# 数据库技术与应用

北京邮电大学计算机学院 肖达

xiaoda99@gmail.com

# 第三章 关系数据库标准语言SQL


SQL概述

数据查询

**数据定义**

数据更新

视图

# 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 3.2　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 | |

# 定义模式（续）

[例1]定义一个学生-课程模式S-T

CREATE SCHEMA S-T AUTHORIZATION WANG;

为用户WANG定义了一个模式S-T

[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>

# 定义模式（续）

- 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。

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（续）

[例3]

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

为用户ZHANG创建了一个模式TEST，并在其中定义了一个表TAB1。

# 二、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。

[例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

# 数据定义

- 模式的定义与删除
- 基本表的定义、删除与修改
- 索引的建立与删除

# 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> ] ）；

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5] 建立“学生”表Student，学号是主码，姓名取值唯一。

主码

```
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)
      )；
```

# 课程表Course

［例6］建立一个“课程”表Course

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

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7] 建立一个“学生选课”表SC

```
CREATE TABLE  SC
        (Sno  CHAR(9)，
        Cno  CHAR(4)，
        Grade    SMALLINT，
        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*/
        );
```

# 二、数据类型

| 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 |

# 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
  - 方法三：设置所属的模式，在创建表时表名中不必给出模式名。

# 四、修改基本表

ALTER TABLE <表名>

[ ADD <新列名> <数据类型> [ 完整性约束 ] ]

[ DROP <完整性约束名> ]

[ ALTER COLUMN<列名> <数据类型> ];

# 修改基本表（续）

[例8]向Student表增加“入学时间”列，其数据类型为日期型。

ALTER TABLE Student ADD S_entrance DATE；

- 不论基本表中原来是否已有数据，新增加的列一律为空值。

[例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；

[例10]增加课程名称必须取唯一值的约束条件。

ALTER TABLE Course ADD UNIQUE(Cname);

# 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

[例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

# 删除基本表（续）


［例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
```

# 数据定义

- 模式的定义与删除
- 基本表的定义、删除与修改
- 索引的建立与删除

# 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

# 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

# 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>] ]...)；

  [例13] CREATE CLUSTER INDEX Stusname

  ON Student(Sname)；

  - 在Student表的Sname（姓名）列上建立一个聚簇索引

- 在最经常查询的列上建立聚簇索引以提高查询效率
- 一个基本表上最多只能建立一个聚簇索引
- 经常更新的列不宜建立聚簇索引

# 建立索引（续）

[例14]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

# 第三章 关系数据库标准语言SQL


SQL概述

数据查询

数据定义

**数据更新**

视图

# 数 据 更 新

插入数据

修改数据

删除数据

# 一、插入元组

- 语句格式

  INSERT

  INTO <表名> [(<属性列1>[，<属性列2 >…)]

  VALUES (<常量1> [，<常量2>] … )

- 功能
  - 将新元组插入指定表中

# 插入元组（续）

［例1］ 将一个新学生元组（学号：200215128；姓名：陈冬；性别：男；所在系：IS；年龄：18岁）插入到Student表中。

```
INSERT
INTO  Student (Sno，Sname，Ssex，Sdept，Sage)
VALUES ('200215128'，'陈冬'，'男'，'IS'，18)；
```

［例2］ 将学生张成民的信息插入到Student表中。

```
INSERT
INTO  Student
VALUES (‘200215126’，  ‘张成民’，  ‘男’，18，
'CS');
```

# 插入元组（续）

［例3］ 插入一条选课记录( '200215128'，'1 ')。

```
INSERT
INTO SC(Sno，Cno)
VALUES (‘ 200215128 ’，  ‘ 1 ’)；
```

RDBMS将在新插入记录的Grade列上自动地赋空值。

或者：

```
INSERT
INTO SC
VALUES (' 200215128 '，' 1 '，NULL);
```

# 二、插入子查询结果

- 语句格式

  INSERT

  INTO <表名> [(<属性列1> [，<属性列2>… )]

  子查询;

- 功能

  将子查询结果插入指定表中

# 插入子查询结果（续）

[例4] 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；
```

# 数 据 更 新

插入数据

修改数据

删除数据

# 修改数据

- 语句格式

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

- 功能
  - 修改指定表中满足WHERE子句条件的元组的指定列值

# 修改数据（续）

[例5] 将学生200215121的年龄改为22岁

```
UPDATE  Student
SET Sage=22
WHERE  Sno=' 200215121 ';
```

[例6] 将所有学生的年龄增加1岁

```
UPDATE Student
SET Sage= Sage+1;
```

[例7] 将计算机科学系全体学生的成绩置零。

```
UPDATE SC
SET  Grade=0
WHERE  'CS'=
          (SELETE Sdept
           FROM  Student
           WHERE  Student.Sno = SC.Sno);
```

# 数 据 更 新

插入数据

修改数据

删除数据

# 删除数据

- 语句格式

  DELETE

  FROM　　<表名>

  [WHERE <条件>];

- 功能
  - 删除指定表中满足WHERE子句条件的元组
- WHERE子句
  - 指定要删除的元组
  - 缺省表示要删除表中的全部元组，表的定义仍在字典中

# 删除数据（续）

[例8] 删除学号为200215128的学生记录。

```
DELETE
FROM Student
WHERE Sno= ‘200215128 ';
```

[例9] 删除所有的学生选课记录。

```
DELETE
FROM SC;
```

[例10] 删除计算机科学系所有学生的选课记录。

```
DELETE
FROM SC
WHERE  'CS'=
            (SELETE Sdept
             FROM Student
             WHERE Student.Sno=SC.Sno);
```

# 修改数据（续）

RDBMS在执行插入、修改和删除语句时会检查操作是否破坏表上已定义的完整性规则

- 实体完整性
- 主码不允许修改
- 用户定义的完整性
  - NOT NULL约束
  - UNIQUE约束
  - 值域约束

# 第三章 关系数据库标准语言SQL


SQL概述

数据查询

数据定义

数据更新

视图

# 视 图

视图的特点

- 虚表，是从一个或几个基本表（或视图）导出的表
- 只存放视图的定义，不存放视图对应的数据
- 基表中的数据发生变化，从视图中查询出的数据也随之改变

基于视图的操作

- 查询
- 删除
- 受限更新
- 定义基于该视图的新视图

# 一、建立视图

- 语句格式

  CREATE VIEW

  <视图名> [(<列名> [，<列名>]…)]

  AS <子查询>

  [WITH CHECK OPTION];

- 组成视图的属性列名：全部省略或全部指定
- 子查询不允许含有ORDER BY子句和DISTINCT短语
- RDBMS执行CREATE VIEW语句时只是把视图定义存入数据字典，并不执行其中的SELECT语句。
- 在对视图查询时，按视图的定义从基本表中将数据查出。

# 建立视图（续）

[例1] 建立信息系学生的视图。

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

[例2]建立信息系学生的视图，并要求进行修改和插入操作时仍需保证该视图只有信息系的学生 。

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

# 建立视图（续）

- 带WITH CHECK OPTION选项时对IS_Student视图的更新操作：
  - 修改操作：自动加上Sdept= 'IS'的条件
  - 删除操作：自动加上Sdept= 'IS'的条件
  - 插入操作：自动检查Sdept属性值是否为'IS'
    - 如果不是，则拒绝该插入操作
    - 如果没有提供Sdept属性值，则自动定义Sdept为'IS'

# 建立视图（续）

- 基于多个基表的视图

[例3] 建立信息系选修了1号课程的学生视图。

```
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'；
```

# 建立视图（续）

- 不指定属性列

[例7]将Student表中所有女生记录定义为一个视图

```
CREATE VIEW F_Student(F_Sno，name，sex，age，dept)
AS
SELECT *
FROM  Student
WHERE Ssex='女'；
```

缺点：

修改基表Student的结构后，Student表与F_Student视图的映象关系被破坏，导致该视图不能正确工作。

# 查询视图

- 用户角度：查询视图与查询基本表相同
- RDBMS实现视图查询的方法
  - 视图消解法（View Resolution）
    - 进行有效性检查
    - 转换成等价的对基本表的查询
    - 执行修正后的查询

# 查询视图（续）

[例9] 在信息系学生的视图中找出年龄小于20岁的学生。

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

IS_Student视图的定义

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

视图消解转换后的查询语句为：

```
SELECT  Sno，Sage
FROM  Student
WHERE  Sdept= 'IS'  AND  Sage<20；
```

# 更新视图

[例12] 将信息系学生视图IS_Student中学号200215122的学生姓名改为“刘辰”。

```
UPDATE  IS_Student
SET  Sname= '刘辰'
WHERE  Sno= ' 200215122 ';
```

转换后的语句：

```
UPDATE  Student
SET Sname= '刘辰'
WHERE Sno= ' 200215122 ' AND Sdept= 'IS';
```

# 更新视图（续）

[例13] 向信息系学生视图IS_S中插入一个新的学生记录：
200215129，赵新，20岁

```
INSERT
INTO IS_Student
VALUES (‘200215129’，‘赵新’，20);
```

转换为对基本表的更新：

```
INSERT
INTO  Student(Sno，Sname，Sage，Sdept)
VALUES (‘200215129 '，'赵新'，20，'IS' );
```

# 更新视图（续）

- 更新视图的限制：一些视图是不可更新的，因为对这些视图的更新不能唯一地有意义地转换成对相应基本表的更新

例：学生平均成绩的视图

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

视图S_G为不可更新视图

```
UPDATE  S_G
SET     Gavg=90
WHERE
Sno='200215121'；
```

这个对视图的更新无法转换成对基本表SC的更新

# 更新视图（续）

- 允许对行列子集视图进行更新
- 对其他类型视图的更新不同系统有不同限制
  - 因为有些视图的更新不能唯一地有意义地转换成对相应基本表的更新。例如DB2规定（在SQL Server中也有类似的规定）：
    - 若视图是由两个以上基本表导出的，则此视图不允许更新。
    - 若视图的字段来自字段表达式或常数，则不允许对此视图执行INSERT和UPDATE操作，但允许执行DELETE操作。
    - 若视图的字段来自聚集函数，则此视图不允许更新。
    - 若视图定义中含有GROUP BY子句，则此视图不允许更新。
    - 若视图定义中含有DISTINCT短语，则此视图不允许更新。
    - 若视图定义中有嵌套查询，并且内层查询的FROM子句中涉及的表也是导出该视图的基本表，则此视图不允许更新。
    - 一个不允许更新的视图上定义的视图也不允许更新。

# 视图的作用

1. 视图能够简化用户的操作

   - 视图机制使用户可以将注意力集中在他所关心的数据上。如果这些数据不是直接来自基本表，则可以通过定义视图，使用户眼中的数据库结构简单、清晰，并且可以简化用户的数据查询操作。

2. 视图使用户能以多种角度看待同一数据

   - 视图机制能使不同的用户以不同的方式看待同一数据，当许多不同种类的用户使用同一个数据库时，这种灵活性是非常重要的。

# 视图的作用（续）

3. 视图对重构数据库提供了一定程度的逻辑独立性

4. 视图能够对机密数据提供安全保护

有了视图机制，就可以在设计数据库应用系统时，对不同的用户定义不同的视图，使机密数据不出现在不应看到这些数据的用户视图上，这样就由视图的机制自动提供了对机密数据的安全保护功能。

# 作业

**Product**

| maker | model | type |
|---|---|---|
| A | 1001 | pc |
| A | 1002 | pc |
| A | 2004 | laptop |
| B | 1003 | pc |
| B | 2005 | laptop |
| C | 1004 | pc |

**PC**

| model | speed | ram | hd | price |
|---|---|---|---|---|
| 1001 | 2.66 | 1024 | 250 | 2114 |
| 1002 | 2.10 | 512 | 250 | 995 |
| 1003 | 1.42 | 512 | 80 | 478 |
| 1004 | 2.80 | 1024 | 250 | 649 |

**Laptop**

| model | speed | ram | hd | screen | price |
|---|---|---|---|---|---|
| 2004 | 2.00 | 512 | 60 | 13.3 | 1150 |
| 2005 | 2.16 | 1024 | 120 | 17.0 | 2500 |

用SQL完成以下操作：

1、用差集法查询选修了全部课程的学生姓名。
2、将所有内存容量为1024的笔记本价格下调200元。
3、创建制造商B生产的所有产品的型号和价格信息的视图。

# 实验2 交互式SQL

- 实验目的
  - 熟悉通过SQL对数据库进行操作
- 实验工具
  - SQL Server 2008提供的交互查询工具
- 实验数据库
  - 自行设计（要求至少包含三个实体、且有实体间一对多或多对多联系）

# 实验内容

- 数据定义
  - 模式的创建(1)和删除(1)
  - 基本表的创建(3)、修改(1)和删除(1)
  - 索引的创建(1)和删除(1)
- 数据操作
  - 各类更新操作（插入数据(n)、修改数据(1)、删除数据(1)）
  - 各类查询操作（单表查询(1)、连接查询(2)、嵌套查询(2)、集合查询(1)）
- 视图操作
  - 视图的创建(1)、删除 (1)、查询(1)、更新(1)

- 注意
  - (x)：x个SQL测试用例
  - 实验内容的顺序
  - SQL Server 2008不能完全支持SQL99

# 实验报告

- 内容
  - 实验环境
  - 实验内容与完成情况
  - 出现的问题及解决方法
- 要求
  - 实验过程中要进行抓图，把某些关键步骤的屏幕截图贴到实验报告中
  - 列出出现的问题以及采取的解决方法
  - 独立完成
- 提交
  - 11月24日晚12:00前发到助教邮箱
    - 15651638081@163.com
    - 命名格式：学号+姓名+实验N.doc/docx/rar