table orders
add column oname char(6) not null          加列
add ~~forei~~foreign key (pid) references pro(pid) 加约束
drop column pid cascade                    删列
drop constraint 约束名 cascade             删约束
alter column oid char(6);
```

查表

```
select ~~cid~~ distinct cid, aid ...
form o, c...
where w.xid in( . . . . )
group by cid
having sum(sid)>5
order by ~~count(*)~~ dis asc/desc;
```

插记录

```
insert into sc (cid, aid)
values ('001', '001'),
       ('003', '004'),
       ('005', '001');
```

字符匹配

like ~~notlike~~ not like

% 0~∞

_ 1

excape '\' 表示 \ 为转义

查询条件

between and , not between and

is null , is not null

in , not in

exists , not exists

插入 2.

列表 creat table ss (cid, ~~pid~~ char(4) primerkey (cid, pid) , pid char(5));

insert into ss (cid, pid) selet cid, Avg(grade) } 插入查询结果至新表.

改记录

update sc set ~~sid~~ sname = '张' where sno = '...';

多改 update sc set sage = sage+1;

删除 delete from sc where ...;

delete from sc; 删除所有

嵌入式 SQL.

```
#include "prompt.h"
exec sql include sqlca;
exec sql begin declare section;
  char csno[10];
    :   ;
        ;
exec sql end declare section;
int main()
   char tishi[] = "请输入~~";

   exec sql declare cursorp cursor for
     select csno, ...
     from ...
     ...
     for update [of 属性名];

   exec sql whenever sqlerror goto error;
   exec sql whenever not found goto finish;

   while(prompt(tishi, 1, 回车符, 10)>0){
      exec sql connect to testdb;
      exec sql open cursorp;
      while(true)
        exec sql fetch cursorp into ->主变量;...;
```

流程：
1. 头文件
2. sqlca
3. 主变量区
4. 入主函数
5. 定义游标
6. 开启检错
7. 入大循环
8. 开库开表
9. 小循环读取
10. sqlcode使用
11. 更新用游标
12. 无数据
13. 断库
14. 结束

结束 ← 12. 无数据

结束 回滚 断库

```
if (sqlca.sqlcode == success){
    printf("%d, ..          );
    printf("是否 -- ?");
    char ans;
    scanf("%c", &ans);
    if (ans == 'y')
    printf("输入 -- ");

        exec sql upde 表名
        set age = :newage
        where current of cursorp;
    }
}else{
    break;
}}  → 小 while
finish:
    exec sql close cursorp;
    exec sql commit work;
} → 大 while
    exec sql disconnect current;
    return 0;
error:
    print("  -  ~  );
    exec sql rollback work;
    exec sql disconnect current;
    return 1;
}
```

```
slock
Xlock
wait
```

触发器

```
create or replace taigger chufa
after / before
insert / update / delete
on sc
for each row
begin
    ...
end;
/
```

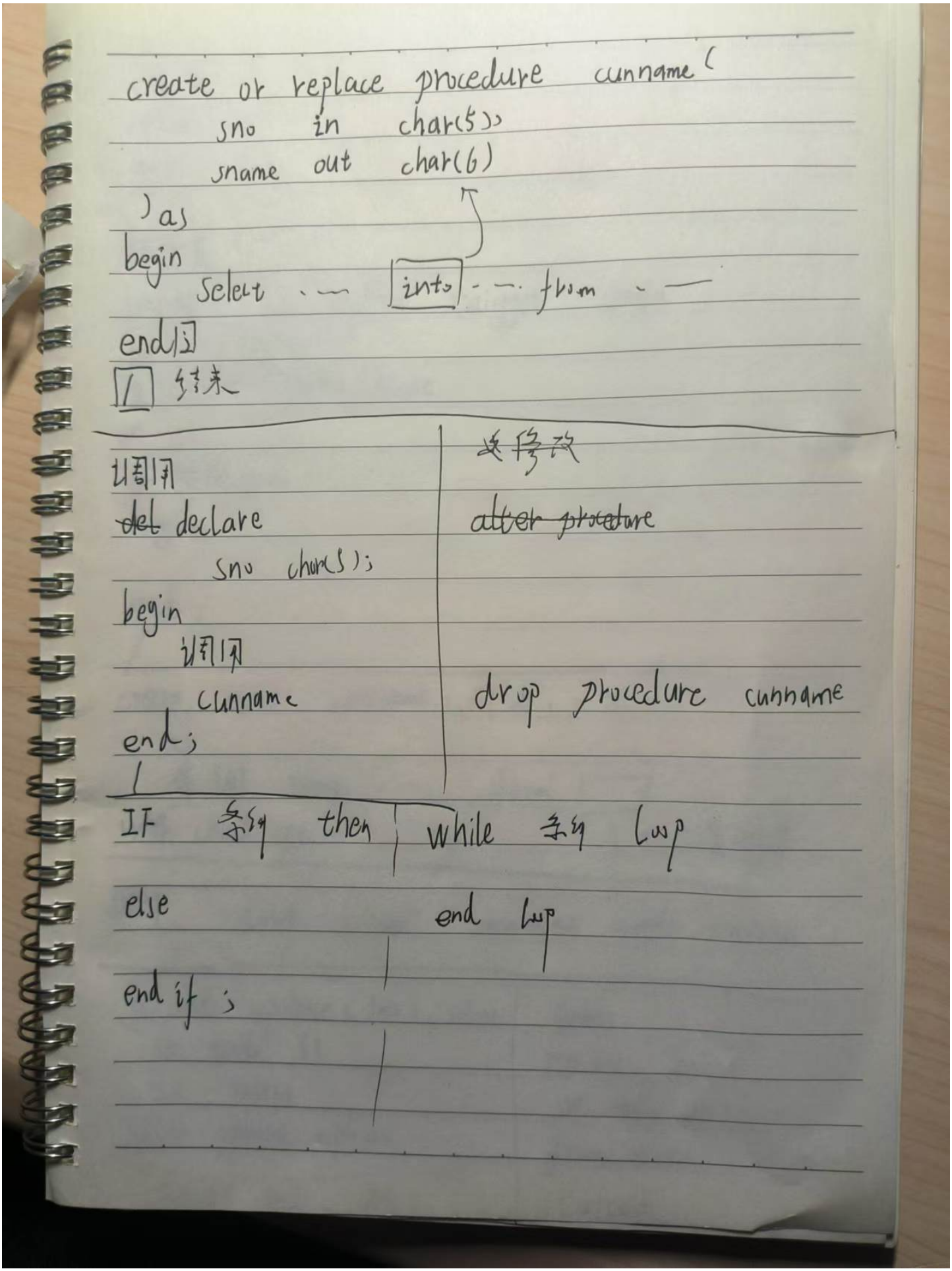
```
creat view viewname (列名, ...)   创
  as
  查询 select ... from □
with check option        ↓ 也可是视图
```

```
删   drop view viewname ~~code~~ cascade
```

```
grant update (sno), select
  on table sc
  to user1
with grant option;
```

```
creat role 角色1
```

```
~~remo~~
revoke select
 on table sc
from user1
 cascade;
```



个人理解的除法逻辑解析

下面解释一下我理解的除法

这么说可能有点难理解，我们可以用一个例子表示：

要求找出所有选了全部课程的学生的姓名

下面是学生表S

学生表 S	学号	姓名
	101	大狗
	102	二狗

下面是课程表C

课程表 C	课号	课名
	201	数据库
	202	高等数学

下面是选课表SC

选课表SC	课号	学号
	201	101
	202	101
	201	102

我们可以看出来，大狗选了所有的课程

代码如下

```
select 姓名  from S
where not exists
(
        select * from C
        where not exists
        (
                select * from SC
                where 学号=S.学号 and 课号=C.课号
        )
);

```

我们可以分步来思考：

所有和S相关的已标黄，和C相关的标灰

一共有三层，

最外层S传进来一个参数 学号，比如叫做101，（之所以这么说是因为最内层用到了最外层S的参数学号）

在最内层，将这个 101 逐一和C中的 课号 比对，看能不能凑一对，如果凑了一对，比如101和201是一对，那么返回一个真（因为找到了，所以返回一个真），但是中间用的是NOT EXISTS,因此返回了一个真，实际上是取反了，那么中间层中的C就会少一个201.（因为C向最内层传递了参数课号）

下一步，101和202比对，看看202是否能和它凑一对，如果可以，那么中间层202也消失了

当101让中间层所有的 课号 都消失了，那么中间层就为空了，只能向最上层返回一个假，由于是NOT EXISTS,因此最上层的 101 对应为真，即101可以对应所有的 课号

然后程序进行102;

当102比对到最后一个课号，发现和202对不上，则发现中间层C还剩下一个202,既然有东西存在，只能向最上层返回一个真，由于是NOT EXISTS,因此最上层的102对应为假，即102不可以对应所有的课号，因此102就消失了

现在所有的学号都比对完成，就剩下了101那一行，提取出那一行对应的姓名，那就是 大狗

(X，Y)÷S(Y，Z)的运算用结构化语言SQL 语句可表达为下列形式

```
select distinct R.X from R R1
where not exists(
    select S.Y from S
    where not exists(
        select * from R R2
        where R2.X=R1.X and R2.Y=S.Y
    )
)
```

我们可以分步来思考：

一共有三层，

最外层R传进来一个参数X，比如叫做X5，（之所以这么说是因为最内层用到了最外层R的参数X）

将这个X5逐一和S中的Y比对，看能不能凑一对，如果凑了一对，比如X5和Y6是一对，那么返回一个真（因为找到了，所以返回一个真），但是中间用的是NOT EXISTS,因此返回了一个真，实际上是取反了，那么中间层中的S就会少一个Y6

下一步，X5和Y7比对，看看Y7是否能和它凑一对，如果可以，那么中间层Y7也消失了

当X5让中间层所有的Y都消失了，那么中间层就为空了，只能向最上层返回一个假，由于是NOT EXISTS,因此最上层的X5对应为真，即X5可以对应所有的Y

然后算法进行X6；

如果当X5比对到最后一个Y,结果发现中间层还剩下一些Y,既然有东西存在，只能向最上层返回一个假，由于是NOT EXISTS,因此最上层的X5对应为假，即X5不可以对应所有的Y，因此X5就消失了

然后算法进行X6；

QCGB