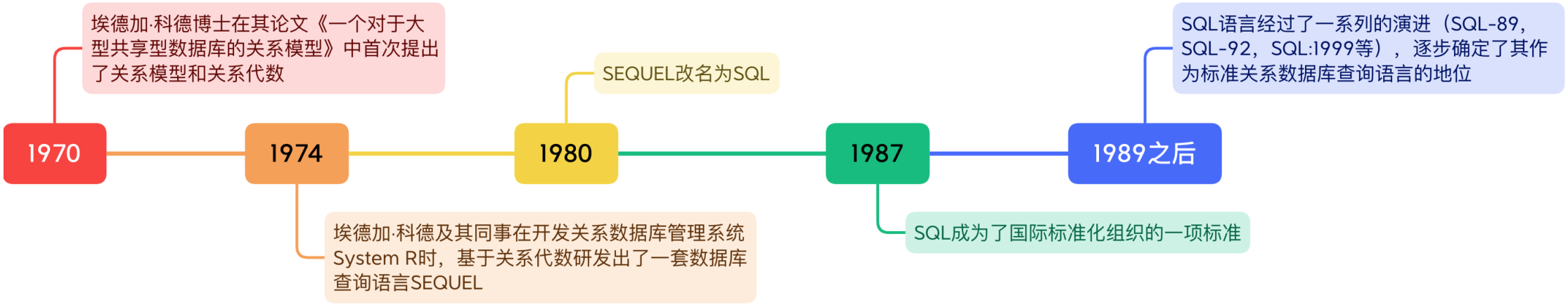
数据库系统概论

计算机与信息学院
人 工 智 能 学 院

SQL

- SQL：Structured Query Language，结构化查询语言
- 关系型数据库中的数据操作语言

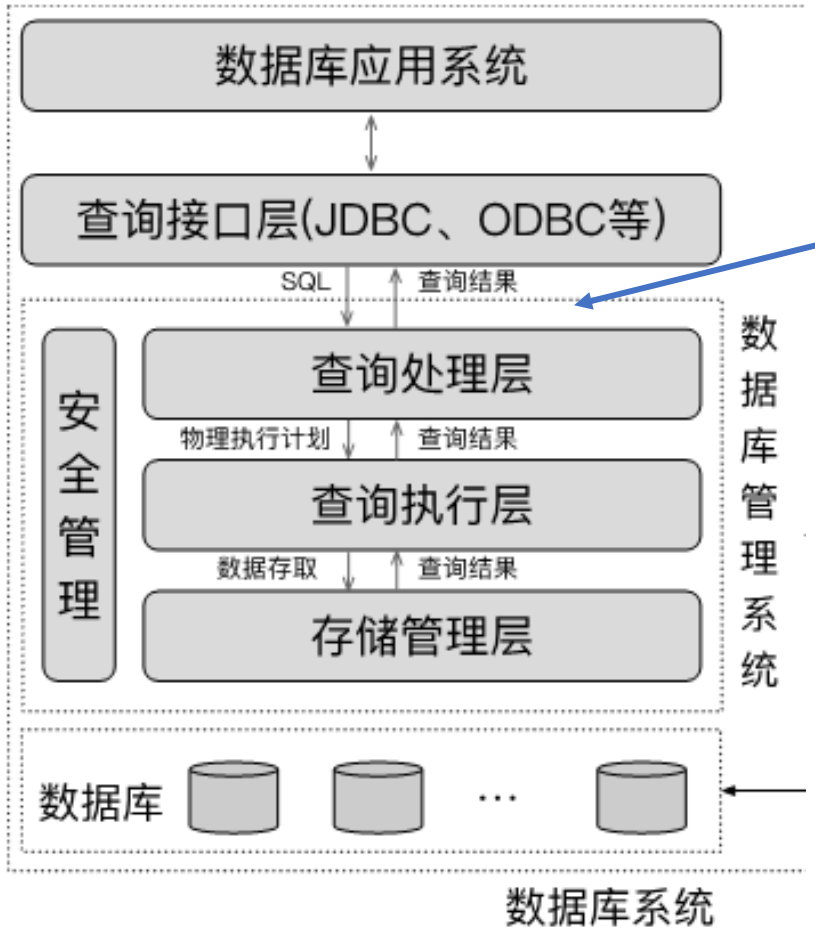


不同的关系数据库在SQL实现上有差异

- ✓ 对标准的支持不同
- ✓ 部分DBMS支持非标准的特性

基础SQL：语法格式

SQL



SQL客户端工具

SQLserver：SQL Server Management Studio

MySQL：Navicat

Oracle： Navicat 、SQLPlus

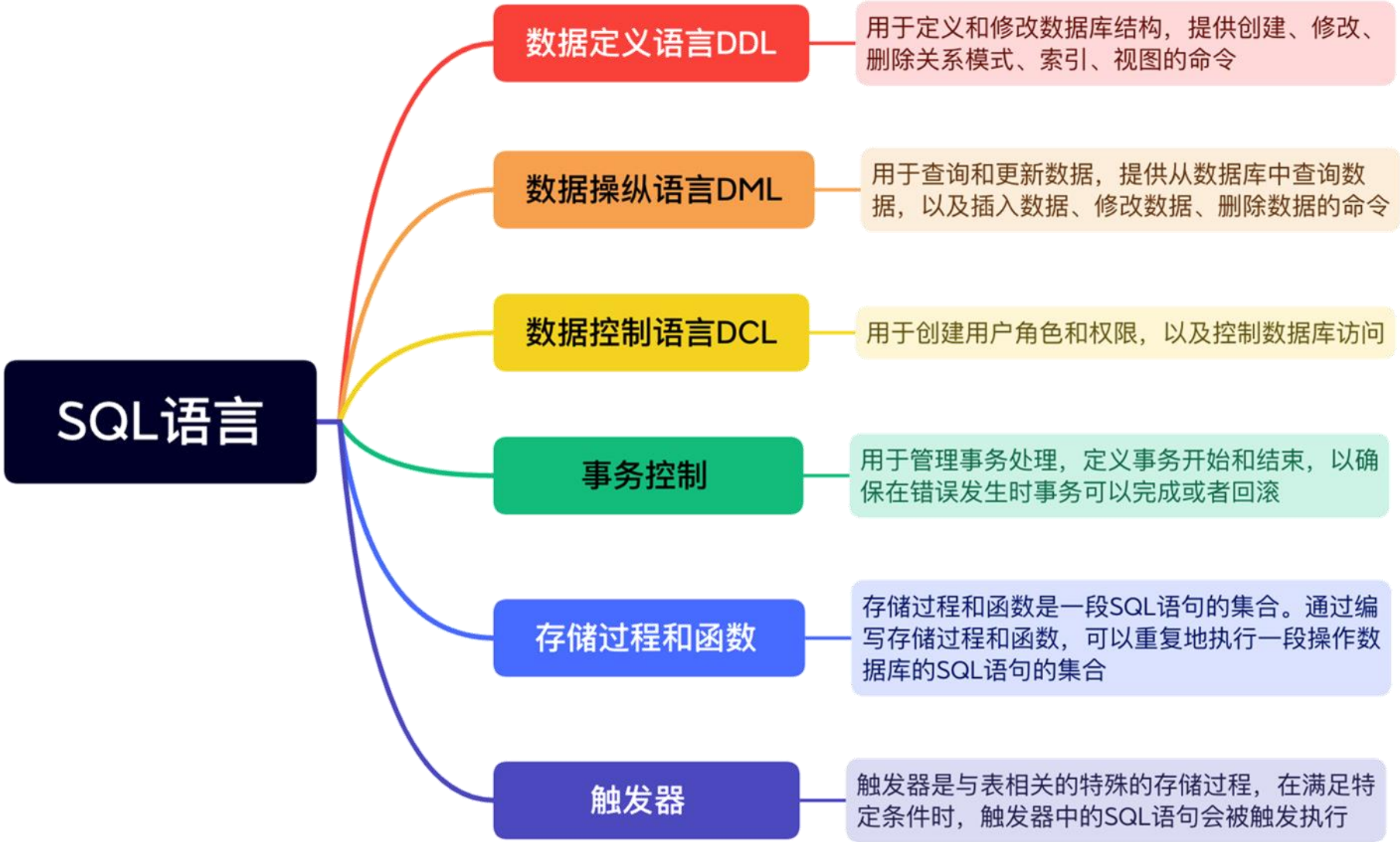
OpenGauss：gsql、DataStudio

GaussDB：DAS、gsql、 Navicat

SQL

sql.js demo: Online SQL interpreter（https://sql.js.org/examples/GUI/）

SQL.js是一个JavaScript库，允许在浏览器中创建和查询一个关系数据库。使用一个存储在浏览器内存中的虚拟数据库文件，所以它不会持久化对数据库的修改。


SQL语言
数据定义语言DDL
用于定义和修改数据库结构，提供创建、修改、删除关系模式、索引、视图的命令
数据操纵语言DML
用于查询和更新数据，提供从数据库中查询数据，以及插入数据、修改数据、删除数据的命令
数据控制语言DCL
用于创建用户角色和权限，以及控制数据库访问
事务控制
用于管理事务处理，定义事务开始和结束，以确保在错误发生时事务可以完成或者回滚
存储过程和函数
存储过程和函数是一段SQL语句的集合。通过编写存储过程和函数，可以重复地执行一段操作数据库的SQL语句的集合
触发器
触发器是与表相关的特殊的存储过程，在满足特定条件时，触发器中的SQL语句会被触发执行

1）DDL：数据定义语言：**创建、修改、删除**

	创建	修改	删除
用户	CREATE USER	ALTER USER	DROP USER
表空间	CREATE TABLESPACE	ALTER TABLESPACE	DROP TABLESPACE
数据库	CREATE DATABASE	ALTER DATABASE	DROP DATABASE
模式	CREATE SCHEMA	ALTER SCHEMA	DROP SCHEMA
表	CREATE TABLE	ALTER TABLE	DROP TABLE
索引	CREATE INDEX	ALTER INDEX	DROP INDEX
视图	CREATE VIEW	ALTER VIEW	DROP VIEW
序列	CREATE SEQUENCE	ALTER SEQUENCE	DROP SEQUENCE

CREATE Table

```
CREATE TABLE <表名> (
    <列名> <数据类型> [列级完整性约束] … [列级完整性约束]
    [,<列名> <数据类型> [列级完整性约束] … [列级完整性约束]]
    …
    [,<列名> <数据类型> [列级完整性约束] … [列级完整性约束]]
    [,表级完整性约束]
    …
    [,表级完整性约束]
);
```

数据类型	含义
CHAR(n)	长度为n的定长字符串
VARCHAR(n)	最大长度为n的变长字符串
INT	长整数（也可以写作INTEGER）
SMALLINT	短整数
NUMERIC(p，d)	定点数，由p位数字（不包括符号、小数点）组成，小数后面有d位数字
REAL	取决于机器精度的浮点数
Double Precision	取决于机器精度的双精度浮点数
FLOAT(n)	浮点数，精度至少为n位数字
DATE	日期，包含年、月、日，格式为YYYY-MM-DD
TIME	时间，包含一日的时、分、秒，格式为HH:MM:SS

```
CREATE TABLE <表名> (
    <列名> <数据类型> [列级完整性约束] … [列级完整性约束]
    [,<列名> <数据类型> [列级完整性约束] … [列级完整性约束]]
    …
    [,<列名> <数据类型> [列级完整性约束] … [列级完整性约束]]
    [,表级完整性约束]
    …
    [,表级完整性约束]
);
```

- Student(Sno, Sname, Ssex, Sbirthdate, Smajor)

```
CREATE TABLE Student
       (Sno    CHAR(8) Primary key,
        Sname VARCHAR(20) not null,
        Ssex     CHAR(6),
        Sbirthdate Date,
        Smajor  VARCHAR(40)
       );
```

- SC(Sno,Cno,Grade,Semester,Teachingclass)

```
Drop table if exists sc;
CREATE TABLE SC
  (Sno CHAR(8),
   Cno CHAR(5),
   Grade SMALLINT,
   Semester CHAR(5),
   Teachingclass CHAR(8),
   PRIMARY KEY (Sno,Cno),
   FOREIGN KEY (Sno) REFERENCES Student(Sno)
);
```

ALTER Table

```
ALTER TABLE <表名> ADD [COLUMN] <列名> <数据类型>;
```

```
ALTER TABLE <表名> DROP [COLUMN] <列名> [RESTRICT | CASCADE];
```

```
ALTER TABLE <表名> ALTER COLUMN <列名> <数据类型>;
```

Enter some SQL

```
1 ALTER TABLE Student ADD Semail VARCHAR(30);
2
3 ALTER TABLE Student DROP Semail;
```

DROP Table

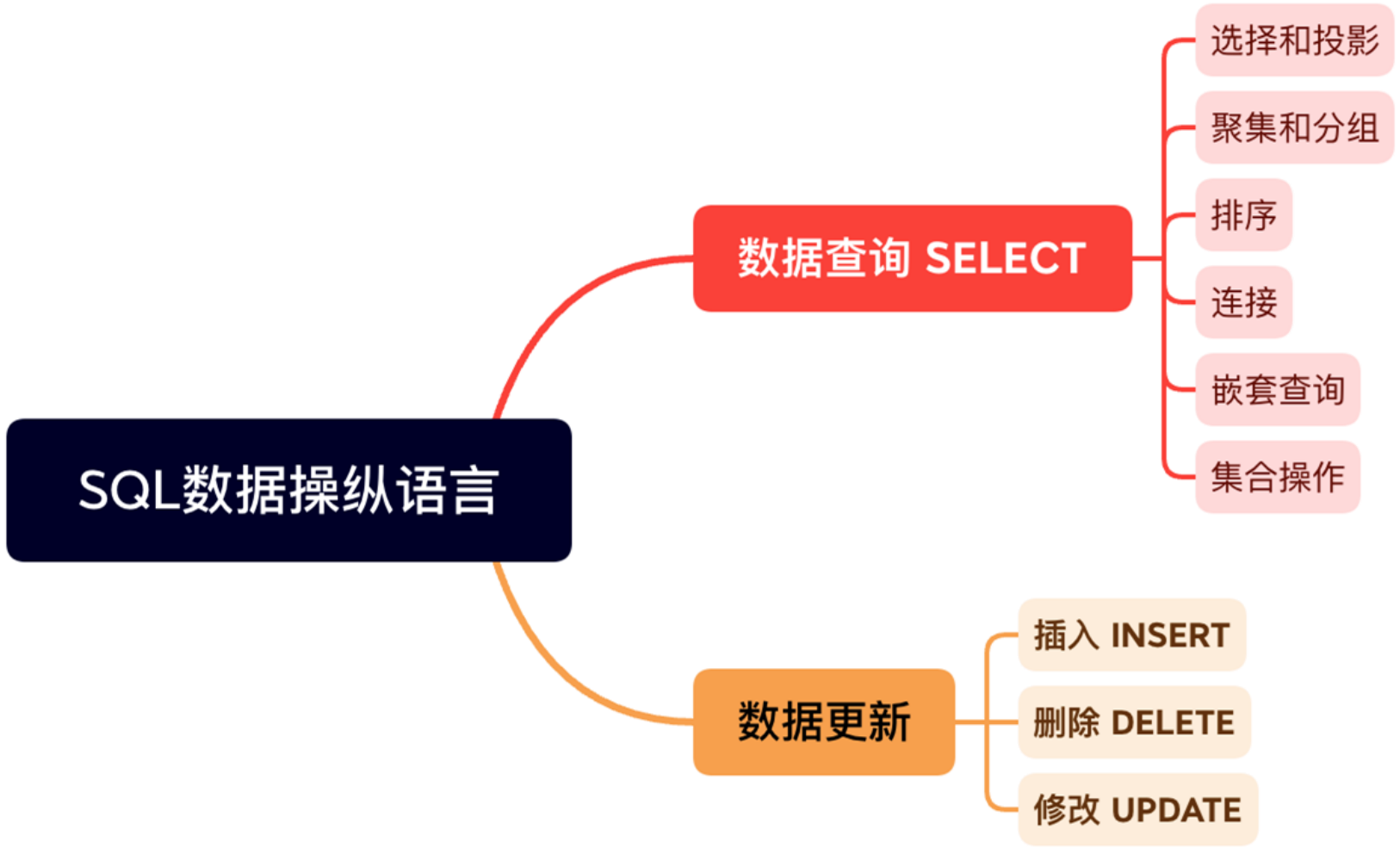
```
DROP TABLE <表名> [,<表名>] … [,<表名>] [RESTRICT | CASCADE];
```

- Student(Sno, Sname, Ssex, Sbirthdate, Smajor)
- Course(Cno,Cname,Ccredit,Cpno)
- SC(Sno,Cno,Grade,Semester,Teachingclass)

Enter some SQL

```
1 drop table student;
2
```

2）DML：数据操纵语言：**增加、修改、删除、查询**



INSERT

INSERT 单个元组

```
INSERT
INTO <表名> [(<列名 1> [,<列名 2>, …, <列名 n>])]
VALUES (<常量 1> [,<常量 2>, …, <常量 n>]);
```

Enter some SQL

```
1  INSERT INTO Student (Sno,Sname,Ssex,Smajor,Sbirthdate)
2  VALUES ('20180009', '陈冬', '男','信息管理与信息系统', '2000-5-22');
3
4  INSERT INTO Student
5  VALUES ('20180008', '张成民', '男','2000-4-15','计算机科学与技术');
6
7
8  select * from student;
9
10
11
12
13
14
15
```

Execute | Save the db | Load an SQLite database file: 选择文件 未选择文件

Sno	Sname	Ssex	Sbirthdate	Smajor
20180009	陈冬	男	2000-5-22	信息管理与信息系统
20180008	张成民	男	2000-4-15	计算机科学与技术

UPDATE

```
UPDATE <表名>
SET <列名 1>=<表达式 1> [,<列名 2>=<表达式 2>, …, <列名 n>=<表达式 n>]
[WHERE <条件>];
```

Enter some SQL

```
1  UPDATE student
2      SET ssex= '女'
3      WHERE sno='20180009';
4
5
6
7  select * from student;
8
9
10
11
12
13
14
```

Execute | Save the db | Load an SQLite database file: 选择文件 未选择文件

Sno	Sname	Ssex	Sbirthdate	Smajor
20180009	陈冬	女	2000-5-22	信息管理与信息系统
20180008	张成民	男	2000-4-15	计算机科学与技术

DELETE

```
DELETE
FROM <表名>
[WHERE <条件>];
```

Enter some SQL

```
1  delete from student
2  where sno='20180009';
3
4
5
6  select * from student;
7
8
9
10
11
12
13
```

Execute | Save the db | Load an SQLite database file: 选择文件 未选择文件

Sno	Sname	Ssex	Sbirthdate	Smajor
20180008	张成民	男	2000-4-15	计算机科学与技术

SELECT

```
SELECT [ALL | DISTINCT] <列表达式>[,<列表达式>,…,<列表达式>]
FROM <表名或视图名>[, <表名或视图名>, … ,<表名或视图名>]
[WHERE <条件表达式>]
[GROUP BY <列名>[,<列名>,…,<列名>]
[HAVING <条件表达式>]]
[ORDER BY <列表达式> [<次序>][,<列表达式> [<次序>],…, <列表达式> [<次序>]]];
```

➢单表查询

1）查询所有学生的基本信息

2）查询所有学生的学号、姓名

3）查询计算机科学与技术专业全体学生的姓名

1）select * from student

2）select sno，sname
from student

3）select sno，sname
from student
where smajor= ‘计算机科学与技术’

1）学生关系模式：Student(Sno, Sname, Ssex, Sbirthdate, Smajor)
包括学号、姓名、性别、出生日期和主修专业等属性

学号 Sno	姓名 Sname	性别 Ssex	出生日期 Sbirthdate	主修专业 Smajor
20180001	李勇	男	2000-3-8	信息安全
20180002	刘晨	女	1999-9-1	计算机科学与技术
20180003	王敏	女	2001-8-1	计算机科学与技术
20180004	张立	男	2000-1-8	计算机科学与技术
20180005	陈新奇	男	2001-11-1	信息管理与信息系统
20180006	赵明	男	2000-6-12	数据科学与大数据技术
20180007	王佳佳	女	2001-12-7	数据科学与大数据技术

```
SELECT [ALL | DISTINCT] <列表达式>[,<列表达式>,…,<列表达式>]
FROM <表名或视图名>[, <表名或视图名>, … ,<表名或视图名>]
[WHERE <条件表达式>]
```

```
SELECT Sname AS NAME, 2021-Sage AS BIRTHDAY, LOWER(Sdept) AS DEPARTMENT
FROM Student;
```

1）学生关系模式：Student(Sno, Sname, Ssex, Sbirthdate, Smajor)
包括学号、姓名、性别、出生日期和主修专业等属性

学号 Sno	姓名 Sname	性别 Ssex	出生日期 Sbirthdate	主修专业 Smajor
20180001	李勇	男	2000-3-8	信息安全
20180002	刘晨	女	1999-9-1	计算机科学与技术
20180003	王敏	女	2001-8-1	计算机科学与技术
20180004	张立	男	2000-1-8	计算机科学与技术
20180005	陈新奇	男	2001-11-1	信息管理与信息系统
20180006	赵明	男	2000-6-12	数据科学与大数据技术
20180007	王佳佳	女	2001-12-7	数据科学与大数据技术

```
SELECT [ALL | DISTINCT] <列表达式>[,<列表达式>,…,<列表达式>]
FROM <表名或视图名>[, <表名或视图名>, … ,<表名或视图名>]
[WHERE <条件表达式>]
```

查询条件	谓词
比较	=，>，<，>=，<=，!=，<>，!>，!<；NOT+上述比较运算符
确定范围	BETWEEN AND，NOT BETWEEN AND
确定集合	IN，NOT IN
字符匹配	LIKE，NOT LIKE
空值	IS NULL，IS NOT NULL
多重条件（逻辑运算）	AND，OR，NOT

- SQL提供了聚集函数，查询统计值（平均值、最大值、最小值等）。

聚集函数	含义
COUNT([DISTINCT | ALL *])	统计元组个数
COUNT([DISTINCT | ALL] <列名>)	统计一列值的个数
SUM([DISTINCT | ALL] <列名>)	统计一列值的总和
AVG([DISTINCT | ALL] <列名>)	统计一列值的平均值
MAX([DISTINCT | ALL] <列名>)	统计一列值的最大值
MIN([DISTINCT | ALL] <列名>)	统计一列值的最小值

➢聚集查询

1）统计学生的人数

2）统计专业的个数

3）统计信息安全的学生人数

4）统计每个专业的学生人数

1）学生关系模式：Student(Sno, Sname, Ssex, Sbirthdate, Smajor)
包括学号、姓名、性别、出生日期和主修专业等属性

学号 Sno	姓名 Sname	性别 Ssex	出生日期 Sbirthdate	主修专业 Smajor
20180001	李勇	男	2000-3-8	信息安全
20180002	刘晨	女	1999-9-1	计算机科学与技术
20180003	王敏	女	2001-8-1	计算机科学与技术
20180004	张立	男	2000-1-8	计算机科学与技术
20180005	陈新奇	男	2001-11-1	信息管理与信息系统
20180006	赵明	男	2000-6-12	数据科学与大数据技术
20180007	王佳佳	女	2001-12-7	数据科学与大数据技术

●Group by：分组统计

```
SELECT [ALL | DISTINCT] <列表达式>[,<列表达式>,…,<列表达式>]
FROM <表名或视图名>[, <表名或视图名>, … ,<表名或视图名>]
[WHERE <条件表达式>]
[GROUP BY <列名>[,<列名>,…,<列名>]
[HAVING <条件表达式>]]
[ORDER BY <列表达式> [<次序>][,<列表达式> [<次序>],…, <列表达式> [<次序>]]];
```

Select smajor, count(*)

From student

Group by smajor

1）学生关系模式：Student(Sno, Sname, Ssex, Sbirthdate, Smajor)
包括学号、姓名、性别、出生日期和主修专业等属性

学号 Sno	姓名 Sname	性别 Ssex	出生日期 Sbirthdate	主修专业 Smajor
20180001	李勇	男	2000-3-8	信息安全
20180002	刘晨	女	1999-9-1	计算机科学与技术
20180003	王敏	女	2001-8-1	计算机科学与技术
20180004	张立	男	2000-1-8	计算机科学与技术
20180005	陈新奇	男	2001-11-1	信息管理与信息系统
20180006	赵明	男	2000-6-12	数据科学与大数据技术
20180007	王佳佳	女	2001-12-7	数据科学与大数据技术

●Group by：分组统计

```
SELECT [ALL | DISTINCT] <列表达式>[,<列表达式>,…,<列表达式>]
FROM <表名或视图名>[, <表名或视图名>, … ,<表名或视图名>]
[WHERE <条件表达式>]
[GROUP BY <列名>[,<列名>,…,<列名>]
[HAVING <条件表达式>]]
[ORDER BY <列表达式> [<次序>][,<列表达式> [<次序>],…, <列表达式> [<次序>]]];
```

- 统计人数超过50人的专业

Select smajor, count(*)

From student

Group by smajor

Having (count(*)>50)

1）学生关系模式：Student(Sno, Sname, Ssex, Sbirthdate, Smajor)
包括学号、姓名、性别、出生日期和主修专业等属性

学号 Sno	姓名 Sname	性别 Ssex	出生日期 Sbirthdate	主修专业 Smajor
20180001	李勇	男	2000-3-8	信息安全
20180002	刘晨	女	1999-9-1	计算机科学与技术
20180003	王敏	女	2001-8-1	计算机科学与技术
20180004	张立	男	2000-1-8	计算机科学与技术
20180005	陈新奇	男	2001-11-1	信息管理与信息系统
20180006	赵明	男	2000-6-12	数据科学与大数据技术
20180007	王佳佳	女	2001-12-7	数据科学与大数据技术

```
SELECT [ALL | DISTINCT] <列表达式>[,<列表达式>,…,<列表达式>]
FROM <表名或视图名>[, <表名或视图名>, … ,<表名或视图名>]
[WHERE <条件表达式>]
[GROUP BY <列名>[,<列名>,…,<列名>]
[HAVING <条件表达式>]]
[ORDER BY <列表达式> [<次序>][,<列表达式> [<次序>],…, <列表达式> [<次序>]]];
```

- Where：在关系上进行选择（选择运算）
- Group by：分组统计
- Having：对分组统计的结果进行条件选择

- 查询2019年第2学期选修了10门以上课程的学生学号

```
SELECT Sno
FROM  SC
WHERE Semester='20192'
GROUP BY

HAVING COUNT(*) >10;

Order by
```

3）学生选课关系模式：SC(Sno,Cno, Grade,Semester,Teachingclass)
包括学号，课程号，成绩，选课学期，教学班等

学号 Sno	课程号 Cno	成绩 Grade	选课学期 Semester	教学班 Teachingclass
20180001	81001	85	20192	81001-01
20180001	81002	96	20201	81002-01
20180001	81003	87	20202	81003-01
20180002	81001	80	20192	81001-02
20180002	81002	98	20201	81002-01
20180002	81003	71	20202	81003-02
20180003	81001	81	20192	81001-01
20180003	81002	76	20201	81002-02
20180004	81001	56	20192	81001-02
20180004	81003	97	20201	81002-02
20180005	81003	68	20202	81003-01

●LIMIT：限制查询结果的（元组）数量

LIMIT <行数1>[OFFSET <行数2>];

3）学生选课关系模式：SC(Sno,Cno, Grade,Semester,Teachingclass)
包括学号，课程号，成绩，选课学期，教学班等

学号 Sno	课程号 Cno	成绩 Grade	选课学期 Semester	教学班 Teachingclass
20180001	81001	85	20192	81001-01
20180001	81002	96	20201	81002-01
20180001	81003	87	20202	81003-01
20180002	81001	80	20192	81001-02
20180002	81002	98	20201	81002-01
20180002	81003	71	20202	81003-02
20180003	81001	81	20192	81001-01
20180003	81002	76	20201	81002-02
20180004	81001	56	20192	81001-02
20180004	81003	97	20201	81002-02
20180005	81003	68	20202	81003-01

- 选修了81001号课程，成绩排名在前10的学生

```
Select sno，grade
From sc
Order by grade
Limit 10
```

- 选择操作可以选择表中的若干元组，通过WHERE子句实现
- 投影操作可以选择表中的若干列，通过SELECT子句实现

- **连接查询：**

$$R \underset{A\theta B}{\bowtie} S = \{ \widehat{t_r t_s} \mid t_r \in R \wedge t_s \in S \wedge t_r[A] \theta t_s[B] \}$$

➢**连接查询：**在FROM子句中指定需要连接的表，在WHERE子句中指定**连接条件**

[<表名1>.]<列名1> <比较运算符> [<表名2>.]<列名2>

1）学生关系模式：Student(Sno, Sname, Ssex, Sbirthdate, Smajor)
包括学号、姓名、性别、出生日期和主修专业等属性

学号 Sno	姓名 Sname	性别 Ssex	出生日期 Sbirthdate	主修专业 Smajor
20180001	李勇	男	2000-3-8	信息安全
20180002	刘晨	女	1999-9-1	计算机科学与技术
20180003	王敏	女	2001-8-1	计算机科学与技术
20180004	张立	男	2000-1-8	计算机科学与技术
20180005	陈新奇	男	2001-11-1	信息管理与信息系统
20180006	赵明	男	2000-6-12	数据科学与大数据技术
20180007	王佳佳	女	2001-12-7	数据科学与大数据技术

3）学生选课关系模式：SC(Sno,Cno, Grade,Semester,Teachingclass)
包括学号，课程号，成绩，选课学期，教学班等

学号 Sno	课程号 Cno	成绩 Grade	选课学期 Semester	教学班 Teachingclass
20180001	81001	85	20192	81001-01
20180001	81002	96	20201	81002-01
20180001	81003	87	20202	81003-01
20180002	81001	80	20192	81001-02
20180002	81002	98	20201	81002-01
20180002	81003	71	20202	81003-02
20180003	81001	81	20192	81001-01
20180003	81002	76	20201	81002-02
20180004	81001	56	20192	81001-02
20180004	81003	97	20201	81002-02
20180005	81003	68	20202	81003-01

- 查询每位学生的选课信息

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

```
SELECT  Student.Sno，Sname，Ssex，Sage，Sdept，Cno，Grade
FROM    Student，SC
WHERE  Student.Sno = SC.Sno；
```

1）学生关系模式：Student(Sno, Sname, Ssex, Sbirthdate, Smajor)
包括学号、姓名、性别、出生日期和主修专业等属性

学号 Sno	姓名 Sname	性别 Ssex	出生日期 Sbirthdate	主修专业 Smajor
20180001	李勇	男	2000-3-8	信息安全
20180002	刘晨	女	1999-9-1	计算机科学与技术
20180003	王敏	女	2001-8-1	计算机科学与技术
20180004	张立	男	2000-1-8	计算机科学与技术
20180005	陈新奇	男	2001-11-1	信息管理与信息系统
20180006	赵明	男	2000-6-12	数据科学与大数据技术
20180007	王佳佳	女	2001-12-7	数据科学与大数据技术

3）学生选课关系模式：SC(Sno,Cno, Grade,Semester,Teachingclass)
包括学号，课程号，成绩，选课学期，教学班等

学号 Sno	课程号 Cno	成绩 Grade	选课学期 Semester	教学班 Teachingclass
20180001	81001	85	20192	81001-01
20180001	81002	96	20201	81002-01
20180001	81003	87	20202	81003-01
20180002	81001	80	20192	81001-02
20180002	81002	98	20201	81002-01
20180002	81003	71	20202	81003-02
20180003	81001	81	20192	81001-01
20180003	81002	76	20201	81002-02
20180004	81001	56	20192	81001-02
20180004	81003	97	20201	81002-02
20180005	81003	68	20202	81003-01

查询选修81002号课程且成绩在90分以上的学生学号和姓名

- 查询每位学生的选课信息

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

- 查询选修81002号课程且成绩在90分以上的学生学号和姓名

```
SELECT Student.Sno,Sname

FROM Student,SC

WHERE Student.Sno=SC.Sno  AND  SC.Cno='81002' AND SC.Grade>90;
```

- 连接的两种写法

```
SELECT Student.Sno,Sname
FROM Student,SC
WHERE Student.Sno=SC.Sno  AND  SC.Cno='81002' AND SC.Grade>90;
```

```
SELECT Student.Sno,Sname
FROM Student
INNER JOIN SC ON Student.Sno=SC.Sno
WHERE SC.Cno='81002' AND SC.Grade>90;
```

- 内连接：join 或 inner join
- 左连接：left join 或 left outer join
- 右连接：right join 或 right outer join
- 外连接：full join 或 full outer join
- 笛卡尔积：cross join

学 号 Sno	姓 名 Sname	性 别 Ssex	年 龄 Sage	所在系 Sdept
200215121	李勇	男	20	CS
200215122	刘晨	女	19	CS
200215123	王敏	女	18	MA
200515125	张立	男	19	IS

Student

学 号 Sno	课程号 Cno	成绩 Grade
200215121	1	92
200215121	2	85
200215121	3	88
200215122	2	90
200215122	3	80

SC

课程号 Cno	课程名 Cname	先行课 Cpno	学分 Ccredit
1	数据库	5	4
2	数学		2
3	信息系统	1	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理		2
7	PASCAL语言	6	4

Course

查询选修了数据库课程的学生信息，给出学号，姓名，成绩

课程号 Cno	课程名 Cname	先行课 Cpno	学分 Ccredit
1	数据库	5	4
2	数学		2
3	信息系统	1	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理		2
7	PASCAL语言	6	4

课程号 Cno	课程名 Cname	先行课 Cpno	学分 Ccredit
1	数据库	5	4
2	数学		2
3	信息系统	1	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理		2
7	PASCAL语言	6	4

例：查询每门课程的先修课的课程名

Select C1.cno , C1.cpno/C2.cno, C2. cname

From course c1, course c2

Where c1. cpno=c2.cno

- **自连接**：**一个关系与自身做笛卡尔积**

课程号 Cno	课程名 Cname	先行课 Cpno	学分 Ccredit
1	数据库	5	4
2	数学		2
3	信息系统	1	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理		2
7	PASCAL语言	6	4

例：查询每门课程的先修课的课程名

Enter some SQL

```
1 select * from course;
2
3
4 Select c1.cno, c2.cno, c2.cname
5 from course c1, course c2
6 where c1.cpno=c2.cno;
```

Execute | Save the db | Load an

- **单表查询**
- **连接查询**
- **嵌套查询**：连接查询可以采用嵌套查询来实现，嵌套查询可以实现更为复杂的查询

➢**嵌套查询**：将一个**查询块**嵌套在另一个查询块的WHERE子句，FROM子句或者HAVING短语中的查询

- 查询块：一个SELECT-FROM-WHERE语句

```
SELECT Sname                    /*外层查询/父查询*/
FROM Student
WHERE Sno IN
        （SELECT Sno            /*内层查询/子查询*/
          FROM SC
          WHERE Cno= ' 2 '）；
```

```
SELECT Sno，Sname
FROM    Student
WHERE Sno  IN
        (SELECT Sno
         FROM    SC
        WHERE  Cno IN
                (SELECT Cno
                  FROM Course
                 WHERE Cname= ‘数据库’
                )
         );
```

- 当内层查询的查询结果是一个值时，使用=,>,<,!=等比较运算符
- 当内层查询的查询结果是一个集合时，使用IN、NOT IN、ANY/SOME、ALL

> ANY	大于子查询结果中的某个值
> ALL	大于子查询结果中的所有值
< ANY	小于子查询结果中的某个值
< ALL	小于子查询结果中的所有值
>= ANY	大于等于子查询结果中的某个值
>= ALL	大于等于子查询结果中的所有值

……

查询非计算机科学技术专业中比计算机科学技术专业任意一个学生年龄小（出生日期晚）的学生的姓名和专业

1）学生关系模式：Student(Sno, Sname, Ssex, Sbirthdate, Smajor)
包括学号、姓名、性别、出生日期和主修专业等属性

学号 Sno	姓名 Sname	性别 Ssex	出生日期 Sbirthdate	主修专业 Smajor
20180001	李勇	男	2000-3-8	信息安全
20180002	刘晨	女	1999-9-1	计算机科学与技术
20180003	王敏	女	2001-8-1	计算机科学与技术
20180004	张立	男	2000-1-8	计算机科学与技术
20180005	陈新奇	男	2001-11-1	信息管理与信息系统
20180006	赵明	男	2000-6-12	数据科学与大数据技术
20180007	王佳佳	女	2001-12-7	数据科学与大数据技术

查询非计算机科学技术专业中比计算机科学技术专业任意一个学生年龄小（出生日期晚）的学生的姓名和专业

```
SELECT Sname, Smajor
   FROM    Student
   WHERE Sbirthdate > ANY (SELECT  Sbirthdate
                           FROM    Student
                           WHERE Smajor= '计算机科学与技术')
   AND Smajor <> '计算机科学与技术 ';          /*父查询块中的条件 */
```

```
SELECT Sname, Smajor
  FROM   Student
  WHERE Sbirthdate > ANY (SELECT  Sbirthdate
                          FROM   Student
                          WHERE Smajor= '计算机科学与技术')
 AND Smajor <> '计算机科学与技术 ';        /*父查询块中的条件 */
```

```
SELECT Sname,Sbirthdate, Smajor
   FROM   Student
   WHERE Sbirthdate >
                    (SELECT MIN(Sbirthdate)
                     FROM Student
                     WHERE Smajor= '计算机科学与技术')
    AND Smajor <>'计算机科学与技术';
```

ANY（或SOME），ALL与聚集函数、IN的等价转换关系

	=	<>或!=	<	<=	>	>=
ANY	IN	--	<MAX	<=MAX	>MIN	>= MIN
ALL	--	NOT IN	<MIN	<= MIN	>MAX	>= MAX

嵌套查询

- **不相关子查询**：子查询的查询条件不依赖父查询
- **相关子查询**：子查询的查询条件依赖父查询

```
SELECT Sname                    /*外层查询/父查询*/
FROM Student
WHERE Sno IN
        （SELECT Sno            /*内层查询/子查询*/
          FROM SC
          WHERE Cno= ' 2 '）；
```

- 查询各院系中年龄小于平均年龄的学生信息

学 号 Sno	姓 名 Sname	性 别 Ssex	年 龄 Sage	所在系 Sdept
200215121	李勇	男	20	CS
200215122	刘晨	女	19	CS
200215123	王敏	女	18	MA
200515125	张立	男	19	IS

```
Select sno，sname
From student s1
Where  sage <(select  avg(sage)
                from student s2
                where s2.sdept=s1.dept)
```

```
Select sno，sname
From student
Where  sage <(select  avg(sage)
                from student )
```

- 查询每位同学的平均成绩
- 查询每个学生超过自己选修课程平均成绩的课程号

学号 Sno	课程号 Cno	成绩 Grade
200215121	1	92
200215121	2	85
200215121	3	88
200215122	2	90
200215122	3	80

SC

- 查询每个学生超过他自己选修课程平均成绩的课程号

```
SELECT Sno, Cno

FROM SC SC1

WHERE Grade>

(SELECT AVG(Grade) FROM SC SC2

WHERE SC2.SNO= SC1.SNO)
```

- **单表查询**
- **连接查询**
- **嵌套查询**：连接查询可以采用嵌套查询来实现，嵌套查询可以实现更为复杂的查询
  - =,>,<,!=
  - IN、NOT IN、ANY/SOME、ALL
  - Exists/not Exists

➢**Exists：** 判断子查询返回的结果是否存在元组

- 存在元组（非空），则外层的WHERE子句返回True
- 不存在元组（空），则外层的WHERE子句返回False

```
SELECT Sname                    /*外层查询/父查询*/
FROM Student
WHERE Sno IN
        （SELECT Sno            /*内层查询/子查询*/
          FROM SC
          WHERE Cno= ' 2 '）；
```

查询选修了2号课程的学生学号，姓名

学 号 Sno	姓 名 Sname	性 别 Ssex	年 龄 Sage	所在系 Sdept
200215121	李勇	男	20	CS
200215122	刘晨	女	19	CS
200215123	王敏	女	18	MA
200515125	张立	男	19	IS

Student

学 号 Sno	课程号 Cno	成绩 Grade
200215121	1	92
200215121	2	85
200215121	3	88
200215122	2	90
200215122	3	80

SC

```
SELECT Sname
FROM Student
WHERE EXISTS
        (SELECT *
         FROM SC
         WHERE Sno=Student.Sno AND Cno= ‘ 2 ');
```

- Not Exists： 子查询的结果集
  - ✓不存在元组（空），则外层的WHERE子句返回True
  - ✓存在元组（非空），则外层的WHERE子句返回False

查询没有选修2号课程的学生姓名

```
SELECT Sname
FROM Student
WHERE NOT EXISTS
        (SELECT *
         FROM SC
         WHERE Sno = Student.Sno AND Cno='2');
```

- **查询15121学生没有选修的课程号**

Select cno

From course

Where not exists (select *

from sc
where cno=course.cno
and sno='15121')

学号 Sno	课程号 Cno	成绩 Grade
200215121	1	92
200215121	2	85
200215121	3	88
200215122	2	90
200215122	3	80

课程号 Cno	课程名 Cname	先行课 Cpno	学分 Ccredit
1	数据库	5	4
2	数学		2
3	信息系统	1	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理		2
7	PASCAL语言	6	4

```
Select  cno
From  course
Where  not  exists
          (select  *
          from  SC
         where sno=  ‘15121’  and cno =course.cno)
```

➢EXISTS

- ➢所有带IN、比较运算符、ANY和ALL的子查询都能用带EXISTS谓词的子查询等价替换
- ➢一些带EXISTS或NOT EXISTS的子查询不能被其他形式的子查询等价替换

——集合包含运算：R=S，R⊆S

例：查询选修了全部课程的学生的学号、姓名

——课程表中不存在一门课，该学生没有选修

- 带有全称量词的谓词：转换为等价的带有存在量词的谓词：

$$(\forall x)\ P \equiv \neg\ (\exists x\ (\neg P))$$

学 号 Sno	姓 名 Sname	性 别 Ssex	年 龄 Sage	所在系 Sdept
200215121	李勇	男	20	CS
200215122	刘晨	女	19	CS
200215123	王敏	女	18	MA
200515125	张立	男	19	IS

学号 Sno	课程号 Cno	成绩 Grade
200215121	1	92
200215121	2	85
200215121	3	88
200215122	2	90
200215122	3	80

课程号 Cno	课程名 Cname	先行课 Cpno	学分 Ccredit
1	数据库	5	4
2	数学		2
3	信息系统	1	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理		2
7	PASCAL语言	6	4

对于15121学生，找其没有选修的课程

```
Select  cno

From course

Where not  exists

            (select *
             from   sc
             where  sno=  '15121'  and
cno= course.cno)
```

学 号 Sno	姓 名 Sname	性 别 Ssex	年 龄 Sage	所在系 Sdept
200215121	李勇	男	20	CS
200215122	刘晨	女	19	CS
200215123	王敏	女	18	MA
200515125	张立	男	19	IS

学号 Sno	课程号 Cno	成绩 Grade
200215121	1	92
200215121	2	85
200215121	3	88
200215122	2	90
200215122	3	80

课程号 Cno	课程名 Cname	先行课 Cpno	学分 Ccredit
1	数据库	5	4
2	数学		2
3	信息系统	1	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理		2
7	PASCAL语言	6	4

1 对于某个学生，找其没有选修的课程

2 如果不存在：没有选修的课程，则该学生满足要求

```
Select  cno
From  course
Where  not  exists
          (select  *
           from  SC
          where sno=  '*****'  and cno =course.cno
```

```
5
6 select sno,sname
7 from student
8 where not exists
9   (select * from course
0    where not exists
1    (select * from sc where sc.sno=student.sno and sc.cno=course.cno)
2    )
```

例：查询至少选修了学生20180002选修的全部课程的学生的学号

——**不存在一门课，**学生20180002选修了，而学生x没有选修

- **单表查询**
- **连接查询**
- **嵌套查询**：连接查询可以采用嵌套查询来实现，嵌套查询可以实现更为复杂的查询
  - =,>,<,!=
  - IN、NOT IN、ANY/SOME、ALL
  - Exists/not Exists
- **集合查询**

➢集合查询（UNION、EXCEPT、INTERSECT）

```
SELECT Sno
FROM SC
WHERE Cno='81001'
INTERSECT
SELECT Sno
FROM SC
WHERE Cno='81002';
```

➢**嵌套查询**：将一个**查询块**嵌套在另一个查询块的WHERE子句，FROM子句或者HAVING短语中的查询

➢From子句中的子查询——派生表

——子查询生成的临时派生表（derived table）成为主查询的查询对象

——派生表是一个中间结果表，查询完成后派生表将被系统自动清除

Select

From table、子查询

Where

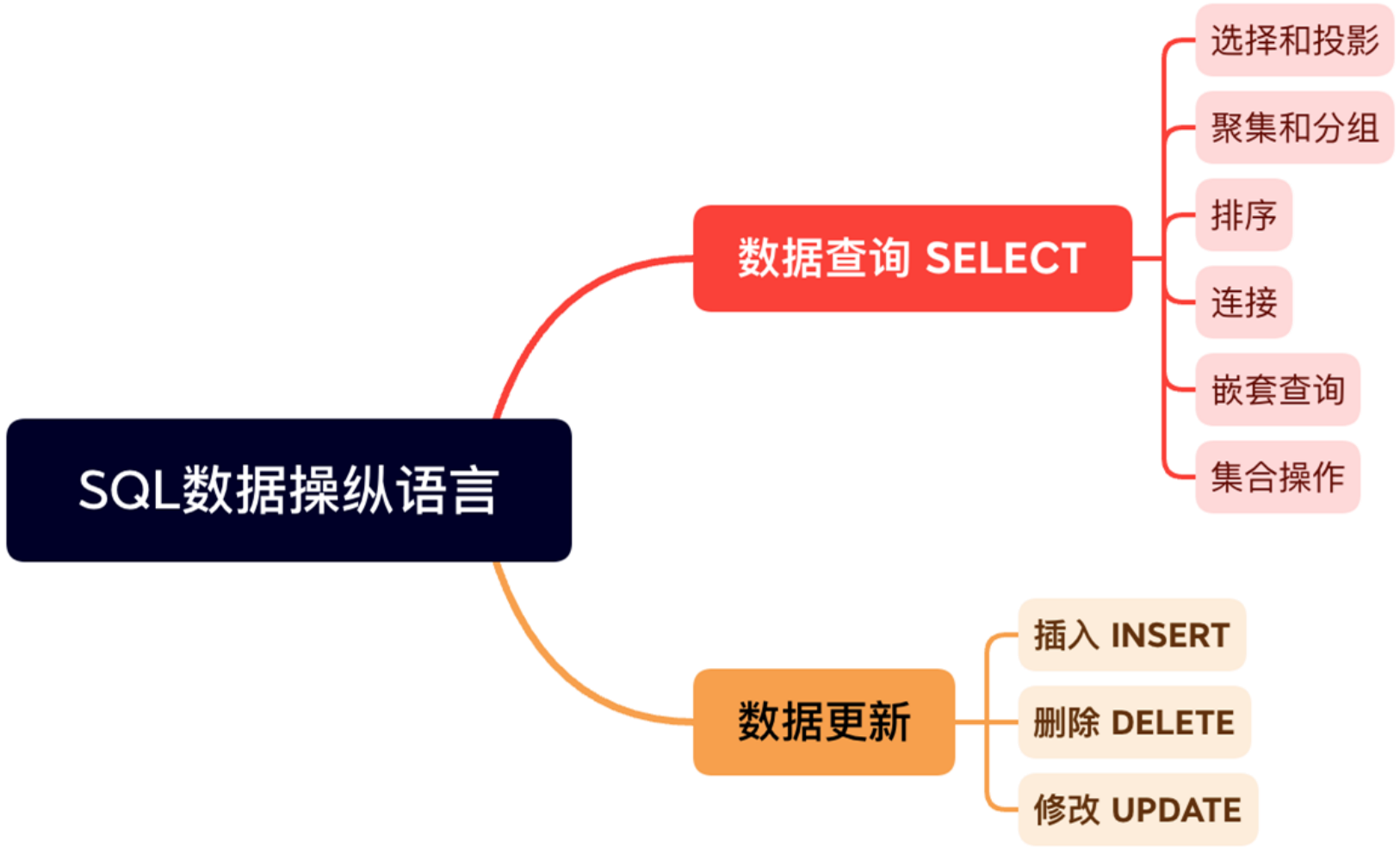
查询每个学生超过他自己选修课程平均成绩的课程号

学号 Sno	课程号 Cno	成绩 Grade
200215121	1	92
200215121	2	85
200215121	3	88
200215122	2	90
200215122	3	80

```
SELECT Sno, Cno
FROM SC SC1
WHERE Grade>
(SELECT AVG(Grade) FROM SC SC2
WHERE SC2.SNO= SC1.SNO)
```

```
SELECT Sno, Cno
FROM SC, (SELECT Sno, Avg(Grade) FROM SC GROUP BY Sno)
        AS Avg_SC(Avg_sno,Avg_grade)
WHERE SC.Sno = Avg_SC.Avg_sno AND SC.Grade >= Avg_SC.Avg_grade;
```

2）DML：数据操纵语言：**增加、修改、删除、查询**



Insert

1）新增一条元组

2）新增子查询的结果

```
INSERT
INTO <表名>  [(<属性列1> [，<属性列2>…  )]
子查询
```

[例2] 对每一个系，求学生的平均年龄，并把结果存入数据库。

第一步：建表

```
CREATE  TABLE  Dept_age
   (Sdept  CHAR(15)          /* 系名*/
    Avg_age SMALLINT);       /*学生平均年龄*/
```

第二步：插入数据

```
INSERT
INTO  Dept_age(Sdept，Avg_age)
    SELECT  Sdept，AVG(Sage)
    FROM  Student
    GROUP BY Sdept;
```

Update

1）修改1条/多条元组

2）带子查询的修改

```
UPDATE SC
        SET Grade=0
        WHERE Sno IN
                ( SELECT Sno
                FROM Student
                WHERE Smajor= '计算机科学与技术');
```

Delete

1）删除1条/多条元组

2）带子查询的删除

```
DELETE FROM SC
        WHERE Sno IN
                ( SELECT Sno
                FROM Student
                WHERE Smajor= '计算机科学与技术' );
```

空值的处理

空值就是“不知道”、“不存在”或“无意义”的值。

——空值约束：在创建基本表时，属性约束NOT NULL，则该属性不能取空值

```
INSERT INTO SC(Sno,Cno,Grade,Semester,Teachingclass)
    VALUES('20180006', '81004',NULL, '20211',NULL);

INSERT INTO SC(Sno,Cno,Semester)
    VALUES('20180006', '81004','20211');
```

➢空值的判断： IS NULL或IS NOT NULL

SELECT *

FROM SC

WHERE grade IS NULL

SELECT Sno

FROM SC

WHERE Grade < 60 AND Cno='81001’;

■空值与另一个值（包括另一个空值）的算术运算的结果为空值

■空值与另一个值（包括另一个空值）的比较运算的结果为UNKNOWN

■逻辑运算（TRUE，FALSE， UNKNOWN ）

选出选修81001号课程且成绩不及格的学生以及缺考的学生

```
SELECT Sno
FROM SC
WHERE Grade < 60 AND Cno='81001'
UNION
SELECT Sno
FROM SC
WHERE Grade IS NULL AND Cno='81001';
```

x	y	x AND y	x OR y	NOT x
T	T	T	T	F
T	U	U	T	F
T	F	F	T	F
U	T	U	T	U
U	U	U	U	U
U	F	F	U	U
F	T	F	T	T
F	U	F	U	T
F	F	F	F	T

表示FALSE，U表示UNKNOWN

```
SELECT Sno
FROM SC
WHERE Cno='81001' AND (Grade < 60 OR Grade IS NULL);
```

1）DDL：数据定义语言：**创建、修改、删除**

	创建	修改	删除
用户	CREATE USER	ALTER USER	DROP USER
表空间	CREATE TABLESPACE	ALTER TABLESPACE	DROP TABLESPACE
数据库	CREATE DATABASE	ALTER DATABASE	DROP DATABASE
模式	CREATE SCHEMA	ALTER SCHEMA	DROP SCHEMA
表	CREATE TABLE	ALTER TABLE	DROP TABLE
索引	CREATE INDEX	ALTER INDEX	DROP INDEX
视图	CREATE VIEW	ALTER VIEW	DROP VIEW
序列	CREATE SEQUENCE	ALTER SEQUENCE	DROP SEQUENCE

视图view

- **虚表**
  - 将重复执行的SQL语句存储起来，并对其命名
  - 只存放视图的定义（SQL语句），使用view的时候执行SQL语句

```
CREATE VIEW <视图名> [(<列名> [,<列名>, …, <列名>])] AS <子查询>;
```

```
CREATE VIEW IS_Student
        AS
        SELECT Sno,Sname,Ssex,Sage, Smajor
        FROM Student
        WHERE Smajor='计算机科学';
```

```
CREATE VIEW IS_Student
        AS
        SELECT Sno,Sname,Ssex,Sage, Smajor
        FROM Student
        WHERE Smajor='计算机科学';
```

- 基本表中的数据发生变化，从视图中查询出的数据也随之改变
- 视图也可以建立在多个基本表上
- 视图也可以建立在一个或多个已定义好的视图上，或建立在基本表与视图上
- 视图一旦定义，将保存在数据字典，之后的所有查询都可以直接引用该视图
- 删除视图：DROP VIEW <视图名> [CASCADE];

- 在视图上查询

查询计算机科学专业年龄小于20的学生

```
SELECT Sno,
FROM IS_Student
WHERE  sage<20;
```

```
CREATE VIEW IS_Student
        AS
        SELECT Sno,Sname,Ssex,Sage, Smajor
        FROM Student
        WHERE Smajor='计算机科学';
```

视图消解：

```
SELECT Sno,Sbirthdate
FROM Student
WHERE Smajor='计算机科学' AND Sage<20
```

- 在视图上查询

查询选修了81001号课程的计算机科学专业的学生

```
CREATE VIEW IS_Student
        AS
        SELECT Sno,Sname,Ssex,Sage, Smajor
        FROM Student
        WHERE Smajor='计算机科学';
```

```
SELECT IS_Student.Sno,Sname
FROM IS_Student，SC
WHERE IS_Student.Sno=SC.Sno AND SC.Cno='81001';
```

```
CREATE VIEW S_GradeAVG(Sno,Gavg)
AS
SELECT Sno, AVG(Grade)
FROM SC
GROUP BY Sno;
```

- 视图消解法的局限

查询平均成绩在90分以上的学生学号和平均成绩

```
SELECT *
FROM  S_GradeAVG
WHERE Gavg>=90;
```

```
SELECT Sno, AVG(Grade)
FROM     SC
WHERE  AVG(Grade)>=90
GROUP BY Sno;
```

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

➢**更新视图**：转换为对基本表的更新

```
CREATE VIEW IS_Student
        AS
        SELECT Sno,Sname,Ssex,Sage, Smajor
        FROM Student
        WHERE Smajor='计算机科学'
        With  check option
```

WITH CHECK OPTION表示对视图进行UPDATE，INSERT和DELETE时保证视图定义中的条件表达式

```
CREATE VIEW IS_Student
        AS
        SELECT Sno,Sname,Ssex,Sage, Smajor
        FROM Student
        WHERE Smajor=‘计算机科学’
```

With check option

```
UPDATE IS_Student
SET Sname=‘*****'
WHERE Sno='20180005';
```

```
UPDATE Student
SET Sname=‘****'
WHERE Sno=‘20180005’ AND Smajor=‘计算机科学';
```

```
CREATE VIEW IS_Student
        AS
        SELECT Sno,Sname,Ssex,Sage, Smajor
        FROM Student
        WHERE Smajor=‘计算机科学’
```

With check option

```
UPDATE IS_Student
SET Sname=‘*****'
WHERE Sno='20180005';
```

```
Select * from
Is_student
```

➢**更新视图**：并不是所有的视图都可更新

```
CREATE VIEW S_GradeAVG(Sno,Gavg)
AS
SELECT Sno, AVG(Grade)
FROM SC
GROUP BY Sno;
```

```
UPDATE S_GradeAVG
 SET Gavg=90
 WHERE Sno='20180001';
```

一般地，行列子集视图是可更新的：若一个视图是从单个基本表导出，并且只是去掉了基本表的某些行和某些列，但保留了主码，则称这类视图为行列子集视图

视图的作用

- 视图的作用：
  - ✓简化用户的查询操作
  - ✓提供一定的安全性
  - ✓提供一定的逻辑独立性
- 对视图的大量查询会有较高的成本，某些DBMS支持物化视图（MATERIALIZED VIEW）

1）DDL：数据定义语言：**创建、修改、删除**

	创建	修改	删除
用户	CREATE USER	ALTER USER	DROP USER
表空间	CREATE TABLESPACE	ALTER TABLESPACE	DROP TABLESPACE
数据库	CREATE DATABASE	ALTER DATABASE	DROP DATABASE
模式	CREATE SCHEMA	ALTER SCHEMA	DROP SCHEMA
表	CREATE TABLE	ALTER TABLE	DROP TABLE
索引	CREATE INDEX	ALTER INDEX	DROP INDEX
视图	CREATE VIEW	ALTER VIEW	DROP VIEW
序列	CREATE SEQUENCE	ALTER SEQUENCE	DROP SEQUENCE

Schema：模式/架构

- 早期的DBMS：DB——用于所有关系的命名空间

  —— 多User使用：命名冲突

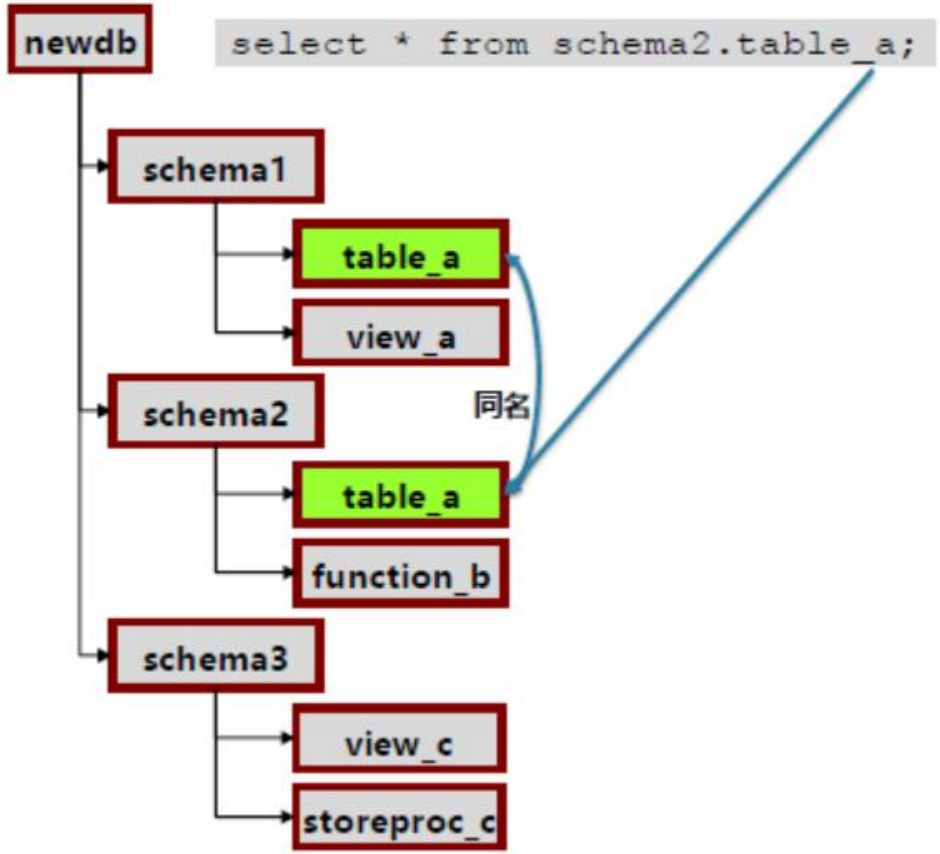
- Schema：包含关系、视图、索引等数据库对象的一个命名空间

——创建模式：create schema <模式名>

SQLserver： create schema <模式名>

MySQL： create database = create schema

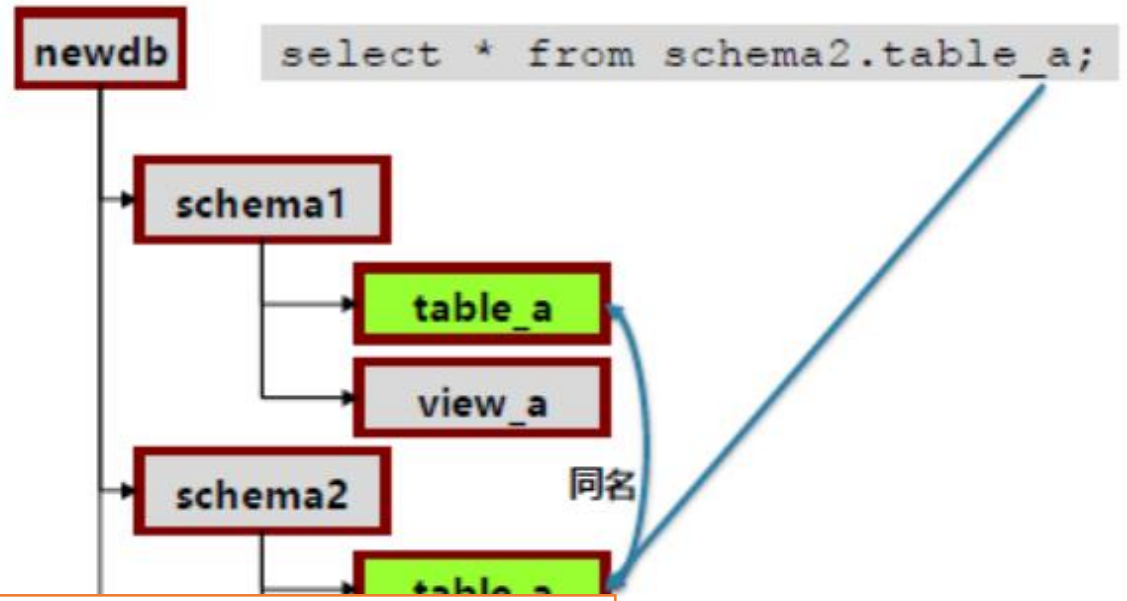
Oracle：create user <用户名>（生成一个与user同



SQLserver： create schema

MySQL： create database =

Oracle： create user <用户名



<模式名>.<表名>

用户在缺省的schema中操作，可以省略模式名

访问非缺省的schema中的对象，必须指定模式名

作业

5．针对习题 4 中的 4 个表试用 SQL 完成以下各项操作：

（1）找出所有供应商的姓名和所在城市；

（2）找出所有零件的名称、颜色、重量；

（3）找出使用供应商 Sl 所供应零件的工程号码；

（4）找出工程项目 J2 使用的各种零件的名称及其数量；

（5）找出上海厂商供应的所有零件号码；

（6）找出使用上海产的零件的工程名称；

（7）找出没有使用天津产的零件的工程号码；

（8）把全部红色零件的颜色改成蓝色；

（9）由 S5 供给 J4 的零件 P6 改为由 S3 供应，请作必要的修改；

（10）从供应商关系中删除 S2 的记录，并从供应情况关系中删除相应的记录；

（11）请将（S2，J6，P4，200）插入供应情况关系。

作业

9．请为三建工程项目建立一个供应情况的视图，包括供应商代码（SNO）、零件代码（PNO）、供应数量（QTY）。针对该视图完成下列查询：

（1）找出三建工程项目使用的各种零件代码及其数量；

（2）找出供应商 S1 的供应情况。