数据库系统原理

教程：数据库系统概论（第5版）

结合：CMU 15-445/645 INTRO TO DATABASE SYSTEMS

华中科技大学 计算机学院

左琼

第三章 关系数据库标准语言SQL

Principles of Database Systems

第三章 关系数据库标准语言SQL

3.1 SQL概述
3.2 学生-课程数据库
3.3 数据定义
3.4 数据查询
3.5 数据更新
3.6 视图
3.7 小结

3.1 SQL概述

- 三种具有相同表达能力的抽象查询语言：
  - 关系代数　　ISBL
  - 元组关系演算语言　　ALPHA，QUEL
  - 域关系演算语言　　QBE
- SQL(Structured Query Language)则是介于关系代数和关系演算之间的标准查询语言。
  - 由IBM提出，是应用得最广泛的关系数据库标准语言。
  - 与之相比，Ingres的QUEL具有“理论优势”；
  - SQL是一种比关系代数表达式更加自然化的查询需求描述语言。

共同特点：

- 语言具有完备的表达能力；
- 是非过程化的集合操作语言；
- 功能强，能嵌入高级语言中使用。

3.1 概述-SQL发展史

- 1974年，Boyce和Chamberlin提出SEQUEL(STUCTURED ENGLISH QUERY LANGUAGE)；IBM公司对其进行了修改，并用于其SYSTEM R关系数据库系统中。1981年改名SQL；
- 1982年，美国国家标准局ANSI开始制定SQL标准；
- 1986年，ANSI的数据库委员会：SQL语言第一个标准SQL86；
- 1987年，ISO通过了SQL86标准；
- 1989年，ISO对SQL86进行了补充，推出了SQL89标准；
- 1992年，ISO又推出了SQL92标准，也称为SQL2；

DBMS：初级、中级和完全级，后又在初、中级间增加过渡级。

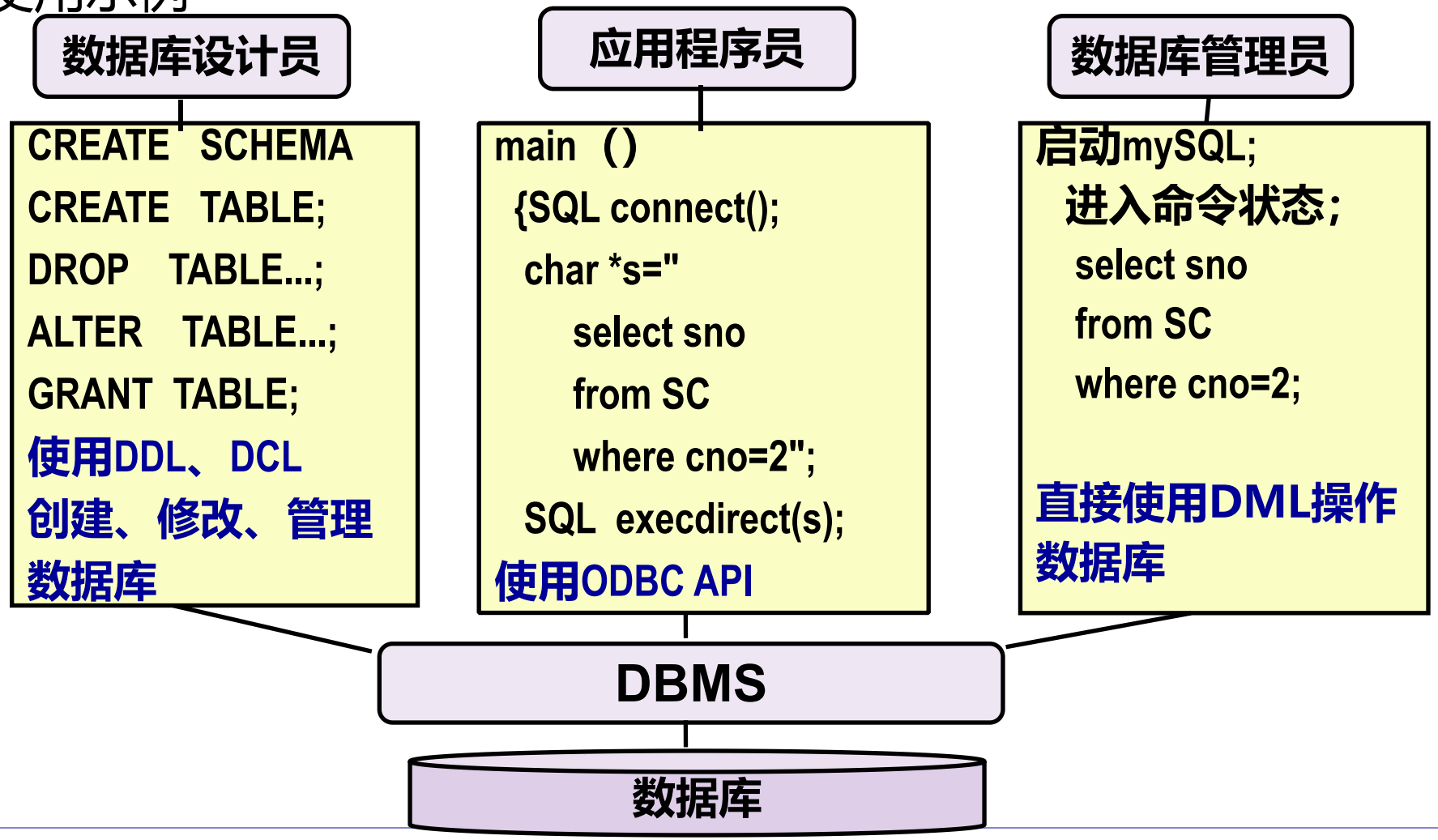
- 1999年，SQL99（即SQL3），分为核心SQL和增强SQL。
- 2003年，SQL2003；SQL2008；2010年，SQL2011。
- SQL标准文本的修改和完善还在继续进行。

其他：

Oracle的PL/SQL语言，SQL Server的T-SQL语言

3.1.2 SQL的特点

- SQL使用示例



3.1.2 SQL的特点

1. 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 / 修改 / 删除关系模式，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（单一的结构——关系，增删改查都只有一种操作符）。

3.1.2 SQL的特点

2. 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“What to do”，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。

3. 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（Set-at-a-time）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

3.1.2 SQL的特点

4.以同一种语法结构提供多种使用方式

- 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: 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。

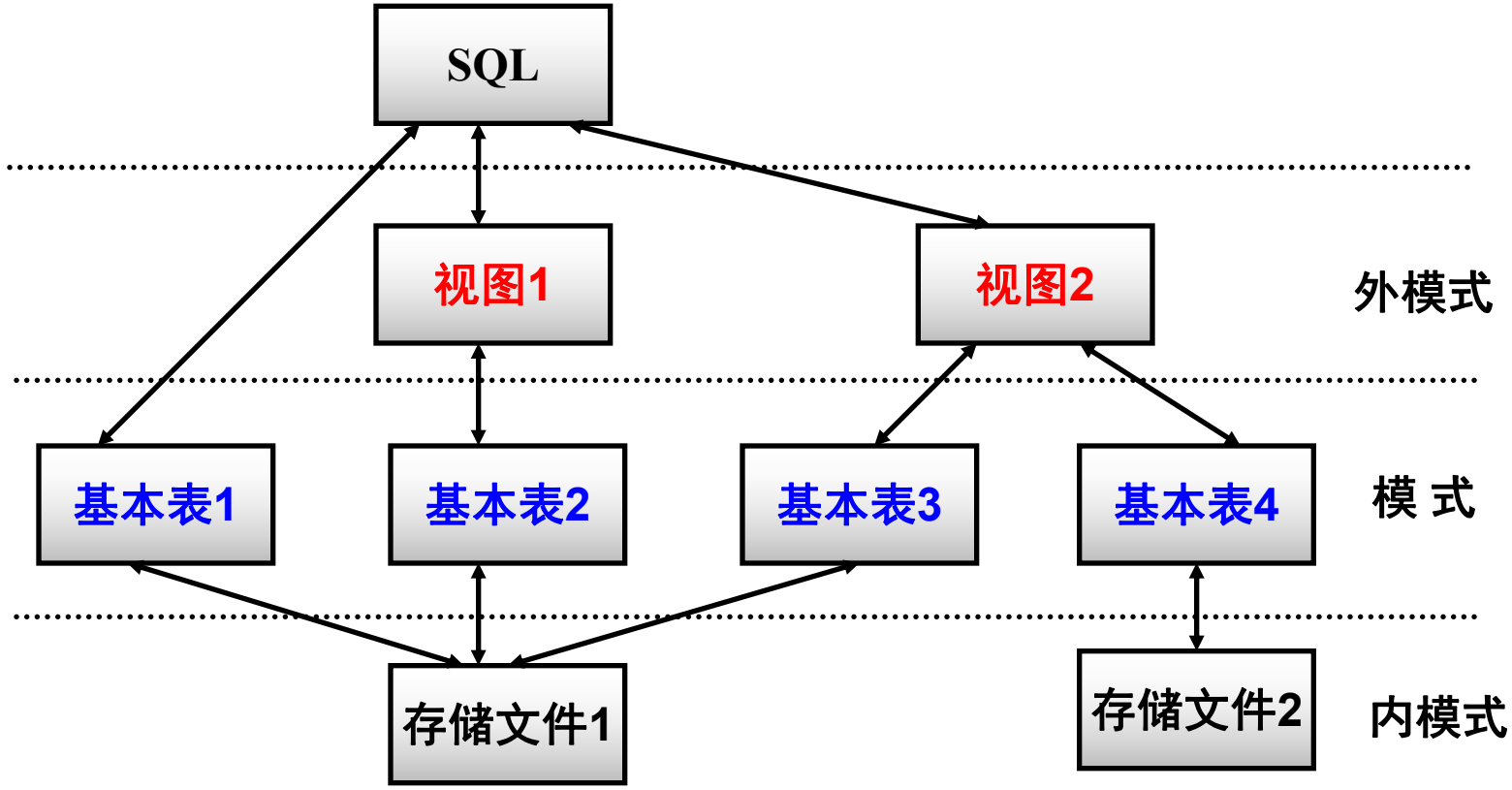
5.语言简洁，易学易用

SQL设计巧妙，核心功能只需9个动词。

- 数据查询（DQL）： select
- 数据定义（DDL）： create, drop, alter
- 数据操纵（DML）： delete, update, insert
- 数据控制（DCL）： grant, revoke

3.1.3 SQL的基本概念

- SQL支持关系数据库三级模式结构：

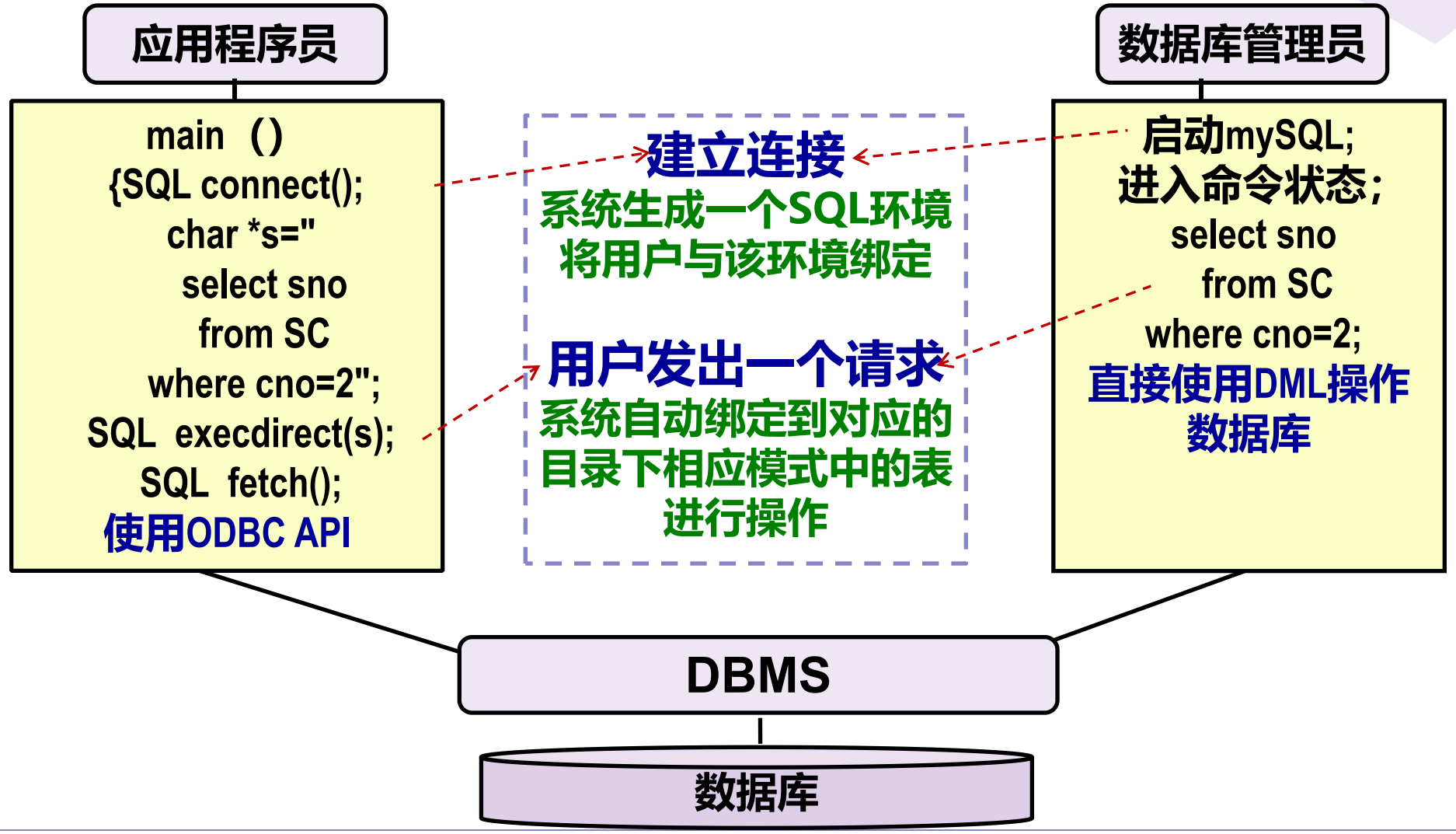


3.1.3 SQL的基本概念

- 基本表（base table）
  - 独立存在的表，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，索引也放在存储文件中。
- 存储文件（stored file）
  - 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（view）
  - 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。

在用户眼中，视图和基本表都是关系，而存储文件对用户是透明的。

SQL使用示例


应用程序员
main ()
{SQL connect();
char *s="
select sno
from SC
where cno=2";
SQL execdirect(s);
SQL fetch();
使用ODBC API
建立连接
系统生成一个SQL环境
将用户与该环境绑定
用户发出一个请求
系统自动绑定到对应的
目录下相应模式中的表
进行操作
数据库管理员
启动mySQL;
进入命令状态;
select sno
from SC
where cno=2;
直接使用DML操作
数据库
DBMS
数据库

SQL使用说明 ·用户/程序如何使用命名空间中的对象?

1）用户（程序）使用数据库时首先必须连接（建立一个会话）

对每个用户，系统自动产生一个目录；

连接时，系统将该用户与其自动目录对应；

用户连接时需提供用户名和密码认证。

2）对应每个连接，系统自动生成一个SQL环境

一个SQL环境是动态的，只与当前用户/程序相关；

系统根据SQL环境，控制当前用户/程序的行为；

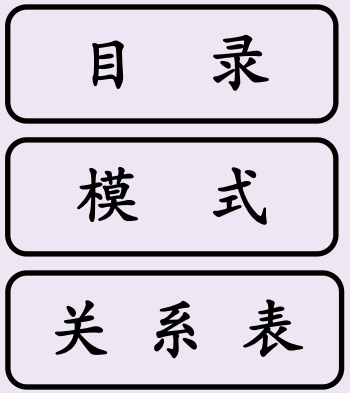
SQL环境中包括当前用户的目录、模式、授权等信息。

3）用户连接后，默认已处于自己的目录，表的定位无需指出目录名和模式名

在一个目录下可以建立多个模式；

在一个模式下可以建立多个关系表。

A schema is a collection of database objects (used by a user).



3.2 学生-课程 数据库

- 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)

学 号 Sno	姓 名 Sname	性 别 Ssex	年 龄 Sage	所在系 Sdept
200215121	李勇	男	20	CS

课程号 Cno	课程名 Cname	先行课 Cpno	学分 Ccredit
1	数据库	5	4

学 号 Sno	课程号 Cno	成绩 Grade
200215121	1	92

3.3 数据定义

- 建立数据库时首先要对数据库中数据的结构及特征进行描述，该过程称为数据定义。SQL语言的数据定义功能是通过SQL模式语句来实现的。
- 在SQL语言中，关于数据的描述信息以SQL模式（Schema）的形式组织。每个模式包含0到多个对象，这些对象被称为模式对象，如基表、视图、索引等。
- 用于模式及模式对象的创建、修改和销毁的语句被统称为SQL模式语句。SQL模式语句可分为三类：CREATE语句、DROP语句和ALTER语句。

3.3 数据定义

- SQL的数据定义语言DDL功能: 模式定义、表定义、视图和索引的定义。

表 SQL 的数据定义语句

操作对象	操作方式		
	创 建	删 除	修 改
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	

DATABASE

PROCEDURE

TRIGGER

FUNCTION

- 层次化数据库对象命名机制:

1个DBMS实例：多个DB;

=> 1个DB中：多个模式;

=> 1个模式下：多个表、视图、索引.....

创建、删除和修改数据库*

1. 创建数据库

一般格式：

```
CREATE DATABASE <数据库名>
 DATAFILE    '<文件路径>'
 SIZE <文件大小>;
```

功能：创建一个新数据库及存储该数据库的文件。

2. 删除数据库

一般格式：

```
DROP DATABASE <数据库名>;
```

功能：删除指定数据库及其包含的所有文件。

3. 修改数据库属性

一般格式：

```
ALTER DATABASE <数据库名>
  ADD DATAFILE    '<文件路径>'
  SIZE <文件大小>;
  |
  MODIFY DATAFILE    '<文件路径>'
  INCREASE <文件增量>;
```

功能：

增大数据库原有文件大小；或在数据库中加入新的数据库文件。

SQL Server创建数据库*

```
CREATE DATABASE database_name
[ ON
    [ < filespec > [ ,...n ] ]
    [ , < filegroup > [ ,...n ] ]
]
 [ LOG ON { < filespec > [ ,...n ] } ]
< filespec > ::=
[ PRIMARY ]
( [ NAME = logical_file_name , ]
   FILENAME = 'os_file_name'
[ , SIZE = size ]
[ , MAXSIZE = { max_size | UNLIMITED } ]
[ , FILEGROWTH = growth_increment ] ) [ ,...n ]
< filegroup > ::=
FILEGROUP filegroup_name < filespec > [ ,...n ]
```

数据库的名称，必须唯一 。

指定数据库的数据文件和文件组 。其中<filespec>用来定义主文件组的数据文件<filegroup>用来定义用户文件组及其文件。

指定数据库的事务日志文件属性 。

指定主文件。一个数据库只能有一个主文件

数据库的逻辑文件名。

数据库的物理文件名及其存储路径 。

数据文件的初始大小。

数据文件大小的最大值

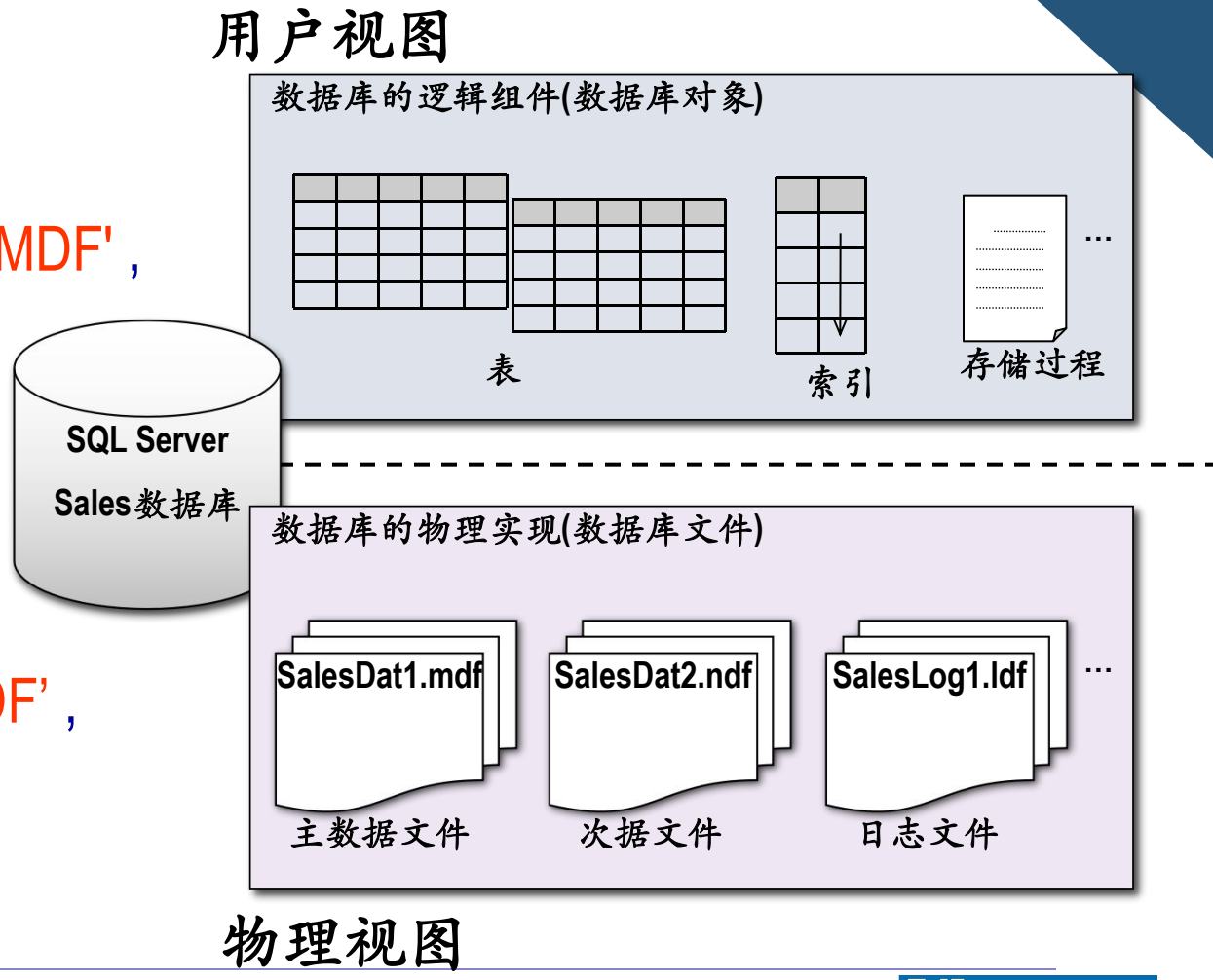
数据文件的增量。0值表示不增长。

指定文件组属性。

SQL Server创建数据库*

例：创建数据库example：

```
CREATE DATABASE [example]  ON
(NAME = 'example_Data',
 FILENAME = 'C:\MSSQL\data\example_Data.MDF' ,
SIZE = 2,
FILEGROWTH = 10%)
LOG ON
(NAME = 'example_Log',
FILENAME ='C:\MSSQL\data\example_Log.LDF' ,
SIZE = 1,
FILEGROWTH = 10%)
```



Mysql创建数据库*

Mysql

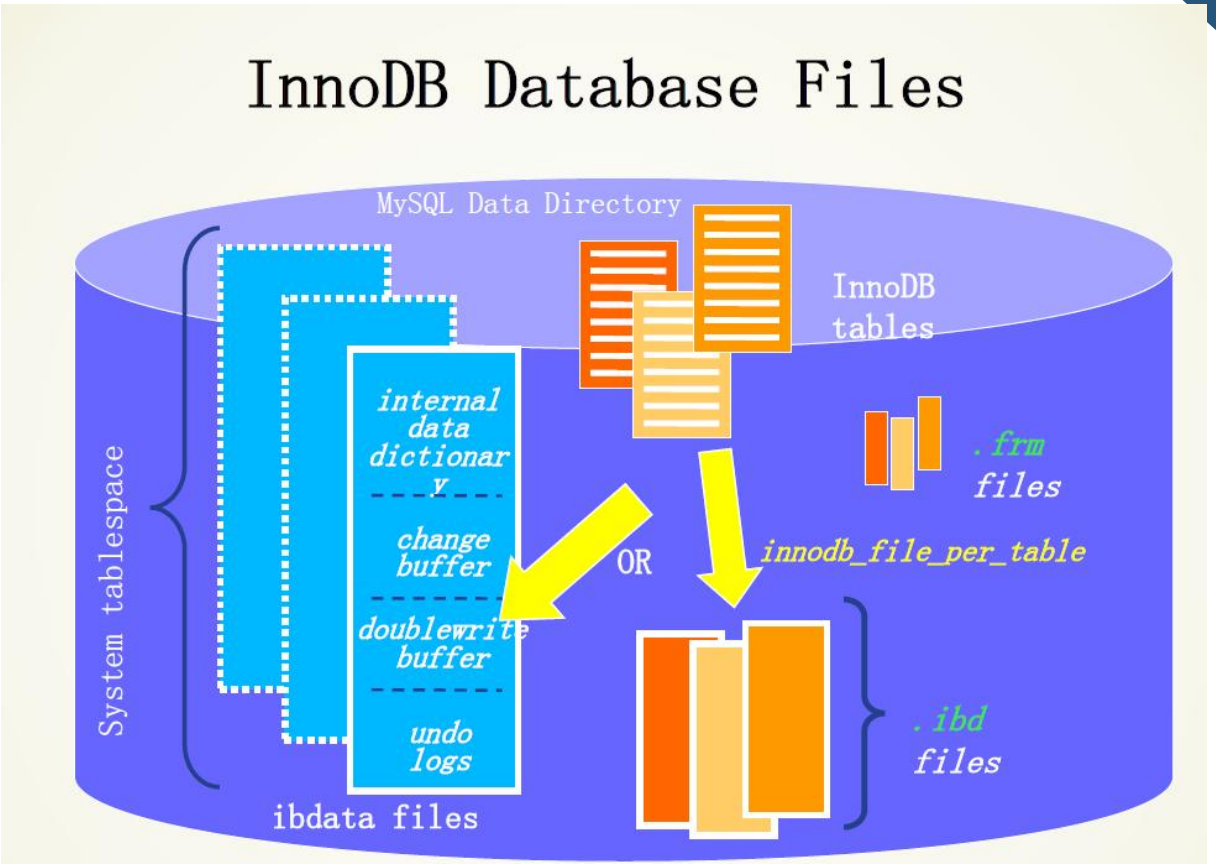
CREATE DATABASE [IF NOT EXISTS] <数据库名>

[[DEFAULT] CHARACTER SET <字符集名>]

[[DEFAULT] COLLATE <校对规则名>];

如：

Create database if not exists MyDB;



3.3.1 模式的定义与删除

- Schema（ISO/IEC 9075-1）：a persistent named collection of descriptors（因DBMS而异)
- SQL server

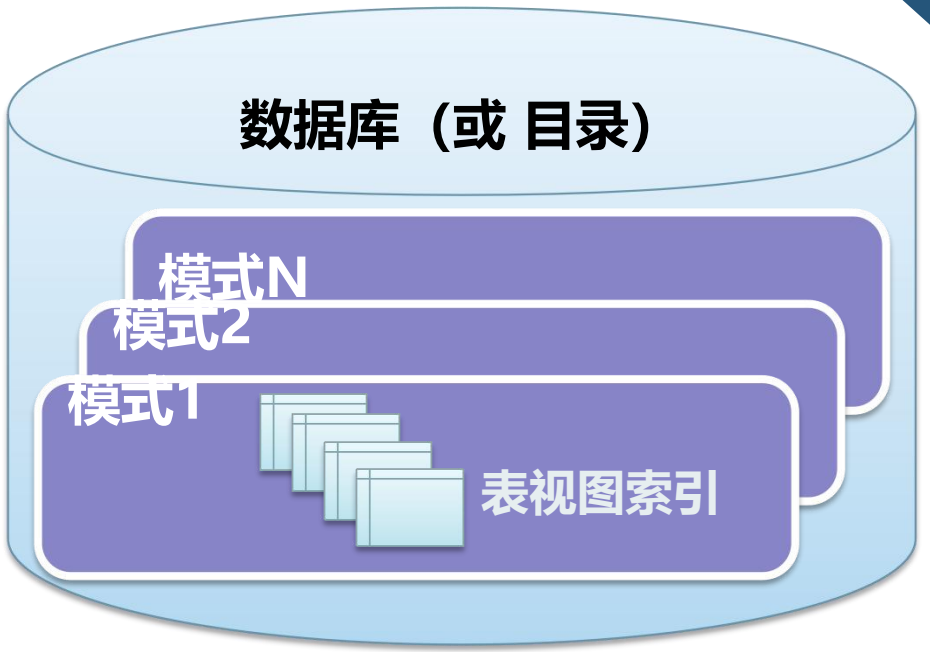
  Database schemas act as namespaces or containers for objects, such as tables, views, procedures, and functions.

- Oracle

  A schema is a collection of logical structures of data. A schema is owned by a database user and has the same name as that user. Each user owns a single schema.

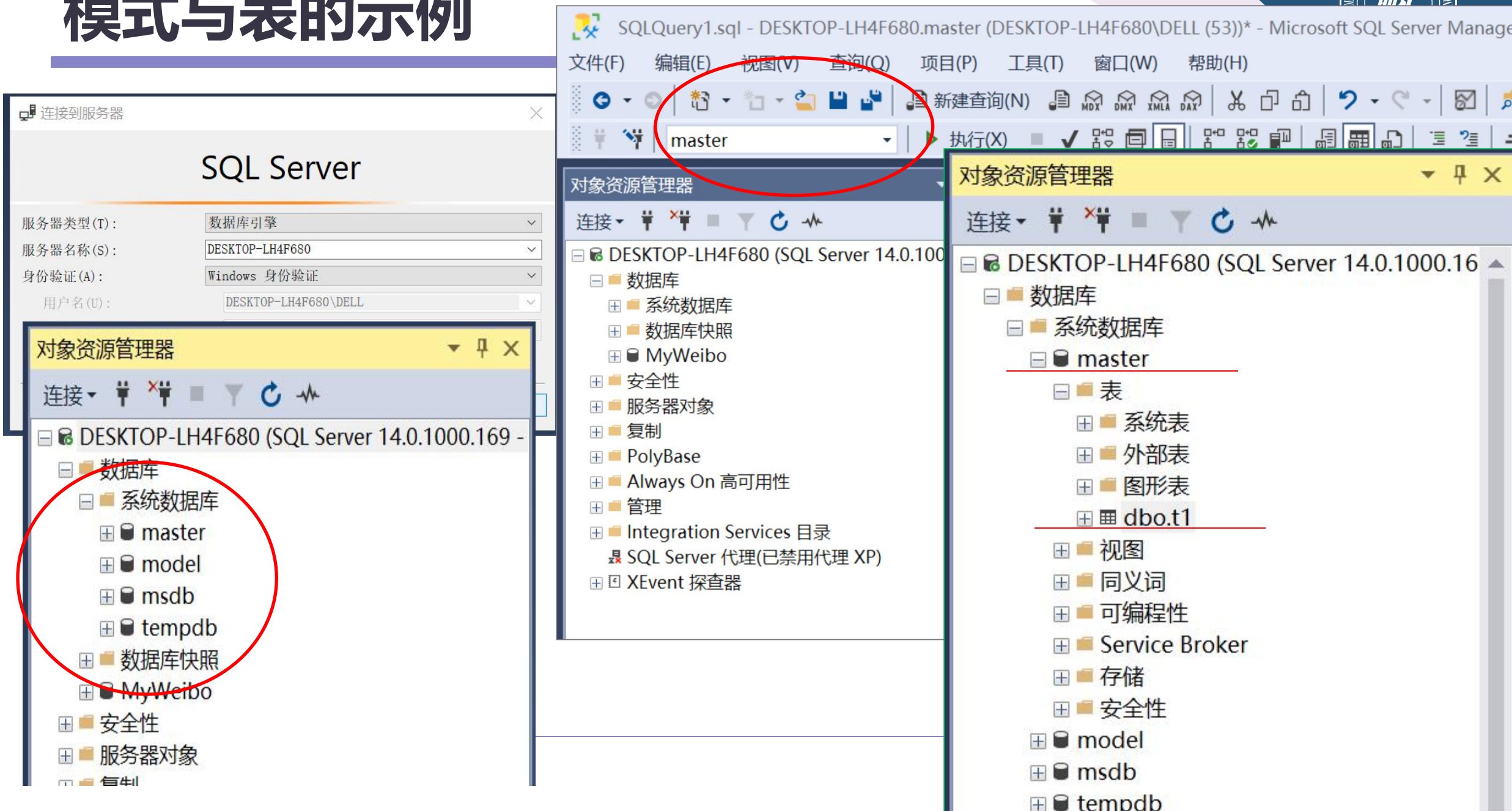
- MySQL

  Physically, a schema is synonymous with a database.



[<数据库名>.][<模式名>.]<表名>

模式与表的示例


连接到服务器
SQL Server
服务器类型(T):
数据库引擎
服务器名称(S):
DESKTOP-LH4F680
身份验证(A):
Windows 身份验证
用户名(U):
DESKTOP-LH4F680\DELL
对象资源管理器
连接
DESKTOP-LH4F680 (SQL Server 14.0.1000.169 -
数据库
系统数据库
master
model
msdb
tempdb
数据库快照
MyWeibo
安全性
服务器对象
复制
SQLQuery1.sql - DESKTOP-LH4F680.master (DESKTOP-LH4F680\DELL (53))* - Microsoft SQL Server Manage
文件(F)
编辑(E)
视图(V)
查询(Q)
项目(P)
工具(T)
窗口(W)
帮助(H)
新建查询(N)
master
执行(X)
对象资源管理器
连接
DESKTOP-LH4F680 (SQL Server 14.0.100
数据库
系统数据库
数据库快照
MyWeibo
安全性
服务器对象
复制
PolyBase
Always On 高可用性
管理
Integration Services 目录
SQL Server 代理(已禁用代理 XP)
XEvent 探查器
对象资源管理器
连接
DESKTOP-LH4F680 (SQL Server 14.0.1000.16
数据库
系统数据库
master
表
系统表
外部表
图形表
dbo.t1
视图
同义词
可编程性
Service Broker
存储
安全性
model
msdb
tempdb

3.3.1 模式的定义与删除

1. 定义模式

- 模式定义语句如下：

```
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>];
```

- 语义：为某用户创建一个模式。定义模式实际上定义了一个命名空间，其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。
  - SQL Server中：一个DB下可以有多个架构，不同架构下允许同名数据对象（db.schema.object）。每个DB有一个缺省的schema叫dbo。
  - Oracle：创建user同时创建同名schema；
  - Mysql：一个DB就是一个schema
- 在CREATE SCHEMA中可以接受CREATE TABLE，CREATE VIEW和GRANT子句。

1. 定义模式

创建模式的用户必须拥有DBA权限，或DBA授予的创建模式权限！

[例1] 为用户WANG定义了一个模式S-T

CREATE SCHEMA “S-T” AUTHORIZATION WANG;

[例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>

[例3] 为用户ZHANG创建一个模式TEST，并在其中定义一个表TAB1 。

CREATE SCHEMA TEST AUTHORIZATION ZHANG

CREATE TABLE TAB1

(COL1 SMALLINT, COL2 INT, COL3 CHAR(20))；

2. 删除模式

DROP SCHEMA <模式名> <CASCADE|RESTRICT>;

- 删除模式时其中已有表如何办?

在删除语句中提供了CASCADE、RESTRICT选择项，说明如何删除:

- 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。

- [例4] DROP SCHEMA ZHANG CASCADE;

  删除例3定义的模式ZHANG，同时该模式中定义的表TAB1也被删除

3.3.2 基本表的定义、删除与修改

- 定义基本表：

  SQL DDL不仅允许定义一组关系，也要说明每个关系的信息：

  - 每个关系的模式
  - 每个属性的值域
  - 完整性约束
  - 每个关系的安全性和权限
  - 每个关系需要的索引集合
  - 每个关系在磁盘上的物理存储结构

- 备注：

  本章主要阐述如何定义模式、域、基本完整性及索引如何在SQL中定义，其他部分在后面章节

3.3.2 基本表的定义、删除与修改

1. 基本表的定义(创建)

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

1. 创建基本表

口 语法：

```
CREATE TABLE  表名(
    列名  数据类型  [DEFAULT  缺省值]  [NOT NULL]
    [, 列名 数据类型  [DEFAULT 缺省值]  [NOT NULL] … ]
    [, PRIMARY KEY(列名 [, 列名] …)]
    [, FOREIGN KEY (列名 [, 列名] …)
      REFERENCES  表名(列名 [, 列名] …)]
    [,  CHECK (条件表达式)] );
```

注：句法中[ ]表示该成分是可选项。

最常用的完整性约束：

- 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：foreign key
- 自定义的完整性约束：check(逻辑表达式)

1. 创建基本表

[例] 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)
 );
```

Primary key 与 unique 的区别？

前者非空，后者允许一个空

学 号 Sno	姓 名 Sname	性 别 Ssex	年 龄 Sage	所在系 Sdept
200215121	李勇	男	20	CS

1. 创建基本表

[例] 建立一个“课程”表Course

```
CREATE TABLE  Course
( Cno      CHAR(4) PRIMARY KEY,
  Cname  CHAR(40) NOT NULL,
  Cpno     CHAR(4) ,
  Ccredit  SMALLINT,
   FOREIGN KEY (Cpno) REFERENCES Course(Cno)
);
```

课程号 Cno	课程名 Cname	先行课 Cpno	学分 Ccredit
1	数据库	5	4

1. 创建基本表

[例] 建立一个“学生选课”表SC

学 号 Sno	课程号 Cno	成绩 Grade
200215121	1	92

```
CREATE TABLE SC
(Sno CHAR(9),
Cno CHAR(4),
Grade SMALLINT CHECK (grade>=0 and grade <=100),
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*/
);
```

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
- SQL 语言常用数据类型:
  - 精确数值类型;
  - 近似数值类型;
  - 字符串类型;
  - 日期时间类型;
  - 二进制对象等。

数据类型	SQL92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）-2^31 ～2^31 -1
SMALLINT	短整数 -2^15 ～2^15 -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
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
INTEVAL	两个date或time类型数据之间的差

3. 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（常用方法2种）：
  - 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

Mysql

```
Create database SCC;
use SCC;
Create table student(…);
Create table Course(…);
Create table SC(…);
```

4. 修改数据表

一般格式：

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

□ 语义：

对名为 <表名>的表做ADD、DROP或ALTER COLUMN的操作：

- ADD可以增加一个新的列；
- DROP只能删除表上的完整性约束；
- ALTER只能更改列上的数据类型。

4. 修改基本表

- [例] 向Student表增加“S_entrance(入学时间)”列，其数据类型为日期型；删除average列。

```
ALTER TABLE Student ADD S_entrance DATE;
```

说明：不论基本表中原来是否已有数据，新增加的列一律为空值，且除非表为空表或为列指定默认值，不允许新增约束为not null的列。

- [例] 将“入学时间”列的缺省值设为当前日期。

```
ALTER  TABLE   STUDENT  ALTER  S_entrance SET
                                DEFAULT CURRENT_DATE;
```

说明：修改列的定义可能破坏原有数据。

4. 修改基本表

- [例] 将表Course增加约束：先行课cpno必须引用本表的Cno 。

```
ALTER TABLE Course ADD CONSTRAINT fk_cpno FOREIGN KEY(cpno) REFERENCES Course(cno);
```

- [例] 删除列average；

```
ALTER TABLE Student DROP average;
```

- [例] 增加课程名称必须取唯一值的约束条件；删除Course表上的外键约束。

```
ALTER TABLE Student ADD UNIQUE(Cname);
ALTER TABLE Student DROP CONSTRAINT(fk_cpno);
```

5. 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。

- [例] 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。

5. 删除基本表

- 例：删除前面建的三个表:Student,Course, SC。

```
Drop table SC;
Drop table Student;
Drop table Course;
```

问：三条删除命令可不可以变动次序？比如将第一条挪到后面？

5. 删除基本表

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

[例] 若表上建有视图，选择RESTRICT时表不能删除。
如果选择CASCADE时可以删除表，视图也自动被删除

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

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

5. 删除基本表

- DROP TABLE时，SQL99 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	SQL99		Kingbase ES		ORACLE 9i		MS SQL SERVER 2000
		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

3.3.3 索引的建立与删除

- 建立索引的目的：加快查询速度
- 谁 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
- 谁 使用索引：

  SQL用户并不直接使用索引。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

- 常用的索引技术

  B+树索引　索引属性值组成B+树，具有动态平衡的优点

  HASH索引　索引属性值分桶，具有查找速度快的特点

  顺序索引　索引属性值排序，可二分查找

  位图索引　索引属性值用位向量描述

- 采用什么索引，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

1. 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>] ]...);
```

其中，

<次序>—指ASC(升序)/DESC(降序)，缺省为升序。

- 功能：在<表名>的表上，对其中的指定列，建立一个名为<索引名>的索引文件。
- 作用：提高查询速度。如从O(n)到O(log2n)。
- DBA建立、系统自动实现，与编程无关。
- 好处：提高速度
- 坏处：过多或不当的索引会耗费空间，且降低插入、删除、更新的效率。

3.3.3 索引的建立与删除

说明：

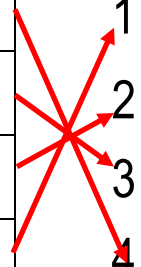
- UNIQUE(单一索引)：
  - 唯一索引，表示索引项值对应元组唯一，不允许存在索引值相同的两行
- CLUSTER(聚集索引)：索引项的顺序与表中记录的物理顺序一致。表中如果有多个记录在索引字段上相同，这些记录构成一簇，只有一个索引值。
- 在最经常查询的列上建立聚簇索引以提高查询效率；
- 经常更新的列不宜建立聚簇索引。
  - 优点：查询速度快。
  - 缺点：维护成本高，且一个表只能建一个聚簇索引。

3.3.3 索引的定义

□ 普通索引 vs 聚簇索引

按学号索引

sno	idx
17121	4
17122	3
17126	2
17128	1

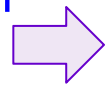


	sno	sname	ssex	sage	sdept
1	17128	李三姐	女	18	IS
2	17126	李一鸣	男	17	IS
3	17122	刘晨	女	19	CS
4	17121	李勇	男	20	CS

按姓名索引

sname	idx
李三姐	1
李一鸣	2
李勇	4
刘晨	3

按学号聚簇索引



sno	sname	ssex	sage	sdept
17121	李勇	男	20	CS
17122	刘晨	女	19	CS
17126	李一鸣	男	17	IS
17128	李三姐	女	18	IS

1. 建立索引

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

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

- [例] 对Student表按姓名same的字典序升序建立一个简单索引。

```
CREATE  INDEX Stuname ON Student(Sname);
    //没有修饰词cluster,unique，列名sname后省略了asc
```

2. 修改索引

- 一般格式：

```
ALTER  INDEX <旧索引名> RENAME TO <新索引名>;
```

- [例] 将SC表索引Scno改名为SCSno。

```
ALTER  INDEX Scno RENAME TO SCSno;
```

3. 删除索引

- 一般格式：

```
DROP INDEX <索引名>;
或
DROP INDEX <索引名> ON <表名>;
```

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] 删除Student表的Stusname索引

```
DROP INDEX Stusname;
DROP INDEX Stusname ON Student;
```

3.3.4 数据字典

- 数据字典： 关系DBMS内部的一组系统表，记录数据库中所有的定义信息。
- 包括：
  - 关系模式定义；
  - 视图定义；
  - 索引定义；
  - 完整性定义；
  - 操作权限；
  - 统计信息等。