数 据 库 原 理

Theory of Database

李静

信息科学与技术学院

复　习

关系运算

- 传统的关系运算：
  - ✓ 并（Union）
  - ✓ 交（Intersection）
  - ✓ 差（Difference）
  - ✓ 笛卡尔乘积（Cartesian Product）
- 专门的关系运算：
  - 选择（Select）
  - 投影（Project）
  - 连接（Join）
  - 除（Divide）

第3章 SQL语言基础及数据定义功能

- 3.1 基本概念
- 3.2 SQL的数据类型
- 3.3 数据定义功能
- 3.4 数据完整性

3.1 基本概念

- 3.1.1 SQL语言的发展
- 3.1.2 SQL语言的特点
- 3.1.3 SQL语言功能概述

3.1.1 SQL语言的发展

SQL（Structured Query Language），即结构化查询语言，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。

3.1.1 SQL语言的发展

- 1986年10月由美国ANSI 公布最早的SQL标准。
- 1989年4月，ISO提出了具备完整性特征的SQL，称为SQL-89 。
- 1992年11月，ISO又公布了新的SQL标准，称为SQL-92（以上均为关系形式）。
- 1999年颁布SQL-99，是SQL92的扩展，也称为SQL3。

SQL和SQL SERVER的区别

SQL（structured query language)结构化查询语言。

它是一种标准，不是一种软件

SQL SERVER是数据库管理系统的一种

它是一种软件，这种软件在遵循SQL这种标准，很多数据库管理软件及开发工具都支持SQL这种标准。

3.1.2 SQL语言的特点

1.综合统一

数据定义语言(DDL)，数据查询语言(DQL)，数据操纵语言(DML)，数据控制语言（DCL）于一体；

可独立完成数据库生命周期中的全部活动,随时修改；

数据操作符统一。

2.高度非过程化

面向对象的第四代语言；

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。

3.1.2 SQL语言的特点

3.面向集合的操作方式

操作对象、查找结果可以是元组的集合；

一次插入、删除、更新操作的对象是元组的集合。

4.以同一种语法结构提供多种使用方式

独立的语言，能够独立用于联机交互的使用方式；

嵌入式语言，能够嵌入到高级语言中使用。

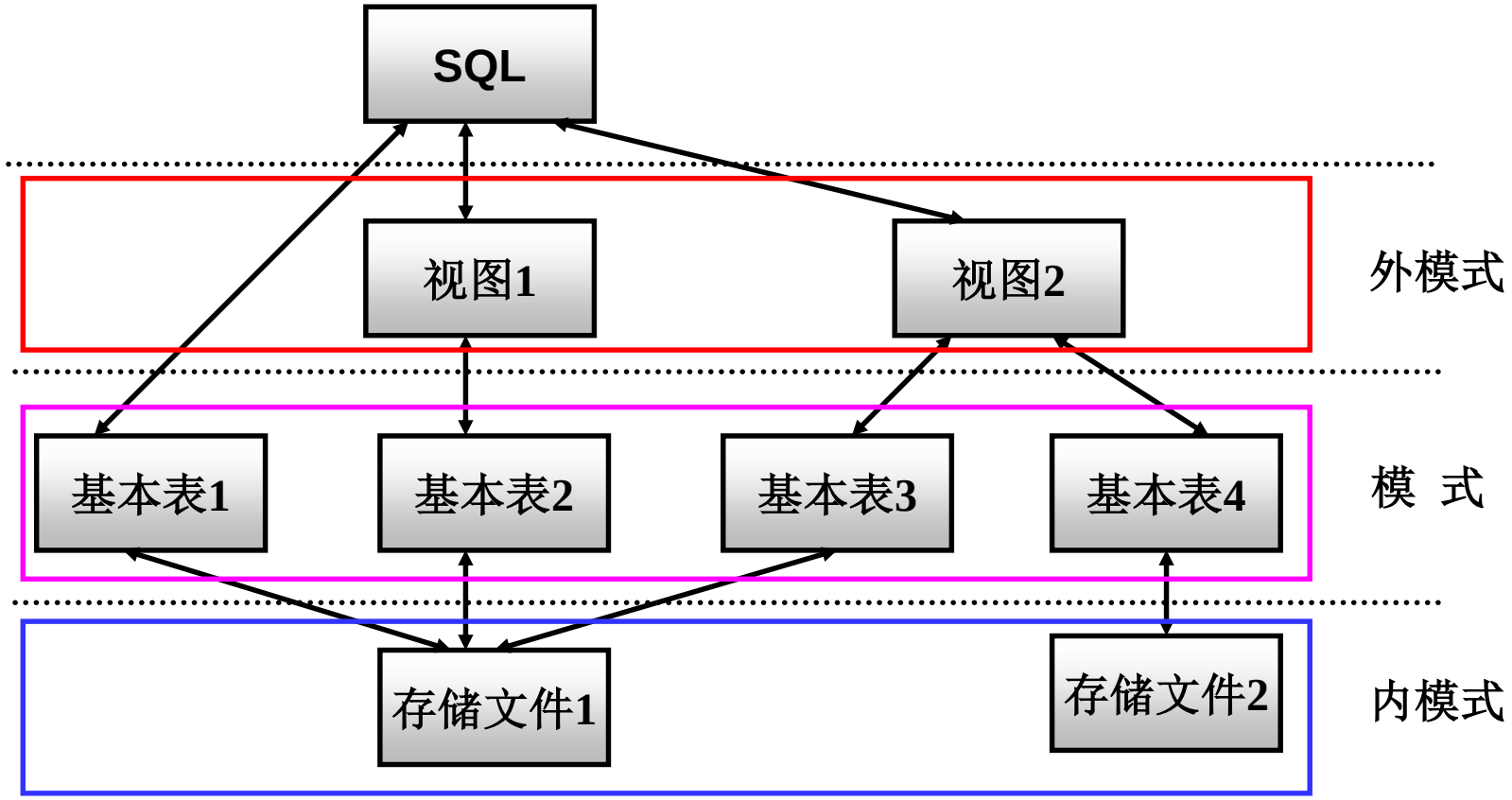
3.1.3 SQL语言功能概述

四部分：数据定义、控制、查询和操纵功能。

SQL功能	命令动词
数据查询	SELECT
数据定义	CREATE、DROP、ALTER
数据操纵	INSERT 、 UPDATE 、 DELETE
数据控制	GRANT、REVOKE

3.1.3 SQL语言功能概述-补充

SQL支持关系数据库三级模式结构



3.1.3 SQL语言功能概述-补充

★基本表
- 本身独立存在的表
- 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

★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
- 视图是一个虚表
- 用户可以在视图上再定义视图

3.1.3 SQL语言功能概述-补充

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

索引

- 为快速访问数据，而在包含数据的表中增加的一种组织。
- 分为聚簇索引和非聚簇索引

指索引项的顺序与表中记录的物理顺序一致的索引组织

3.2 SQL的数据类型

- 数值型
- 字符串型
- 日期时间型

数值型

- 准确型
  - 整数

    Bigint: 8字节　　Int：4字节

    Smallint：2字节　　Tinyint：1字节，0~255整数

    Bit：1位，存储1或0
  - 小数

    Numeric（p,q）或Decimal（p,q），

    其中：p为数字位长度，q：小数位长度。
- 近似型

  Float：8字节

  Real：4字节

字符串型

- 普通编码字符串类型
- 统一字符编码字符串类型
- 二进制字符串类型

普通编码字符串类型

- **Char（n）:**定长存储，**n<=8000**
- **Varchar（n）：**

不定长存储（按实际），长度不超过n，n<=8000

注：n 为字符个数

- **Text：** 存储大于8000字节的文本

统一字符编码字符串类型

- **nchar（n）**:定长存储，**n<=4000**
- **nvarchar（n）**：

  不定长存储，长度最大不超过n，**n<=4000**
- **ntext**： 存储大于**8000**字节的文本

特点：每个字符占两个字节

二进制字符串类型

- **Binary(n)**： 固定长度，n <= 8000。
- **Varbinary(n)**： 可变长度，n <＝8000 。

  注：n为二进制数据的字节数
- **image**： 大容量、可变长二进制字符数据，可用于存储文件。

日期时间型

- **Datetime**： 8字节，年月日时分秒毫秒

  （例： ‘2001/08/03 10:30:00.000’）

- **SmallDateTime**： 4字节，年月日时分

  （例： ‘2001/08/03 10:30:00’）

日期、时间的输入格式：参加教材

3.3 数据定义功能

- 3.3.1 基本表的定义与删除
- 3.3.2 修改表结构

3.3.1 基本表的定义与删除

❖1. 定义基本表

CREATE TABLE <表名>

(<列名> <数据类型> [列级完整性约束定义]

{, <列名> <数据类型>

[列级完整性约束定义] … }

[, 表级完整性约束定义]

)

在列级完整性约束定义处可以定义的约束

- **NOT NULL**：限制列取值非空。
- **DEFAULT**：给定列的默认值。
- **UNIQUE**：限制列取值不重。
- **CHECK**：限制列的取值范围。
- **PRIMARY KEY**：指定本列为主码。
- **FOREIGN KEY**：定义本列为引用其他表的外码。

使用形式为：

［FOREIGN KEY(<外码列名>)］

REFERENCES <外表名>(<外表列名>)

几点说明

- NOT NULL和DEFAULT只能是列级完整性约束；
- 其他约束均可在表级完整性约束处定义。
- 注意以下三点：
  - 第一，如果CHECK约束是定义多列之间的取值约束，则只能在表级完整性约束处定义；
  - 第二，如果表的主码由多个列组成，则也只能在表级完整性约束处定义，并将主码列用括号括起来，

    即：PRIMARY KEY（列1｛[，列2]…｝）；
  - 第三，如果在表级完整性约束处定义外码，

    则“FOREIGN KEY (<外码列名>)”部分不能省。

约束定义

- ① 列取值非空约束

  <列名> <类型> NOT NULL

例：sname char(10) NOT NULL

约束定义（续）

- ② 表主码约束
- 在定义列时定义主码（仅用于单列主码）

  列定义 **PRIMARY KEY**

  例： **SNO char(7) PRIMARY KEY**
- 在定义完列后定义主码（用于单/多列主码）

  PRIMARY KEY （<列名序列>）

  例：**PRIMARY KEY(SNO)**

  PRIMARY KEY(SNO，CNO)

约束定义（续）

③外码引用约束

- 指明本表外码列引用的表及表中的主码列。

[foreign key (<本表列名>)]

references <外表名>(<外表主码列名>)

例：

foreign key (sno) references 学生表(sno)

创建学生表

```
CREATE TABLE Student (
  Sno     char ( 7 )  PRIMARY KEY,
  Sname char ( 10 ) NOT NULL,
  Ssex    char (2),
  Sage    tinyint ,
  Sdept   char (20)
)
```

创建课程表

```
CREATE TABLE Course (
  Cno       char(10)  NOT NULL,
  Cname     char(20)  NOT NULL,
  Ccredit   tinyint ,
  Semester  tinyint,
  PRIMARY KEY(Cno)
)
```

创建SC表

```
CREATE TABLE SC
(
  Sno   char(7)  NOT NULL,
  Cno   char(10)  NOT NULL,
  Grade  tinyint,
  XKLB   char(4) ,
  PRIMARY KEY ( Sno, Cno ),
  FOREIGN KEY (Sno) REFERENCES  Student (Sno),
  FOREIGN KEY (Cno) REFERENCES  Course (Cno)
)
```

删除表

- 当确信不再需要某个表时，可以将其删除
- 删除表时会将与表有关的所有对象一起删掉，包括表中的数据。
- 删除表的语句格式为：

DROP TABLE <表名> { [, <表名>] … }

- 例：删除test表的语句为：

DROP TABLE test

3.3.2 修改表结构

- 在定义完表之后，如果需求有变化，比如添加列、删除列或修改列定义，可以使用**ALTER TABLE**语句实现。

- **ALTER TABLE**语句可以对表添加列、删除列、修改列的定义、定义主码、外码，也可以添加和删除约束。

修改表结构语法

ALTER TABLE <表名>

[ALTER COLUMN <列名> <新数据类型>]

| [ADD [COLUMN] <列名> <数据类型>

| [DROP COLUMN <列名>]

| [ADD PRIMARY KEY（列名 [，…n] ）]

| [ADD FOREIGN KEY（列名）

REFERNECES 表名（列名）]

示例

- 例1：为SC表添加“修课类别”列，此列的定义为：XKLB char(4)

```
ALTER TABLE SC
ADD XKLB char(4) NULL
```

示例

- 例2：　将新添加的XKLB的类型改为char(6)。

```
ALTER TABLE SC
    ALTER COLUMN XKLB char(6)
```

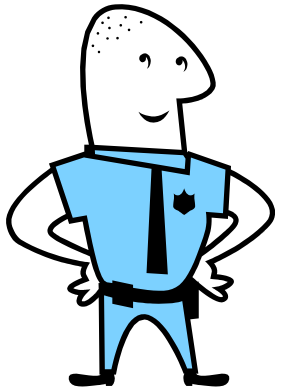
示例

- 例3：删除Course表的Period列。

```
ALTER TABLE Course
DROP COLUMN Period
```

3.4 数据完整性

- 3.4.1 完整性约束条件的作用对象
- 3.4.2 实现数据完整性



完整性约束条件的作用对象

- 完整性检查是围绕完整性约束条件进行的，因此，完整性约束条件是完整性控制机制的核心。
- 完整性约束条件的作用对象可以是表、元组和列。
  - 列级约束
  - 元组约束
  - 关系约束

列级约束

列级约束主要是对列的类型、取值范围、精度等的约束。

- 对数据类型的约束：

  包括数据类型、长度、精度等。

- 对数据格式的约束：

  如:规定学号的前两位表示学生的入学年份，第三位表示系的编号，第四位表示专业编号，第五位代表班的编号等等。

- 对取值范围的约束：

  如:学生的成绩取值范围为0～100。

- 对空值的约束。

元组约束

- 元组的约束是元组中各个字段之间的相互约束，如：
  - 开始日期小于结束日期，
  - 职工的最低工资不能低于规定的最低保障金。

关系约束

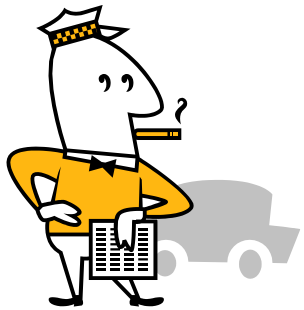
- 指若干元组之间、关系之间的联系的约束，如：
  - 学号的取值不能重复也不能取空值，
  - 学生修课表中的学号的取值受学生表中的学号取值的约束

实现数据完整性

- 声明完整性
  - 在表定义时声明
    - 使用约束、缺省值(DEFAULT)等
  - 由SQL Server自动加以保证
- 过程完整性
  - 在客户端或服务器端用编程语言或工具实现
    - 在Server端用触发器（trigger）来实现

实现约束

1. PRIMARY KEY 约束
2. UNIQUE 约束
3. FOREIGN KEY 约束
4. DEFAULT 约束
5. CHECK 约束



PRIMARY KEY 约束

- 保证实体完整性
- 每个表有且只有一个**PRIMARY KEY** 约束
- 格式：

ALTER TABLE 表名

ADD [CONSTAINT *约束名* **]**

PRIMARY KEY （列名 [, … n] ）

示例

- 例：对雇员表和工作表分别添加主码约束。

```
ALTER TABLE 雇员表
  ADD  CONSTRAINT  PK_EMP
  PRIMARY KEY (雇员编号)
ALTER TABLE 工作表
  ADD  CONSTRAINT  PK_JOB
  PRIMARY KEY (工作编号)
```

UNIQUE 约束

- 确保在非主键列中不输入重复值。
- 应用在客观具有唯一性质的列上，如身份证号、社会保险号等。
- 格式：

```
ALTER TABLE 表名
  ADD [ CONSTRAINT 约束名 ]
  UNIQUE（<列名> [, … n] ）
```

示例

- 例．为雇员表的“电话号码”列添加UNIQUE约束。

```
ALTER TABLE 雇员表
  ADD  CONSTRAINT  UK_SID
    UNIQUE (电话号码)
```

FOREIGN KEY约束

- 用于建立和加强两个表数据之间的连接的一列或多列
- 格式：

ALTER TABLE 表名

ADD [CONSTRAINT 约束名]

FOREIGN KEY (<列名>)

REFERENCES 引用表名 (<列名>)

示例

- 例：为雇员表的工作编号添加外码引用约束，此列引用工作表的工作编号列。

```
ALTER TABLE  雇员
  ADD CONSTRAINT FK_job_id
  FOREIGN KEY (工作编号)
  REFERENCES 工作表 (工作编号)
```

DEFAULT约束

当向表中插入数据时，如果没有为定义了DEFAULT的列提供值，则此列使用默认值。

- 一个Default只能约束一列。
- 格式：

ALTER TABLE 表名

ADD [CONSTRAINT 约束名]

DEFAULT 默认值 FOR 列名

示例

- 例：定义雇员表的工资的默认值为1000。

```
ALTER TABLE 雇员
  ADD CONSTRAINT  DF_SALARY
  DEFAULT  1000  FOR 工资
```

CHECK约束

- 通过限制输入到列中的值来强制域的完整性。
- 可定义同表多列之间的约束关系
- 格式：

ALTER TABLE 表名

ADD [CONSTRAINT 约束名]

CHECK （逻辑表达式）

示例1

- 例1：在雇员表中，添加限制雇员的工资必须大于等于500的约束。

```
ALTER TABLE 雇员
  ADD CONSTRAINT  CHK_Salary
CHECK ( 工资 >= 500 )
```

示例2

- 例2：添加限制工资表的最低工资小于等于最高工资的约束。

```
ALTER TABLE 工作
ADD CONSTRAINT  CHK_Job_Salary
    CHECK ( 最低工资 <= 最高工资 )
```

综合起来

```
CREATE TABLE 工作(
    工作编号  char(8) PRIMARY KEY,
    最低工资 int ,
    最高工资 int,
    CHECK ( 最低工资 <= 最高工资 ) )
CREATE TABLE 雇员 (
   雇员编号 char(7) PRIMARY KEY,
   雇员名    char(10),
   工作编号 char(8) REFERENCES 工作 ( 工作编号 ),
   工资   int DEFAULT 1000 CHECK ( 工资 >= 500 ),
   电话号码 char(8) not null UNIQUE )
```

数据库的创建

- 语法结构：

Create database 数据库名称

on primary （ name=数据文件逻辑名,

filename=数据文件物理名,

size=初始大小,

maxsize=最大,

filegrowth=增长率)

log on(日志文件的性质…..);

创建一个数据库

- 例如：

```
Create database  mydb
  on primary  (  name=mydb1,
                 filename='c:\db1',
                 size=10mb,
                 maxsize=100mb,
                 filegrowth=10%
               )
  log on(…..);
```

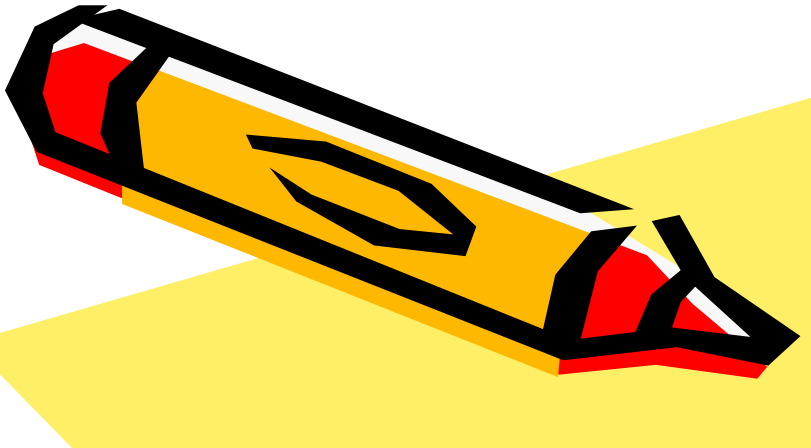
删除数据库

- 语法格式：

  Drop database 数据库名称；

- 例如：

  Drop database mydb;



作业

📖 P44 10题、11题、12题、13题