注意link里的例题 ❗

Data Definition Language —— DDL ： 数据定义语言 🚀

数据库系统 | 笔记整理（3）——SQL概述与数据定义 - 知乎 (zhihu.com)

Create、Drop、Alter(修改表结构)

包括定义：

- Entity integrity（实体完整性）——primary key
- Referential integrity（实体完整性）——foreign key
- user-defined integrity（用户定义的完整性）——not null，unique，check，……

```
CREATE TABLE Course
(Cno CHAR(4) PRIMARY KEY,    /* 列级完整性约束条件，Cno是主码*/
Cname CHAR(40) NOT NULL,     /* 列级完整性约束条件，Cname不能取空值*/
Cpno CHAR(4),                /* Cpno的含义是先修课*/
Ccredit SMALLINT,
FOREIGN KEY (Cpno) REFERENCES Course(Cno)
/* 表级完整性约束条件，Cpno是外码，被参照表是Course，被参照列是Cno*/
);

FOREIGN KEY (Cpno) REFERENCES Course(Cno) 中，REFERENCES Course(Cno) 部分指定了外键的
参照关系。这里的 Course 是表名，Cno 是该表中的列名。
```

1. 定义模式
   - `CREATE SCHEMA "S-T" AUTHORIZATION CHEN;`
   - `CREATE SCHEMA AUTHORIZATION CHEN;`

2. 删除模式

   - `DROP SCHEMA <模式名> CASCADE;`
   - `DROP SCHEMA <模式名> RESTRICT;`

3. 定义基本表

   - `CREATE TABLE Student (Sno CHAR(9) PRIMARY KEY, Sname CHAR(20) UNIQUE, Ssex CHAR(2), Sage SMALLINT, Sdept CHAR(20));`
   - `CREATE TABLE Course (Cno CHAR(4) PRIMARY KEY, Cname CHAR(40) NOT NULL, Cpno CHAR(4), Ccredit SMALLINT, FOREIGN KEY (Cpno) REFERENCES Course(Cno));`
   - `CREATE TABLE SC (Sno CHAR(9), Cno CHAR(4), Grade SMALLINT, PRIMARY KEY (Sno,Cno), FOREIGN KEY (Sno) REFERENCES Student(Sno), FOREIGN KEY (Cno) REFERENCES Course(Cno));`

4. 修改基本表

   - `ALTER TABLE Student ADD S_entrance DATE;`
   - `ALTER TABLE Student ALTER COLUMN Sage INT;`
   - `ALTER TABLE Course ADD UNIQUE(Cname);`

5. 删除基本表

   - `DROP TABLE Student CASCADE;`
   - `DROP TABLE Student RESTRICT;`

6. 建立索引

   - `CREATE UNIQUE INDEX Stusno ON Student(Sno);`
   - `CREATE UNIQUE INDEX Coucno ON Course(Cno);`
   - `CREATE UNIQUE INDEX SCno ON SC(Sno ASC, Cno DESC);`

7. 修改索引

   - `ALTER INDEX SCno RENAME TO SCSno;`

8. 删除索引

   - `DROP INDEX Stusname;`

这些语句涵盖了模式的定义与删除、基本表的定义、删除与修改，以及索引的建立、修改与删除的关键操作。

Data Query Language —— DQL：数据查询语言🚀

数据库系统 | 笔记整理（4）——数据查询 - 知乎 (zhihu.com)

Select ❗

Note:

- the qualifier of **distinct all** （对重复元组（duplicate rows）的处理）
- Correlated sub query: the main query and the sub query are relaying on the same table. 相关子查询：主查询（父查询）与子查询依赖于同一张表。与之对应的是不相关子查询：no correlated sub query
- Derived table：派生表（子查询不仅可以出现在where子句中，还可以出现在FROM子句中，这时子查询生成的临时表称为派生表。）
- 其它……

单表查询

1. 选择表中的若干列

```sql
SELECT Sno, Sname FROM Student;
SELECT * FROM Student;
SELECT Sname, 2021-Sage FROM Student;
```

2. 选择表中的若干元组

```sql
SELECT Sno, Sname, Sdept FROM Student WHERE Sdept = 'CS';
SELECT Sname, Sage FROM Student WHERE Sage < 20;
SELECT DISTINCT Sno FROM SC WHERE Grade < 60;
SELECT Sname, Sdept, Sage FROM Student WHERE Sage BETWEEN 20 AND 23;
SELECT Sname, Ssex FROM Student WHERE Sdept IN ('CS', 'MA', 'IS');
SELECT Sname FROM Student WHERE Sname LIKE '刘%';
SELECT Sno, Cno FROM SC WHERE Grade IS NULL;
SELECT Sname FROM Student WHERE Sdept = 'CS' AND Sage < 20;
```

3. **ORDER BY** 子句

```sql
SELECT Sno, Grade FROM SC WHERE Cno = '3' ORDER BY Grade DESC;
SELECT * FROM Student ORDER BY Sdept, Sage DESC;
```

4. 聚集函数

```
SELECT COUNT(*) FROM Student;
SELECT COUNT(DISTINCT Sno) FROM SC;
SELECT AVG(Grade) FROM SC WHERE Cno = '1';
SELECT MAX(Grade) FROM SC WHERE Cno = '1';
SELECT SUM(Ccredit) FROM SC, Course WHERE Sno = '201215012' AND SC.Cno =
Course.Cno;
```

5. **GROUP BY** 子句

```
SELECT Cno, COUNT(Sno) FROM SC GROUP BY Cno;
SELECT Sno FROM SC GROUP BY Sno HAVING COUNT(*) > 3;
SELECT Sno, AVG(Grade) FROM SC GROUP BY Sno HAVING AVG(Grade) >= 90;
```

连接查询

1. 等值与非等值连接查询

```
SELECT Student.*, SC.* FROM Student, SC WHERE Student.Sno = SC.Sno;
```

2. 自身连接

```
SELECT FIRST.Cno, SECOND.Cpno FROM Course FIRST, Course SECOND WHERE
FIRST.Cpno = SECOND.Cno;
```

3. 外连接

```
SELECT Student.Sno, Sname, Ssex, Sage, Sdept, Cno, Grade FROM Student LEFT
OUTER JOIN SC ON (Student.Sno = SC.Sno);
```

4. 多表连接

```
SELECT Student.Sno, Sname, Cname, Grade FROM Student, SC, Course WHERE
Student.Sno = SC.Sno AND SC.Cno = Course.Cno;
```

嵌套查询

1. 带有 **IN** 谓词的子查询

```sql
SELECT Sname FROM Student WHERE Sno IN (SELECT Sno FROM SC WHERE Cno = '2');
```

2. 带有比较运算符的子查询

```sql
SELECT Sno, Cno FROM SC x WHERE Grade >= (SELECT AVG(Grade) FROM SC y WHERE
y.Sno = x.Sno);
```

3. 带有 **ANY**（**SOME**）或 **ALL** 谓词的子查询

```sql
SELECT Sname, Sage FROM Student WHERE Sage < ANY (SELECT Sage FROM Student
WHERE Sdept = 'CS') AND Sdept <> 'CS';
```

4. 带有 **EXISTS** 谓词的子查询

```sql
SELECT Sname FROM Student WHERE EXISTS (SELECT * FROM SC WHERE Sno =
Student.Sno AND Cno = '1');
SELECT Sname FROM Student WHERE NOT EXISTS (SELECT * FROM SC WHERE Sno =
Student.Sno AND Cno = '1');
```

Data Manipulation Language —— DML：数据操纵语言

Insert，Update(修改表中的数据)，Delete

数据库系统 | 笔记整理（5）——数据更新、空值的处理、视图 - 知乎 (zhihu.com)

插入数据

1. 插入元组

```sql
INSERT INTO Student (Sno, Sname, Ssex, Sdept, Sage) VALUES ('201215128', '陈
冬', '男', 'IS', 18);
INSERT INTO Student VALUES ('201215126', '张成民', '男', 18, 'CS');
INSERT INTO SC (Sno, Cno) VALUES ('201215128', '1'); -- 或者 INSERT INTO SC
VALUES ('201215128', '1', NULL);
```

2. 插入子查询结果

```
-- 创建表
CREATE TABLE Dept_age (Sdept CHAR(15), Avg_age SMALLINT);
-- 插入数据
INSERT INTO Dept_age (Sdept, Avg_age) SELECT Sdept, AVG(Sage) FROM Student
GROUP BY Sdept;
```

修改数据

```
UPDATE Student SET Sage=22 WHERE Sno='201215121';
UPDATE Student SET Sage=Sage+1;
UPDATE SC SET Grade=0 WHERE Sno IN (SELECT Sno FROM Student WHERE Sdept='CS');
```

删除数据

```
DELETE FROM Student WHERE Sno='201215128';
DELETE FROM SC;
DELETE FROM SC WHERE Sno IN (SELECT Sno FROM Student WHERE Sdept='CS');
```

空值的处理

1. 空值的产生

```
INSERT INTO SC (Sno, Cno, Grade) VALUES ('201215126', '1', NULL);
UPDATE Student SET Sdept=NULL WHERE Sno='201215200';
```

2. 空值的判断

```
SELECT * FROM Student WHERE Sname IS NULL OR Ssex IS NULL OR Sage IS NULL OR
Sdept IS NULL;
```

视图

1. 定义视图

```
CREATE VIEW IS_Student AS SELECT Sno, Sname, Sage FROM Student WHERE
Sdept='IS';
CREATE VIEW IS_Student AS SELECT Sno, Sname, Sage FROM Student WHERE
Sdept='IS' WITH CHECK OPTION;
CREATE VIEW IS_S1 (Sno, Sname, Grade) AS SELECT Student.Sno, Sname, Grade FROM
Student, SC WHERE Sdept='IS' AND Student.Sno=SC.Sno AND SC.Cno='1';
CREATE VIEW IS_S2 AS SELECT Sno, Sname, Grade FROM IS_S1 WHERE Grade>=90;
CREATE VIEW BT_S (Sno, Sname, Sbirth) AS SELECT Sno, Sname, 2021-Sage FROM
Student;
CREATE VIEW S_G (Sno, Gavg) AS SELECT Sno, AVG(Grade) FROM SC GROUP BY Sno;
CREATE VIEW F_Student (F_sno, name, sex, age, dept) AS SELECT * FROM Student
WHERE Ssex='女';
```

2. 删除视图

```
DROP VIEW BT_S;
DROP VIEW IS_S1 CASCADE;
```

3. 查询视图

```
SELECT Sno, Sage FROM IS_Student WHERE Sage<20;
SELECT IS_Student.Sno, Sname FROM IS_Student, SC WHERE IS_Student.Sno=SC.Sno
AND SC.Cno='1';
```

4. 更新视图

```
UPDATE IS_Student SET Sname='刘辰' WHERE Sno='201215122';
INSERT INTO IS_Student VALUES ('201215129', '赵新', 20);
DELETE FROM IS_Student WHERE Sno='201215129';
```

Data Control Language —— DCL：数据控制语言

Grant，Revoke

GRANT 用法

GRANT语句用于给用户授权，允许他们执行特定的数据库操作，如查询、更新、删除数据等。

```sql
-- 授予用户SELECT权限，允许其查询指定的表
GRANT SELECT ON table_name TO user_name;

-- 授予用户对表的SELECT和INSERT权限
GRANT SELECT, INSERT ON table_name TO user_name;

-- 授予用户所有权限
GRANT ALL PRIVILEGES ON table_name TO user_name;

-- 授予用户对数据库所有表的SELECT权限
GRANT SELECT ON DATABASE database_name TO user_name;

-- 授予用户对表的SELECT权限，并允许他们将这个权限授予其他用户
GRANT SELECT ON table_name TO user_name WITH GRANT OPTION;
```

REVOKE 用法

`REVOKE`语句用于撤销之前通过 `GRANT`授予的权限。

```sql
-- 撤销用户对表的SELECT权限
REVOKE SELECT ON table_name FROM user_name;

-- 撤销用户对表的SELECT和INSERT权限
REVOKE SELECT, INSERT ON table_name FROM user_name;

-- 撤销用户所有权限
REVOKE ALL PRIVILEGES ON table_name FROM user_name;

-- 撤销用户对数据库所有表的SELECT权限
REVOKE SELECT ON DATABASE database_name FROM user_name;

-- 如果用户被授予了WITH GRANT OPTION的权限，也可以撤销
REVOKE GRANT OPTION FOR SELECT ON table_name FROM user_name;
```

在使用 `GRANT`和 `REVOKE`语句时，必须拥有足够的权限来授予或撤销相应的权限，并且通常这些操作是由数据库管理员执行的。需要注意的是，`REVOKE`语句只能撤销用户直接被授予的权限，如果用户通过角色或其他途径间接获得的权限，则需要单独处理。

其他

> 要求能正确书写各类SQL代码，特别是 **数据定义、数据查询**。 典型题型：教材第3章中的各例题。

> 区别:alter用来修改基本表,是对表的结构进行操作，比如对字段增加，删除，修改类型,而update用来修改表中的数据，修改某一行某一列的值

表、视图、查询表的区别（关系的三种基本类型）_查询表和视图表的区别-CSDN博客

```
CREATE VIEW IS_Student
AS
SELECT Sno,Sname,Sage
FROM Student
WHERE Sdept='IS'
WITH CHECK OPTION;
和
SELECT Sno,Sname,Sage
FROM Student
WHERE Sdept='IS'
WITH CHECK OPTION;
有什么区别呢

两者都是查询操作，但它们用于不同的目的。
第一个查询是用来创建一个名为 "IS_Student" 的视图。视图是存储在数据库中的预定义的SQL查询。
它可以从一张或多张表中获取数据，并用单独的视图名存储起来，以简化复杂查询的使用。视图本质上
是基于SQL查询的虚拟表。使用CREATE VIEW语句，你可以创建一个视图，此后可以像查询一个真实的表
一样查询这个视图。在这个例子中，IS_Student视图将包含所有Sdept为IS的学生信息。WITH CHECK
OPTION 保证了数据的完整性，这意味着对视图的修改（如果允许）只能在满足查询条件的前提下进行，
否则会被拒绝。
第二个查询直接从 "Student" 表中选择所有Sdept为IS的学生。这是一个简单的SELECT查询，它返回
的是一次性的查询结果，并没有创建持久的视图。需要注意的是，WITH CHECK OPTION只能用在CREATE
VIEW命令中，你在普通的SELECT语句中看到这一选项可能是因为文档写错了，把创建视图的命令和普通
查询混淆了。在普通查询中它没有效果，使用它可能会导致错误。
简而言之，视图查询和直接查询的主要区别在于，视图可以保存并重复使用，而直接查询则是一次性
的。
```