2025年春季学期

数据库系统概论

An Introduction to Database Systems

第三章 关系数据库标准语言SQL

中国科学技术大学
人工智能与数据科学学院

黄振亚，huangzhy@ustc.edu.cn

第三章 关系数据库标准语言SQL

3.1 SQL概述

- **SQL（Structured Query Language）**

  结构化查询语言，是关系数据库的标准语言

- **SQL是一个通用的、功能极强的关系数据库语言**

SQL概述（续）

- 3.1.1 SQL 的产生与发展
- 3.1.2 SQL的特点
- 3.1.3 SQL的基本概念

SQL标准的进展过程

标准	篇幅（约）	发布日期
SQL/86		1986.10
SQL/89（FIPS 127-1）	120页	1989年
SQL/92	622页	1992年
SQL99（SQL 3）	1700页	1999年
SQL2003	3600页	2003年
SQL2008	3777页	2008年
SQL2011	3817页	2011年
SQL2016	4035页	2016年

目前，没有一个数据库系统能够支持SQL标准的所有概念和特性

http://www.sac.gov.cn/

3.1 SQL概述

- 3.1.1 SQL 的产生与发展
- 3.1.2 SQL的特点
- 3.1.3 SQL的基本概念

3.1.2 SQL的特点

1.综合统一

2.高度非过程化

3.面向集合的操作方式

4.以同一种语法结构提供多种使用方式

5.语言简洁，易学易用

1. 综合统一

- **SQL特点：综合统一**
  - 集数据定义语言（DDL），数据操纵语言（DML），数据控制语言（DCL）功能于一体。
  - 可以独立完成数据库生命周期中的全部活动：
    - 定义关系模式，插入数据，建立数据库；
    - 对数据库中的数据进行查询和更新；
    - 数据库重构和维护
    - 数据库安全性、完整性控制等
  - 用户数据库投入运行后，可根据需要随时逐步修改模式，不影响数据的运行。
  - 数据操作符统一（数据结构：关系）

2.高度非过程化

- **SQL特点：高度非过程化**
  - 非关系数据模型的数据操纵语言“面向过程”，必须制定存取路径
  - SQL只要提出“做什么”，无须了解存取路径。
  - 存取路径的选择以及SQL的操作过程由系统自动完成。
  - 4GL：面向问题，描述性语言

3.面向集合的操作方式

- SQL特点：面向集合的操作方式
  - 非关系数据模型采用面向记录的操作方式，操作对象是一条记录
  - SQL采用集合操作方式
    - 操作对象、查找结果可以是元组的集合
    - 一次插入、删除、更新操作的对象可以是元组的集合

4.以同一种语法结构提供多种使用方式

- SQL特点：同一种语法结构，多种使用方式
  - SQL是独立的语言
    - 能够独立地用于联机交互的使用方式
  - SQL又是嵌入式语言
    - SQL能够嵌入到高级语言（例如C，C++，Java）程序中，供程序员设计程序时使用
      - JDBC，ODBC，第八章

5.语言简洁，易学易用

- SQL特点：语言简介，易学易用
  - SQL功能极强，完成核心功能只用了9个动词。

表3.2　SQL 完成核心功能的9 个动词

SQL 功　能	动　　词
数 据 定 义	CREATE, DROP, ALTER
数 据 查 询	SELECT
数 据 操 纵	INSERT, UPDATE, DELETE
数 据 控 制	GRANT, REVOKE

3.1 SQL概述

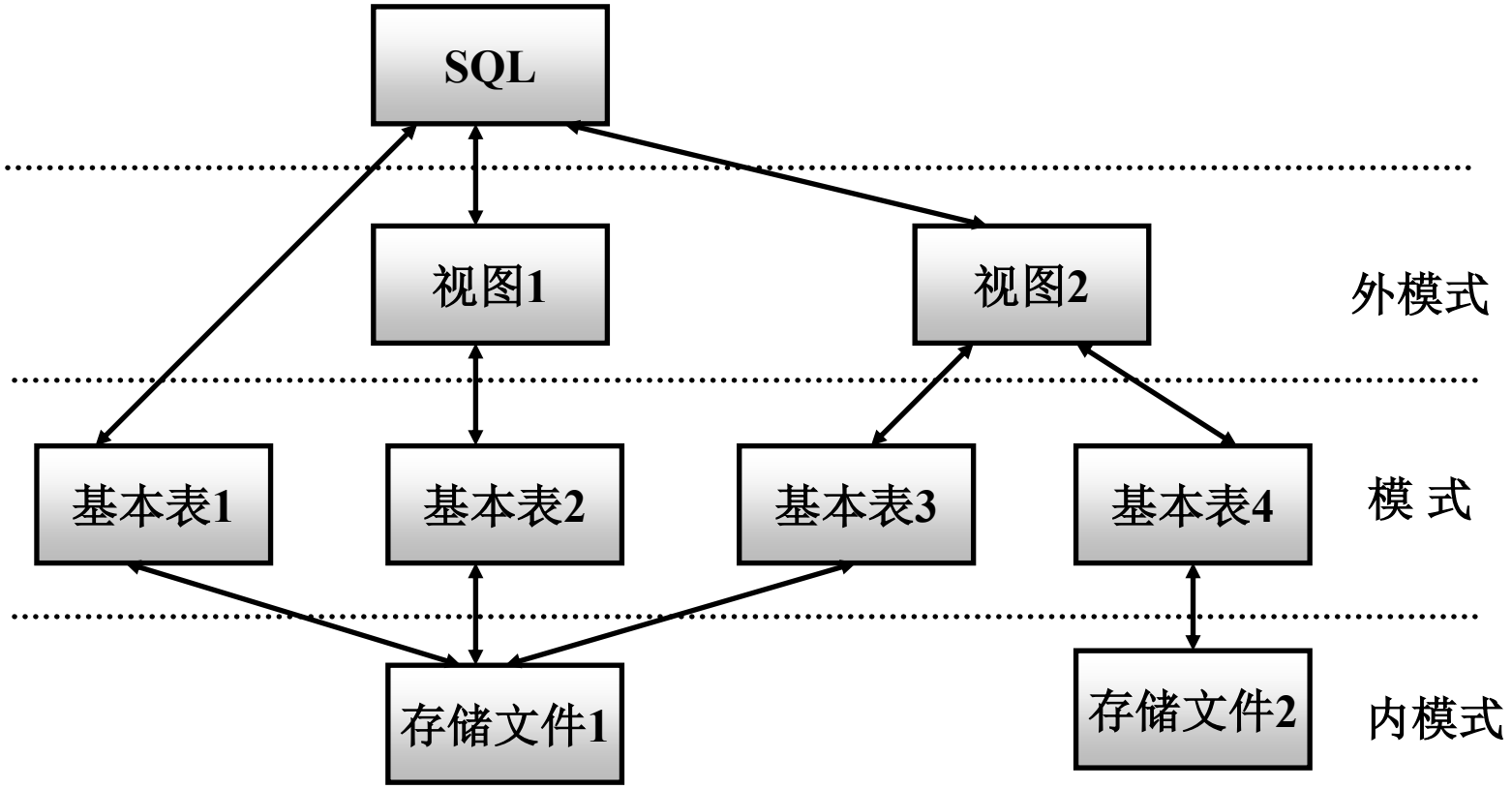
- 3.1.1 SQL 的产生与发展
- 3.1.2 SQL的特点
- 3.1.3 SQL的基本概念

3.1.3 SQL的基本概念

- **SQL语言支持关系数据库三级模式结构**
  - **外模式：视图（View）和部分基本表（Base table）**
  - **模式：若干基本表（Base table）**
  - **内模式：存储文件（stored file）**

3.1.3 SQL的基本概念

SQL语言支持关系数据库三级模式结构



3.1.3 SQL的基本概念（续）

- 基本表
  - 本身独立存在的表，关系表，是实表
  - SQL中一个关系就对应一个基本表
  - 一个(或多个)基本表对应一个存储文件
  - 一个表可以带若干索引
- 存储文件
  - 逻辑结构组成了关系数据库的内模式
  - 物理结构是任意的，对用户透明
- 视图
  - 从一个或几个基本表导出的表
  - 数据库中只存放视图的定义而不存放视图对应的数据
  - 视图是一个虚表
  - 用户可以在视图上再定义视图
  - 例如，查询语句执行之后的结果

第三章 关系数据库标准语言SQL

3.2 学生-课程 数据库

- 数据定义、数据操纵、数据查询、数据控制
- 学生-课程模式 S-T：

  学生表：Student(Sno,Sname,Ssex,Sage,Sdept)

  课程表：Course(Cno,Cname,Cpno,Ccredit)

  学生选课表：SC(Sno,Cno,Grade,Semester)

Student表

Student表

学 号 Sno	姓 名 Sname	性 别 Ssex	年 龄 Sage	所在系 Sdept
20180001	李勇	男	20	CS
20180002	刘晨	女	19	CS
20180003	王敏	女	18	MA
20180004	张立	男	19	IS
20180005	陈新奇	男	19	DS

Course表

Course表

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	Python语言	6	4
8	数据分析及实践	6	3

SC表

学 号 Sno	课程号 Cno	成绩 Grade	开课学期 Semester
20180001	1	92	20192
20180001	2	85	20201
20180001	3	88	20202
20180002	2	90	20192
20180002	3	80	20201

第三章 关系数据库标准语言SQL

3.3 数据定义

- SQL的数据定义功能: 模式定义、表定义、视图和索引的定义

表3.3 SQL 的数据定义语句

操作对象	操作方式		
	创建	删除	修改
数据库模式	CREATE SCHEMA	DROP SCHEMA	
表	CREATE TABLE	DROP TABLE	ALTER TABLE
视图	CREATE VIEW	DROP VIEW	
索引	CREATE INDEX	DROP INDEX	ALTER INDEX

- ✓ 一个RDBMS的实例可以建立多个数据库
- ✓ 一个数据库可以建立多个模式
- ✓ 一个模式下包含多个表、视图、索引等数据库对象

数据库（有的系统称为目录）

⇩

模式

⇩

表以及视图、索引等

3.3 数据定义

- 3.3.1 模式的定义与删除
- 3.3.2 基本表的定义、删除与修改
- 3.3.3 索引的建立与删除
- 3.3.4 数据字典

一. 定义模式（续）

- 定义模式：

  CREATE SCHEMA <模式名> AUTHORIZATION <用户名>;

  [<表定义子句>|<视图定义子句>|<授权定义子句>]

  - 如果没有指定<模式名>，隐含为<用户名>
  - 用户拥有DBA权限，或者获得DBA授予的CREATE SCHEMA权限

一. 定义模式（续）

[例3.1] 定义一个学生-课程模式S-C-SC

CREATE SCHEMA `S-C-SC` AUTHORIZATION WANG;

为用户WANG定义了一个模式S-C-SC

- 用户WANG必须是已经存在的数据库用户

[例3.2] CREATE SCHEMA AUTHORIZATION WANG;

<模式名>隐含为用户名WANG

- 如果没有指定<模式名>，那么<模式名>隐含为<用户名>

注：不同的数据库系统/软件对SQL标准（命令集）的支持情况不同，也可能会有些修改或扩充，例如，在mysql中给用户授予某个schema/database的权限不是用authorization，而是用grant

一. 定义模式（续）

[例3.1] 定义一个学生-课程模式S-C-SC (Mysql 例子)

```
CREATE SCHEMA `S-C-SC`;
```

- root用户默认拥有模式中的全部权限

```
GRANT ALL ON *.* TO 'WANG'@'%';
```

- Grant指令把模式权限分配给一个已存在的用户WANG

```
GRANT <权限> ON <模式名>.<数据表> TO <用户名> @ <主机名>;
```

注：不同的数据库系统/软件对SQL标准（命令集）的支持情况不同，也可能会有些修改或扩充，例如，在mysql中给用户授予某个schema/database的权限不是用authorization，而是用grant

一. 定义模式（续）

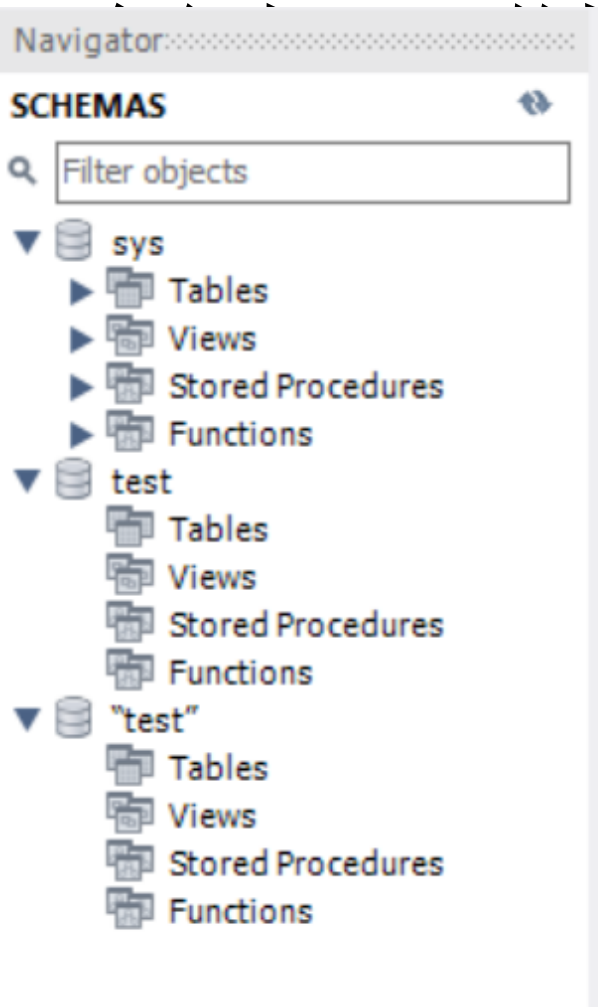
- 创建一个名为TEST的模式，哪一个是正确的？
  - CREATE SCHEMA “TEST”; --- “TEST”
  - CREATE SCHEMA `TEST`; --- TEST
  - CREATE SCHEMA TEST; --- TEST
  - CREATE DATABASE TEST; --- TEST
- 创建一个名为S-T的模式呢？
  - CREATE SCHEMA S-T

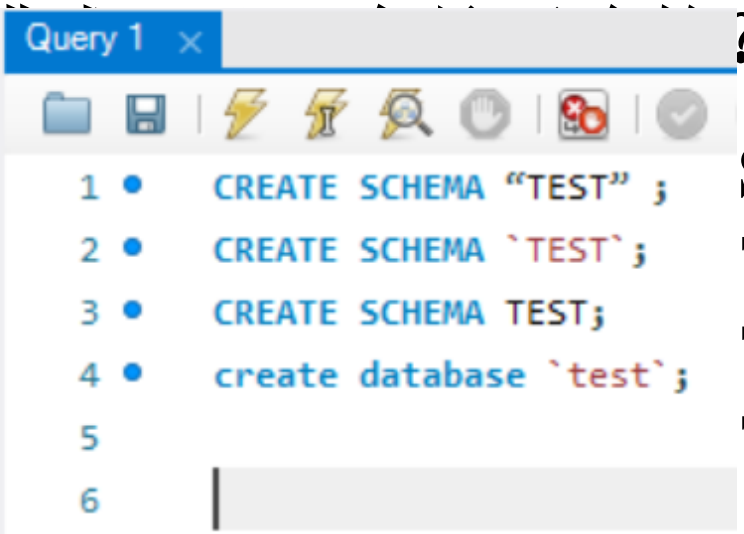
分号是个分隔符

PS：若程序里面写sql,就不要加分号，在程序里面编译器会把分号当做sql本身的一部分，所以会报错；

一. 定义模式（续）

- 创建
  - Cl
  - Cl
  - Cl
  - Cl





?

ST"

T

T

T

一. 定义模式（续）

CREATE SCHEMA <模式名> AUTHORIZATION <用户名>
[<表定义子句>|<视图定义子句>|<授权定义子句>]

- 定义模式实际上定义了一个命名空间
  - 在这个空间中可以定义该模式包含的数据库对象，例如基本表、视图、索引等。
  - 在CREATE SCHEMA中可以接受CREATE TABLE，CREATE VIEW和GRANT子句

一. 定义模式（续）

[例3.3] 为用户ZHANG创建了一个模式TEST，并在其中定义了一个表TAB1。

```
CREATE SCHEMA TEST AUTHORIZATION ZHANG
   CREATE TABLE TAB1(COL1 SMALLINT，
                     COL2 INT，
                     COL3 CHAR(20)，
                     COL4 NUMERIC(10，3)，
                     COL5 DECIMAL(5，2)
                    );
```

一. MySQL中的实际操作

- CREATE SCHEMA `TEST`;

  Use TEST; **#Otherwise Default Schema**

  CREATE TABLE TAB1 (COL1 smallint,
  COL2 INT,
  COL3 CHAR(20),
  COL4 NUMERIC(10,3),
  COL5 DECIMAL(5,2)
  );

二、删除模式

- 删除模式：

DROP SCHEMA <模式名> <CASCADE|RESTRICT>;

- **CASCADE**(级联)
  - 删除模式的同时把该模式中所有的数据库对象全部删除
- **RESTRICT**(限制)
  - 如果该模式中定义了下属的数据库对象（如表、视图等），则拒绝该删除语句的执行。
  - 当该模式中没有任何下属的对象时 才能执行。

删除模式（续）

[例3.4] DROP SCHEMA ZHANG CASCADE;

删除模式ZHANG

同时，该模式中定义的表TAB1也被删除

MySQL中的实际操作

- 错误：drop SCHEMA `TEST` cascade;
- 正确： drop SCHEMA `TEST` ;
- 原因：
  - 在mysql中drop schema即可，不支持cascade
    drop schema会把这个schema下的table也一并删除

```
10  drop schema test cascade;
11  drop schema test restrict;
```

3.3 数据定义

- 3.3.1 模式的定义与删除
- 3.3.2 基本表的定义、删除与修改
- 3.3.3 索引的建立与删除
- 3.3.4 数据字典

3.3.2 基本表的定义、删除与修改

一、定义基本表

CREATE TABLE <表名>

（<列名> <数据类型>[<列级完整性约束条件>]

[，<列名> <数据类型>[<列级完整性约束条件>]]

…

[，<表级完整性约束条件>] ）；

- <表名>：所要定义的基本表的名字
- <列名>：组成该表的各个属性（列）
- <列级完整性约束条件>：涉及相应属性列的完整性约束条件
- <表级完整性约束条件>：涉及一个或多个属性列的完整性约束条件

✓ 如果完整性约束条件涉及到该表的多个属性列，则必须定义在表级上，否则既可以定义在列级也可以定义在表级

学生表Student

[例3.5] 建立“学生”表Student，学号是主码，姓名取值唯一

```
CREATE TABLE Student
    (Sno   CHAR(9) PRIMARY KEY，  /* 列级完整性约束条件*/
     Sname  CHAR(20) UNIQUE，      /* Sname取唯一值*/
     Ssex    CHAR(2)，
     Sage   SMALLINT，
     Sdept  CHAR(20)
    );
```

主码

UNIQUE 约束

学生表Student

[例3.5] 建立“学生”表Student，学号是主码，姓名取值唯一

Sno	Sname	Ssex	Sage	Sdept

字符型	字符型	字符型	整数	字符型
长度为 9	长度为 20	长度为 2		长度为 20

不能为空值

课程表Course

［例3.6］建立一个“课程”表Course

```
CREATE TABLE  Course
    ( Cno      CHAR(4) PRIMARY KEY,    /* 列级完整性约束条件*/
      Cname  CHAR(40),
      Cpno     CHAR(4) ,
      Ccredit  SMALLINT,
      FOREIGN KEY (Cpno) REFERENCES   Course(Cno)
       /* 表级完整性约束条件*/
      );
```

先修课

Cpno是外码
被参照表是Course
被参照列是Cno

学生选课表SC

[例3.7] 建立一个“学生选课”表SC

```
CREATE TABLE SC
        (Sno  CHAR(9),
        Cno  CHAR(4),
        Grade   SMALLINT,
        Semester  CHAR(5),
        PRIMARY KEY (Sno，Cno),
            /* 主码由两个属性构成，必须作为表级完整性进行定义*/
        FOREIGN KEY (Sno) REFERENCES Student(Sno),
            /* 表级完整性约束条件，Sno是外码，被参照表是Student */
        FOREIGN KEY (Cno) REFERENCES Course(Cno)
            /* 表级完整性约束条件， Cno是外码，被参照表是Course*/
        );
```

Mysql实际操作-Student

- Create Schema `S-T`;
- Use `S-T`;　　#` `对于 -
- CREATE TABLE Student

```
(Sno   CHAR(9) PRIMARY KEY, /* 列级完整性约束条件*/
 Sname  CHAR(20) UNIQUE,    /* Sname取唯一值*/
 Ssex    CHAR(2),
 Sage   SMALLINT,
 Sdept  CHAR(20)
);
```

基本表的定义、删除与修改

[例3.5] 建立“学生”表Student，学号是主码，姓名取值唯一

- 常用完整性约束
  - 主码约束： **PRIMARY KEY**
  - 唯一性约束：**UNIQUE**
  - 非空值约束：**NOT NULL**
  - 参照完整性约束

PRIMARY KEY与 UNIQUE的区别？

实际操作-SC

- Use `S-T`;
- CREATE TABLE SC

```
(Sno  CHAR(9),
Cno  CHAR(4),
Grade   SMALLINT,
Semester  CHAR(5),
PRIMARY KEY (Sno, Cno),
                /* 主码由两个属性构成，必须作为表级完整性进行定义*/
FOREIGN KEY (Sno) REFERENCES Student(Sno),
                /* 表级完整性约束条件，Sno是外码，被参照表是Student */
FOREIGN KEY (Cno) REFERENCES Course(Cno)
                /* 表级完整性约束条件， Cno是外码，被参照表是Course*/
);
```

Mysql实际操作

[例5、6、7] 在Mysql中的例子

```
CREATE TABLE Student
(Sno    CHAR(9) PRIMARY KEY,
    Sname   CHAR(20) UNIQUE,
    Ssex    CHAR(2),
    Sage    SMALLINT,
    Sdept   CHAR(20)
);
```

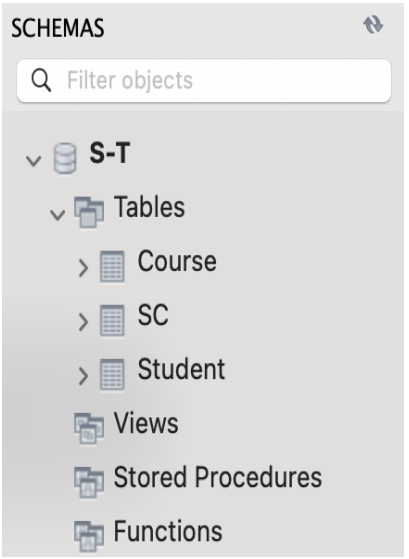
```
CREATE TABLE  Course
(Cno CHAR(4) PRIMARY KEY,
    Cname   CHAR(40),
    Cpno    CHAR(4),
    Ccredit  SMALLINT,
    FOREIGN KEY (Cpno) REFERENCES Course(Cno)
);
```

```
CREATE TABLE SC
(Sno  CHAR(9),
    Cno  CHAR(4),
    Grade    SMALLINT,
    Semester  CHAR(5),
    PRIMARY KEY (Sno, Cno),
    FOREIGN KEY (Sno) REFERENCES Student(Sno),
    FOREIGN KEY (Cno) REFERENCES Course(Cno)
);
```

执行 show tables; 命令，在workbench中可以看到

Result Grid Filter Rows: Search Export:

Tables_in_s-t
Course
SC
Student



或者直接在workbenck左侧的schemas中查看情况

数据定义DDL

执行CREATE TABLE语句后：

- 有关表的定义、约束条件定义等翻译成内部表、存储在系统的数据字典中
- 数据库为基本表分配了（预留了）存储空间

2、数据类型

- SQL中域的概念用数据类型来实现
- 定义表的属性时 需要指明其数据类型及长度
- 选用哪种数据类型
  - 取值范围
  - 要做哪些运算

2、数据类型

数据类型	含义
CHAR(*n*),CHARACTER(*n*)	长度为n的定长字符串
VARCHAR(*n*), CHARACTERVARYING(*n*)	最大长度为n的变长字符串
CLOB	字符串大对象
BLOB	二进制大对象
INT，INTEGER	整数（4字节），取值范围是[-2147483648,2147483647]
SMALLINT	短整数（2字节），取值范围是[-32768,32767]
BIGINT	大整数（8字节），取值范围是[-2^63,2^63-1]
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
DECIMAL(*p*，*d*)，DEC(*p*，*d*)	同NUMERIC类似，但数值精度不受*p*和*d*的限制
REAL	取决于机器精度的单精度浮点数
DOUBLE PRECISION	取决于机器精度的双精度浮点数
FLOAT(*n*)	可选精度的浮点数，精度至少为*n*位数字
BOOLEAN	逻辑布尔量
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
TIMESTAMP	时间戳类型
INTERVAL	时间间隔类型

三、模式与表

- 每一个基本表都属于某一个模式
- 一个模式包含多个基本表
- 定义基本表所属模式
  - 方法一：在表名中明显地给出模式名

    Create table ` S-T `.Student（.....）； /*模式名为 S-T*/

    Create table ` S-T `.Cource（.....）；

    Create table ` S-T `.SC（.....）；
  - 方法二：在创建模式语句中同时创建表
    - **例3.3， Create schema ` S-T ` create table Student**
  - 方法三：设置所属的模式
    - **例如Use `S-T`;**

实际操作

- create table `S-T`.CloneStudent
  (Sno CHAR(9) PRIMARY KEY, # 列级完整性约束条件
  Sname CHAR(20) UNIQUE, # Sname取唯一值
  Ssex CHAR(2),
  Sage SMALLINT,
  Sdept CHAR(20)
  );

```
create table `S-T`.CloneStudent
(Sno    CHAR(9) PRIMARY KEY,
    Sname   CHAR(20) UNIQUE,
    Ssex     CHAR(2),
    Sage    SMALLINT,
    Sdept   CHAR(20)
);
```

四、修改基本表

修改基本表：

ALTER TABLE <表名>

[ADD[COLUMN] <新列名> <数据类型> [完整性约束]]

[ADD <表级完整性约束>]

[DROP [COLUMN] <列名> [CASCADE| RESTRICT]]

[DROP CONSTRAINT<完整性约束名>[RESTRICT | CASCADE]]

[ALTER COLUMN <列名><数据类型>] ;

四、修改基本表

- <表名>是要修改的基本表
- ADD子句用于增加新列、新的列级完整性约束条件和新的表级完整性约束条件
- DROP COLUMN子句用于删除表中的列
  - 如果指定了CASCADE短语，则自动删除引用了该列的其他对象
  - 如果指定了RESTRICT短语，则如果该列被其他对象引用，关系数据库管理系统将拒绝删除该列
- DROP CONSTRAINT子句用于删除指定的完整性约束条件
- ALTER COLUMN子句用于修改原有的列定义，包括修改列名和数据类型

修改基本表（续）

[例3.8]向Student表增加“邮箱地址”列Semail，其数据类型为字符型。

```
ALTER TABLE Student ADD Semail VARCHAR(30);
```

- 不论基本表中原来是否已有数据，新增加的列一律为空值。
- 删除呢？

```
ALTER TABLE Student drop Semail;
```

[例3.9]将年龄的数据类型由字符型（假设原来的数据类型是字符型）改为整数。

```
ALTER TABLE Student ALTER COLUMN Sage INT;      #Mysql报错
```

[例3.10]增加课程名称必须取唯一值的约束条件。

```
ALTER TABLE Course ADD UNIQUE(Cname);
```

修改基本表（续）

- MySQL （两种方法）
  - ALTER TABLE 表名称 **change** 字段名称 字段名称 字段类型 [是否允许非空];
  - ALTER TABLE 表名称 **modify** 字段名称 字段类型 [是否允许非空];

```
alter table Student change Sage StudentAage INT;     #修改列名

alter table Student modify Sage INT not null;        #mysql
```

- 例如，修改表expert_info中的字段birth,允许其为空

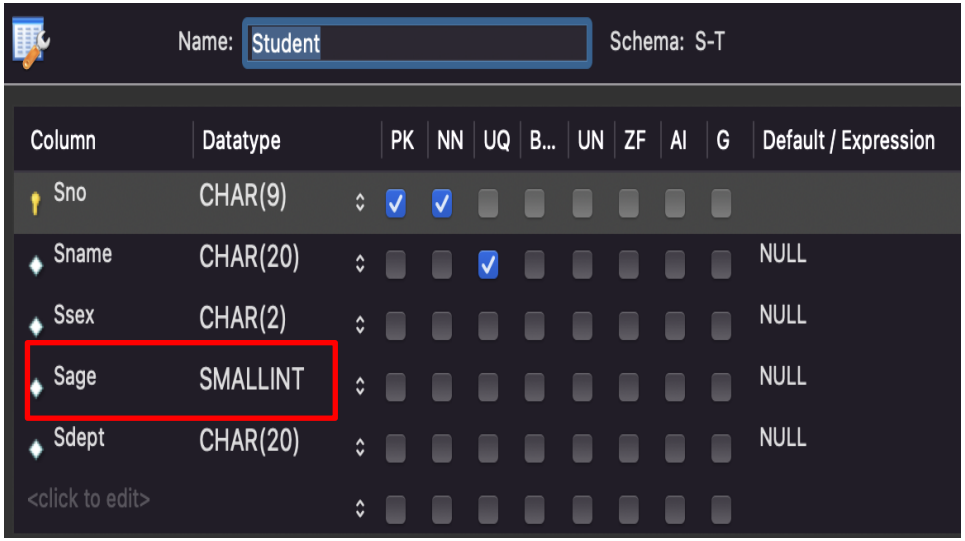
ALTER TABLE expert_info CHANGE birth birth varchar(20) **null**;

修改基本表（续）

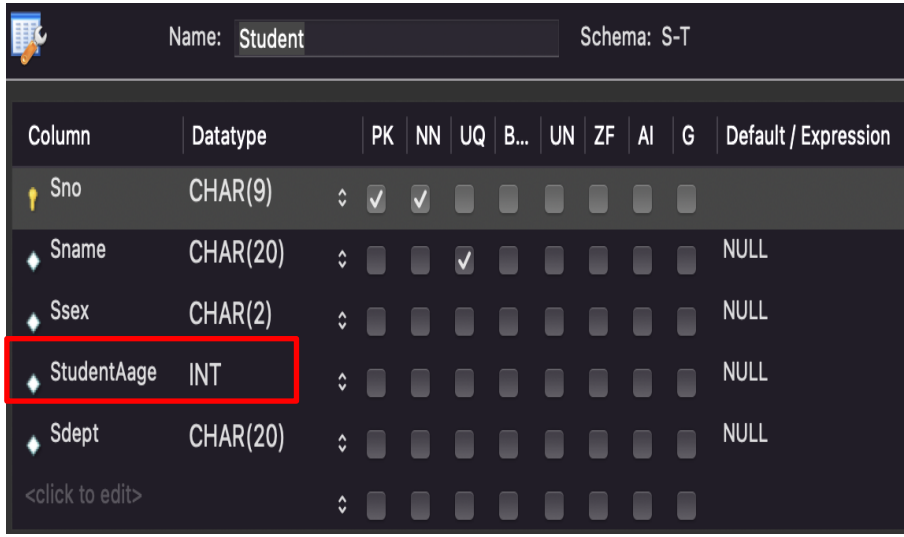
- MySQL （两种方法）

```
alter table Student change Sage StudentAage INT;
alter table Student modify Sage INT not null;
```

修改前



修改后



五、删除基本表

DROP TABLE <表名>［RESTRICT| CASCADE］；

- RESTRICT：删除表是有限制的。
  - 欲删除的基本表不能被其他表的约束所引用
  - 如果存在依赖该表的对象，则此表不能被删除
- CASCADE：删除该表没有限制。
  - 在删除基本表的同时，相关的依赖对象一起删除
- 默认：RESTRICT

删除基本表(续)

[例3.11] 删除Student表

DROP TABLE Student CASCADE;

- 基本表定义被删除，数据被删除
- 表上建立的索引、视图、触发器等一般也将被删除
- 非常危险
  - SC表也删除

删除基本表（续）

［例3.12］若表上建有视图，选择RESTRICT时表不能删除

```
CREATE VIEW IS_Student
AS
   SELECT Sno，Sname，Sage
   FROM  Student
   WHERE Sdept = 'IS';


DROP TABLE Student RESTRICT;
-----ERROR: cannot drop table Student because other
                objects depend on it
```

删除基本表（续）

[例3.12]如果选择CASCADE时可以删除表，视图也自动被删除

```
DROP TABLE Student CASCADE;
 --NOTICE: drop cascades to view IS_Student
SELECT * FROM IS_Student;
--ERROR: relation " IS_Student " does not exist
```

删除基本表（续）-不同数据库产品略有差别

DROP TABLE时，SQL2011 与 3个RDBMS的处理策略比较

序号	标准及主流数据库的处理方式 / 依赖基本表的对象	SQL2011		Kingbase ES		ORACLE 12c		MS SQL SERVER 2012
		R	C	R	C		C	
1.	索引	无规定		√	√	√	√	√
2.	视图	×	√	×	√	√ 保留	√ 保留	√ 保留
3.	DEFAULT，PRIMARY KEY，CHECK（只含该表的列）NOT NULL等约束	√	√	√	√	√	√	√
4.	Foreign Key	×	√	×	√	×	√	×
5.	TRIGGER	×	√	×	√	√	√	√
6.	函数或存储过程	×	√	√ 保留	√ 保留	√ 保留	√ 保留	√ 保留

R表示RESTRICT , C表示CASCADE

'×'表示不能删除基本表，'√'表示能删除基本表，‘保留’表示删除基本表后，还保留依赖对象

课后思考：MySQL呢?

删除基本表（续）

[例3.11] 删除Student表（Mysql ）

```
Drop Table Student;
```

报错，因为存在约束被SC表引用！

```
11  11:25:37  Drop Table Student  Error Code: 3730. Cannot drop table 'student' referen...  0.0013 sec
```

```
Drop Table SC;
Drop Table Student;
```

先删除有外键约束的SC表，再删除Student

```
13  11:33:15  D  0 row(s) affected  0.011 sec
14  11:33:17  D  0 row(s) affected  0.0072 sec
```

注：另一种方式是禁用外键约束，再删除数据

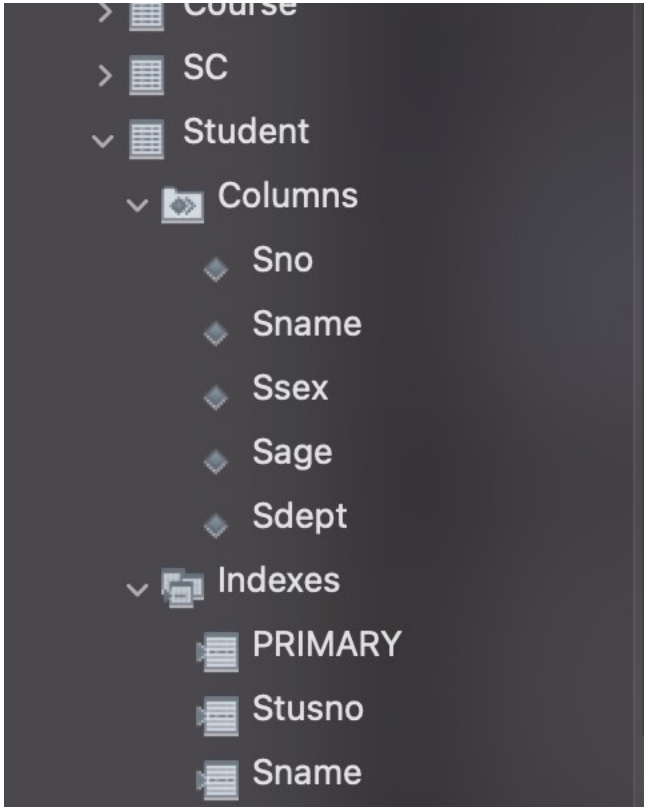
SET　FOREIGN_KEY_CHECKS=0;

3.3 数据定义

- 3.3.1 模式的定义与删除
- 3.3.2 基本表的定义、删除与修改
- 3.3.3 索引的建立与删除
- 3.3.4 数据字典

3.3.3 索引的建立与删除

- 建立索引（Index）的目的：加快查询速度
- 谁可以建立索引
  - DBA 或 表的属主（即建立表的人）
  - DBMS一般会自动建立以下列上的索引
    - PRIMARY KEY
    - UNIQUE
- 谁维护索引
  - DBMS自动完成
- 使用索引
  - DBMS自动选择是否使用索引以及使用哪些索引



3.3.3 索引的建立与删除

- 建立索引（Index）的目的：加快查询速度
- 关系数据库管理系统中常见索引：
  - 顺序文件上的索引
  - B+树索引（数据结构课程）
  - 散列（hash）索引（数据结构课程）
  - 位图索引
- 特点：
  - B+树索引具有动态平衡的优点
  - HASH索引具有查找速度快的特点

索 引

- RDBMS中索引一般采用B+树、HASH索引来实现
  - B+树索引具有动态平衡的优点
  - HASH索引具有查找速度快的特点
- 采用B+树，还是HASH索引 则由具体的RDBMS来决定
- 索引是关系数据库的内部实现技术，属于内模式的范畴
- CREATE INDEX语句定义索引时，可以定义索引是唯一索引、非唯一索引或聚簇索引
- 聚簇索引：元组按照索引键值的顺序存储

一、建立索引

- 语句格式

CREATE [UNIQUE] [CLUSTER] INDEX <索引名>

ON <表名>(<列名>[<次序>][,<列名>[<次序>]]...);

- **<表名>：** 要建索引的基本表的名字
- 索引：可以建立在该表的一列或多列上，各列名之间用逗号分隔
- **<次序>：** 指定索引值的排列次序，升序：**ASC**，降序：**DESC**。缺省值：**ASC**
- **UNIQUE：** 此索引的每一个索引值只对应唯一的数据记录
- **CLUSTER：** 表示要建立的索引是聚簇索引
  - **Section 7.5.2**

一、建立索引

- 唯一值索引
  - 对于已含重复值的属性列不能建UNIQUE索引
  - 对某个列建立UNIQUE索引后，插入新记录时DBMS会自动检查新记录在该列上是否取了重复值。这相当于增加了一个UNIQUE约束

建立索引（续）

- 聚簇索引
  - 是顺序结构与数据存储物理结构一致的一种索引
  - 建立聚簇索引后，基表中数据也需要按指定的聚簇属性值的升序或降序存放。即聚簇索引的索引项顺序与表中记录的物理顺序一致

- 非聚簇索引
  - 记录的物理顺序与逻辑顺序没有必然的联系，与数据的存储物理结构没有关系
  - 一个表对应的非聚簇索引可以有多条，根据不同列的约束可以建立不同要求的非聚簇索引

细节在第7.5.2节

建立索引（续）

[例]在Student表的Sage列上建立一个聚簇索引

```
CREATE CLUSTER INDEX Stuage
 ON  Student(Sage);
```

“年龄”属性建立聚簇索引，是合适的

[例] CREATE CLUSTER INDEX Stusname

```
 ON  Student(Sname);
```

在Student表的Sname（姓名）列上建立一个聚簇索引，而且Student表中的记录将按照Sname值的升序存放

建立索引（续）

- 一个基本表上最多只能建立一个聚簇索引
- 聚簇索引的用途
  - 对于某些类型的查询，可以提高查询效率
  - 在最经常查询的列上建立聚簇索引以提高查询效率
- 聚簇索引的适用范围
  - 很少对基表进行增删操作
  - 很少对其中的变长列进行修改操作
- 聚簇索引的不适用情形
  - 表记录太少
  - 经常插入、删除、修改的表
  - 数据分布平均的表字段

建立索引（续）

[例3.13]为学生-课程数据库中的Student，Course，SC三个表建立索引。

Student表按学号升序建唯一索引

Course表按课程号升序建唯一索引

SC表按学号升序和课程号降序建唯一索引

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

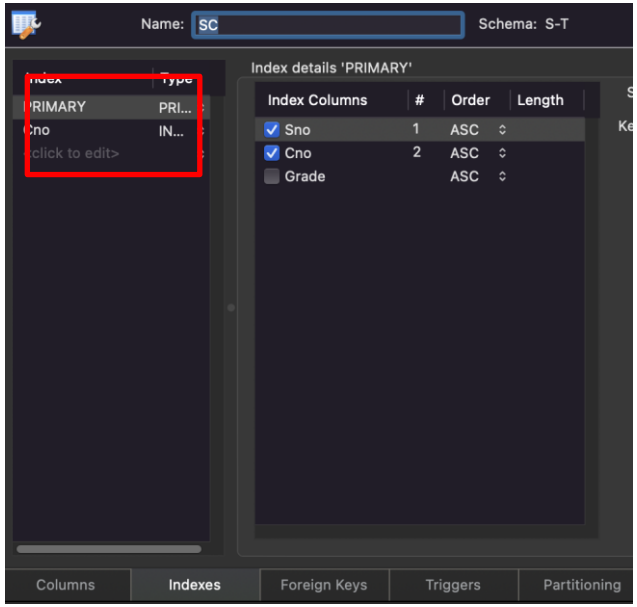
建立索引（续）

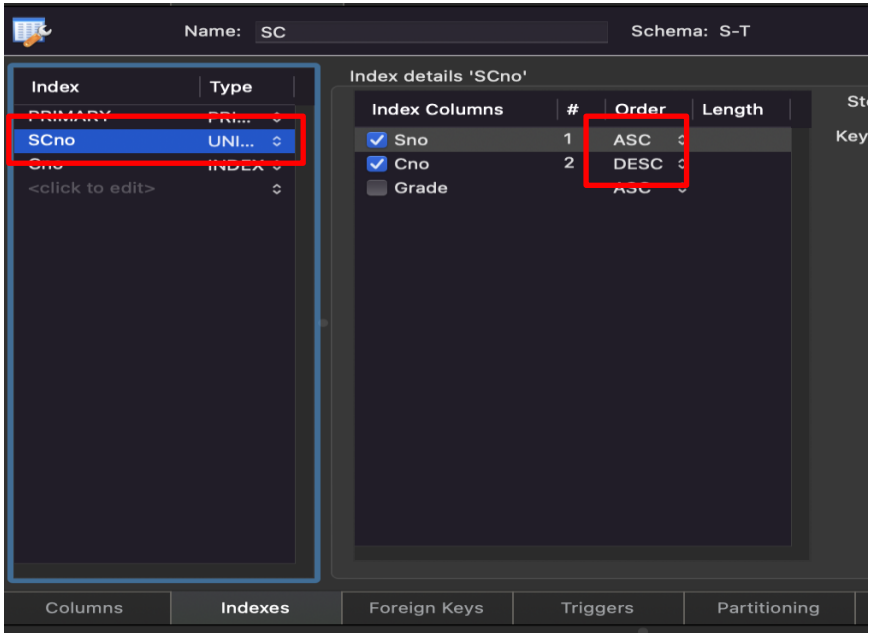
[例3.13]为学生-课程数据库中的Student，Course，SC三个表建立索引(Mysql 例子)

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

注意，mysql在当创建或设置主键的时候，mysql会自动添加一个与主键对应的唯一索引！

以SC表为例，建立索引前和建立索引后的情况





二、更新索引

- **ALTER INDEX <旧索引名> RENAME TO <新索引名>**

  - **[例3.14] 将SC表的SCno索引名改为SCSno**

    ALTER INDEX SCno RENAME TO SCSno;

三、删除索引

- DROP INDEX <索引名>;

  删除索引时，系统会从数据字典中删去有关该索引的

  描述。

  [例3.15] 删除Student表的Stusname索引

  DROP INDEX Stusname;

  DROP INDEX Stusname on student;

3.3 数据定义

- 3.3.1 模式的定义与删除
- 3.3.2 基本表的定义、删除与修改
- 3.3.3 索引的建立与删除
- 3.3.4 数据字典

3.3.4 数据字典

- 数据字典是关系数据库管理系统内部的一组系统表，它记录了数据库中所有定义信息：
  - 关系模式定义
  - 视图定义
  - 索引定义
  - 完整性约束定义
  - 各类用户对数据库的操作权限
  - 统计信息等
- RDBMS在执行SQL的数据定义语句时，实际上就是在更新数据字典表中的相应信息。