数据库系统概论
An Introduction to Database System

第三章 关系数据库标准语言SQL

刘 淇
Email: qiliuql@ustc.edu.cn

课程主页：
http://staff.ustc.edu.cn/~qiliuql/DB2021.html

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	大致页数	发布日期
SQL/86		1986.10
SQL/89(FIPS 127-1)	120页	1989年
SQL/92	622页	1992年
SQL99	1700页	1999年
SQL2003	3600页	2003年
SQL2008	3777页	2006年
SQL2011		2010年

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

- 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

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“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.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

5.语言简洁，易学易用

- SQL功能极强，完成核心功能只用了9个动词。

表 3.1　SQL 语言的动词

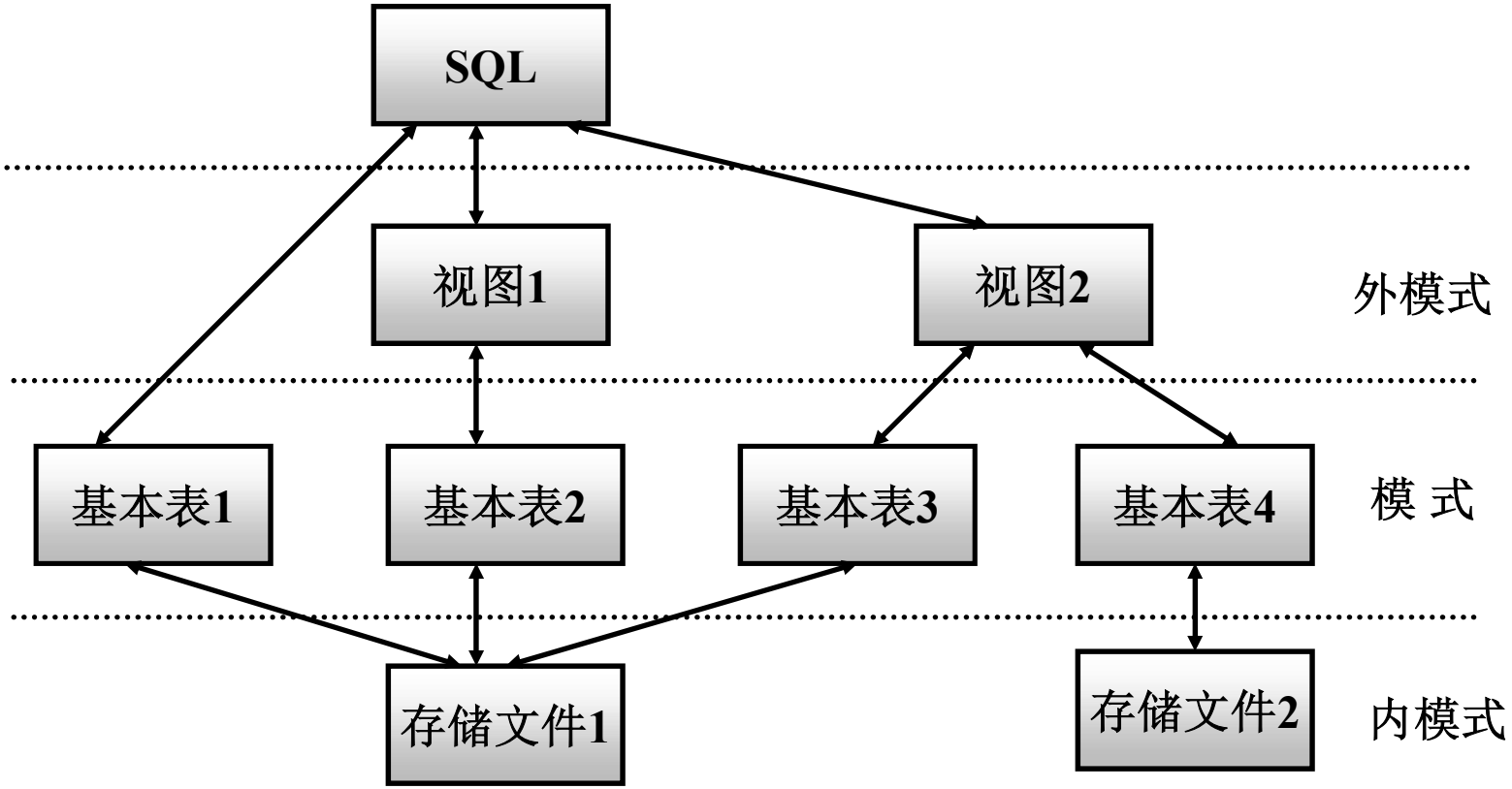
SQL 功　能	动　　词
数 据 查 询	SELECT
数 据 定 义	CREATE, DROP, ALTER
数 据 操 纵	INSERT，UPDATE DELETE
数 据 控 制	GRANT，REVOKE

3.1 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

SQL的基本概念（续）

SQL语言支持关系数据库三级模式结构

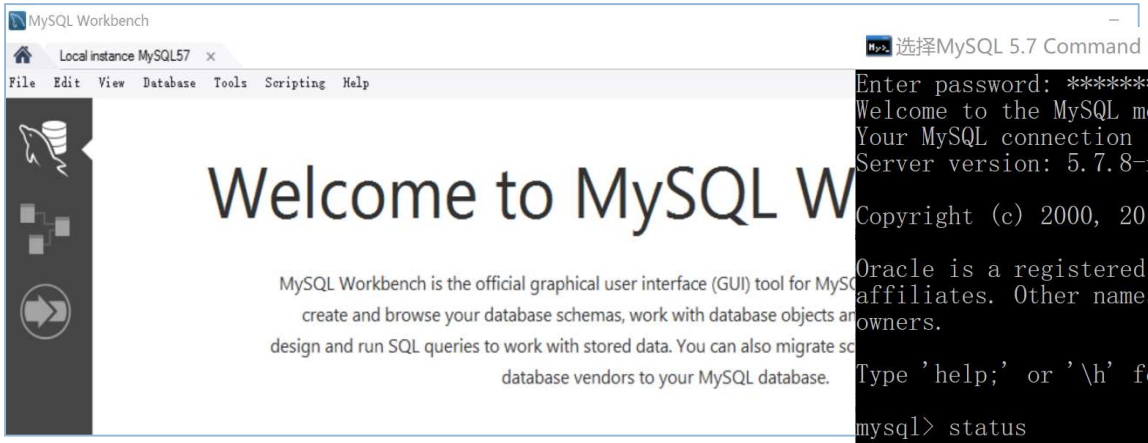


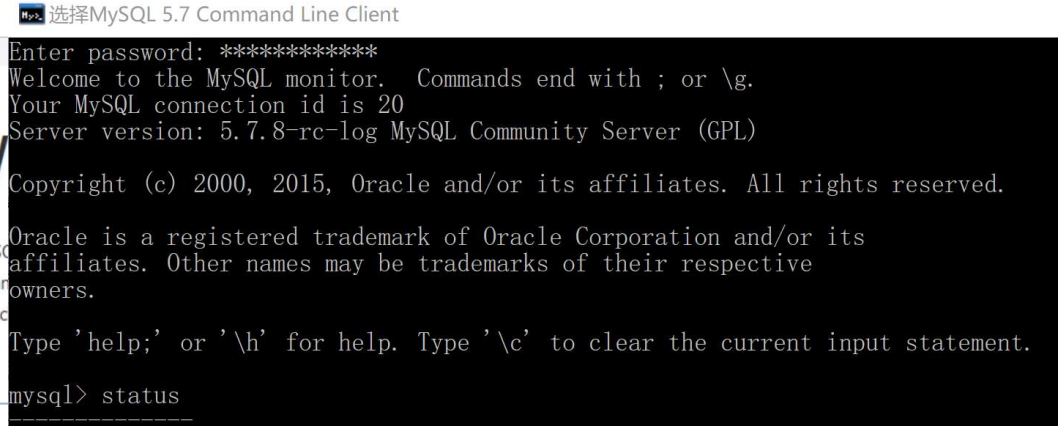
SQL的基本概念（续）

- 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
- 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

Mysql Workbench

- 专为MySQL设计的ER/数据库建模工具
- 是可视化数据库设计、管理的工具，它同时有开源和商业化的两个版本
- 该软件支持Windows和Linux系统
- 安装之前请确保mysql server已经安装在本机
- http://staff.ustc.edu.cn/~qiliuql/DB2021.html





第三章 关系数据库标准语言SQL

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)**

Student表

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

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	PASCAL语言	6	4

SC表

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

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 3.2　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	ALTER INDEX

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

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。
- 语句格式：

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>]

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

- 用户WANG必须是已经存在的数据库用户

[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>

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

QI1

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 SCHEMA TEST;** **--- TEST**
- 创建一个名为S-T的模式呢？
  - **CREATE SCHEMA S-T**

如果你在程序里面写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；

`是键盘左上角的~

分号是个分隔符，看到分号就标志着本条sql语句结束

QI1 QI, 2017/3/10

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。

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

- **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

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

3.3.2 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>] ）；

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*/
        );
```

实际操作-Student

- Create Schema `S-T`;
- Use `S-T`; #we have to use `` for -
- 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)
);
```

实际操作-SC

- Use `S-T`;
- 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

- 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

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，4字节）
SMALLINT	短整数（2字节）
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
    - **Create schema XXX create table XXX**
  - 方法三：设置所属的模式
    - **Use schema XXX**

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)
  );
```

模式与表（续）

- 创建基本表（其他数据库对象也一样）时，若没有指定模式，系统根据搜索路径来确定该对象所属的模式
- **RDBMS**会使用模式列表中第一个存在的模式作为数据库对象的模式名
- 若搜索路径中的模式名都不存在，系统将给出错误
- 显示当前的搜索路径： **SHOW search_path;**
- 搜索路径的当前默认值是：**$user， PUBLIC**。其含义是首先搜索与用户名相同的模式名，如果该模式名不存在，则使用**PUBLIC**模式。

模式与表（续）

- DBA用户可以设置搜索路径，然后定义基本表

  SET search_path TO “S-T”，PUBLIC;

  Create table Student（.....）;

  结果建立了S-T.Student基本表。

  RDBMS发现搜索路径中第一个模式名S-T存在，就把该模式作为基本表Student所属的模式。

有些DBMS，如MYSQL，没有搜索路径的概念

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。
- 删除呢？

alter table Student drop S_entrance;

[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； #Mysql会报错

[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);

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**;

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

删除基本表(续)

[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
```

删除基本表（续）

[例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，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

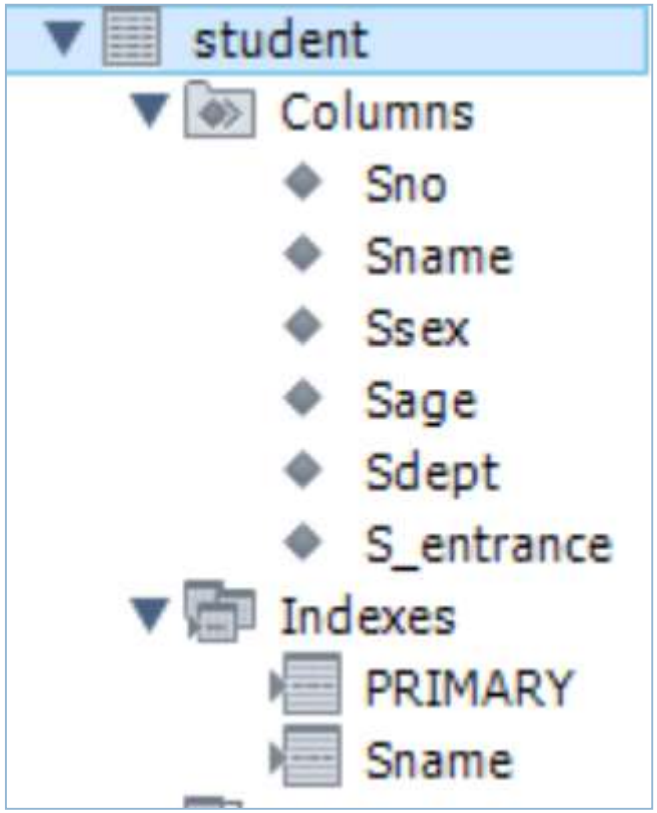
注：MySQL呢?

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

3.3.3 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



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

B+/-树索引

- 从二叉树讲起

二叉排序树上查找某关键字等于给定值的结点过程，其实就是走了一条从根到该结点的路径。

比较的关键字次数

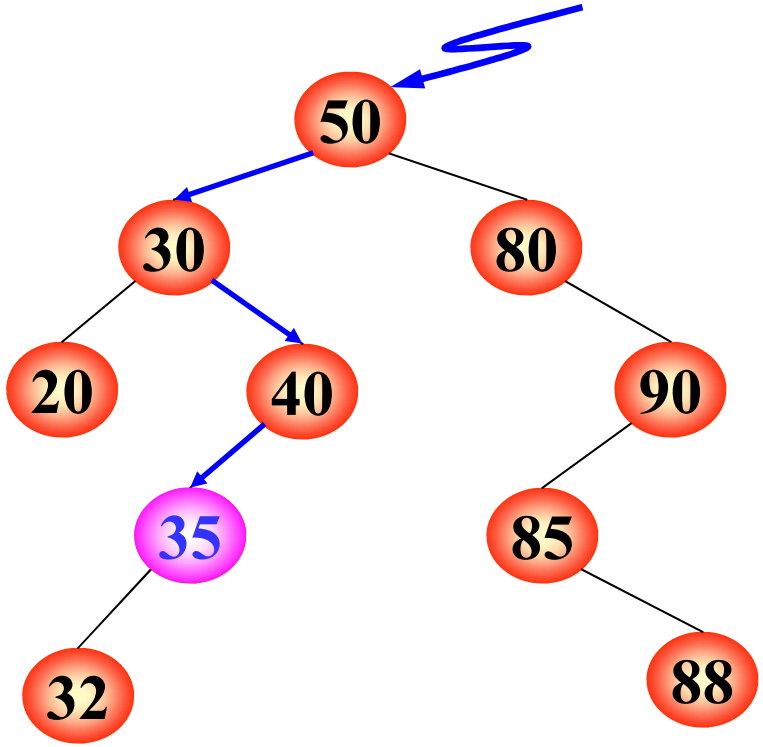
=

此结点所在层次数

最多的比较次数

=

树的深度

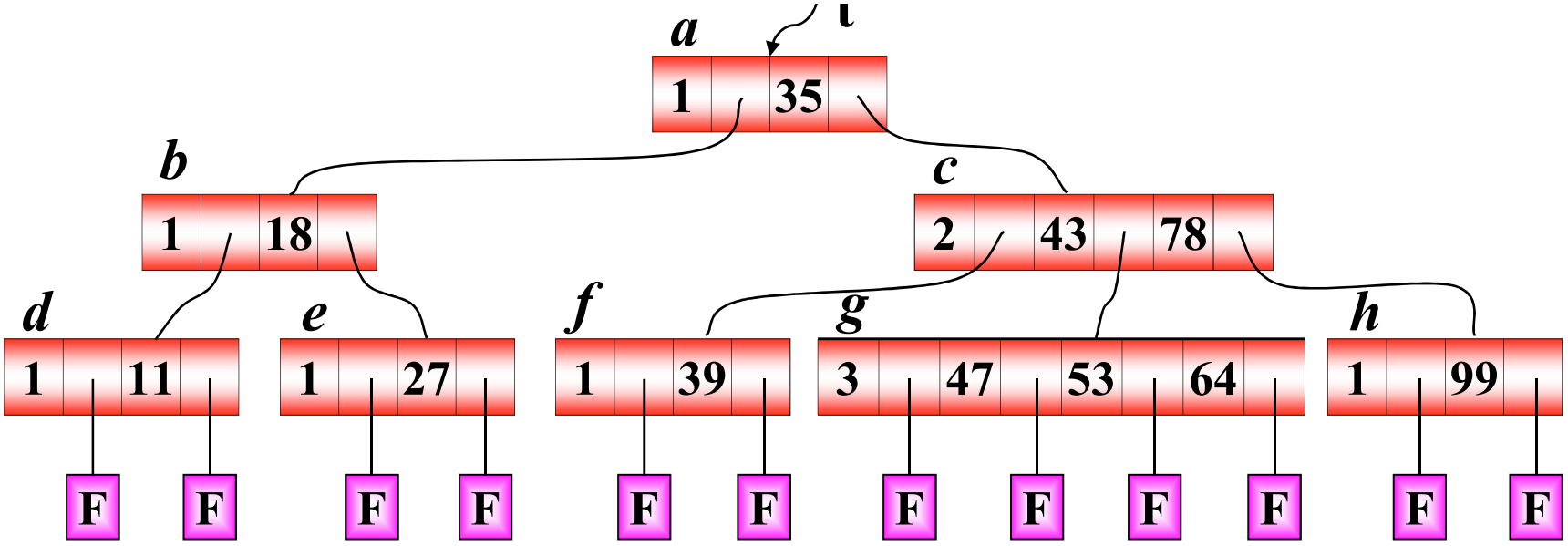


查找关键字：35

B+/-树—B-树索引

一棵 *m* 阶的 B- 树，或为空树或为满足下列特性的 *m* 叉树：

(5)、所有叶子结点在同一个层次上，且不含有任何信息。（可以看作是外部结点或查找失败的结点，实际上这些结点不存在，指向这些结点的指针为空）。

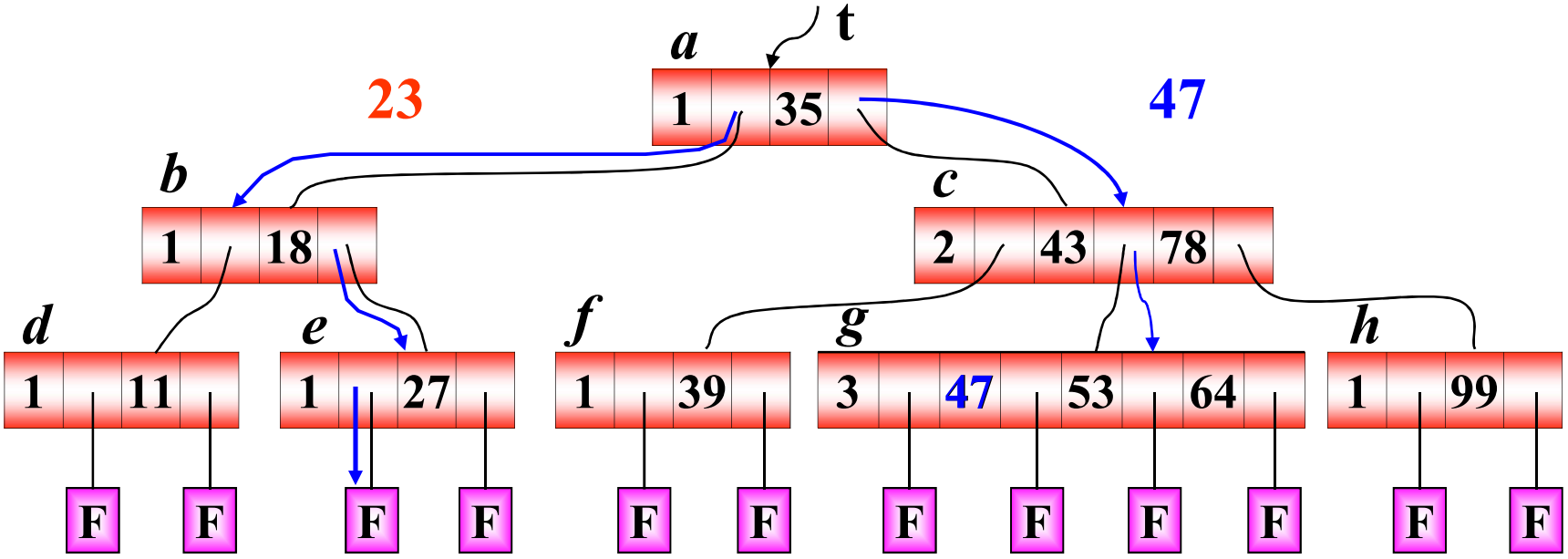


B+/-树—B-树索引

从根结点出发，沿指针搜索结点和在结点内进行顺序（或折半）查找两个过程交叉进行。

若查找成功，则返回指向被查关键字所在结点的指针和关键字在结点中的位置；

若查找不成功，则返回插入位置。



B+树索引

B+树是B-树的变体，也是一种多路搜索树：

1.其定义基本与B-树同，除了：

2.非叶子结点的子树指针与关键字个数相同；

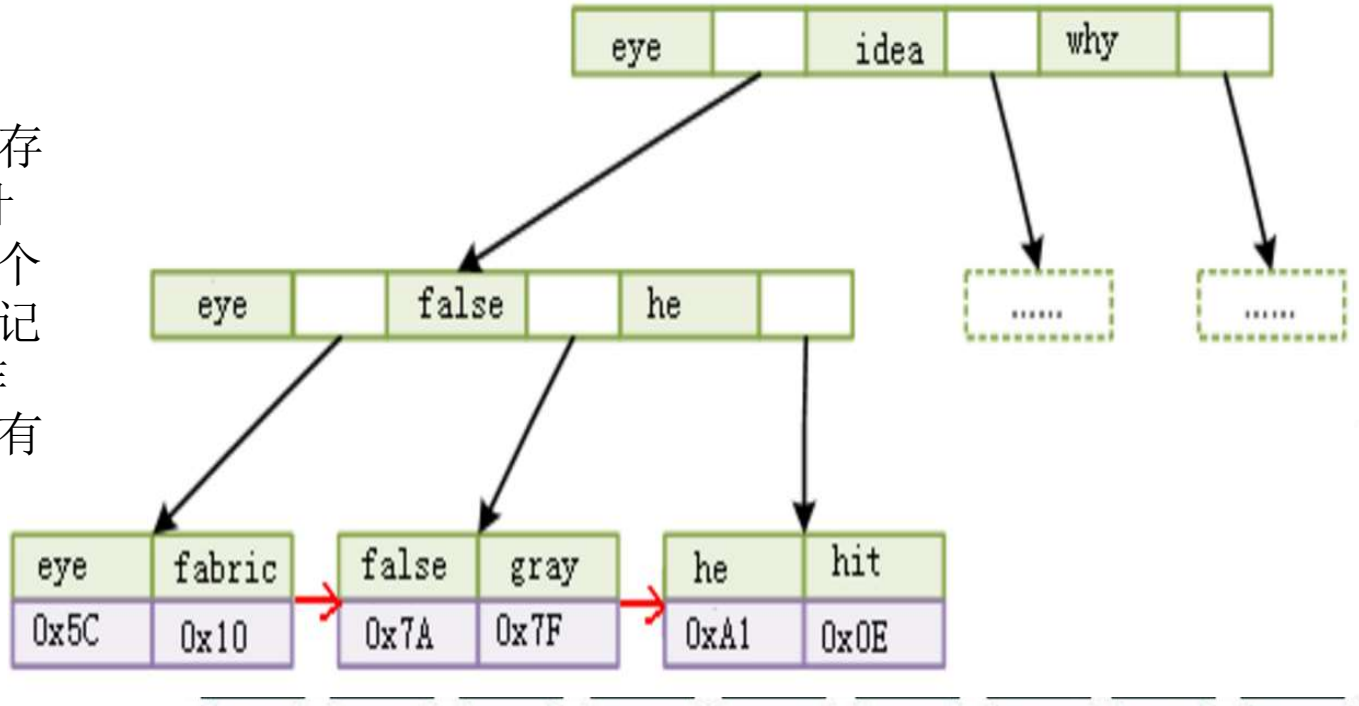
3.非叶子结点的子树指针P[i]，指向关键字值属于[K[i], K[i+1])的子树（B-树是开区间）；

5.为所有叶子结点增加一个链指针；

6.所有关键字都在叶子结点出现；

如：（M=3）

B+Tree：非叶子节点只存key，大大滴减少了非叶子节点的大小，那么每个节点就可以存放更多的记录，树更矮了，I/O操作更少了。所以B+Tree拥有更好的性能。



Hash表索引

- 预先知道所查关键字在表中的位置。即，记录在表中的位置和其关键字之间存在一种确定的关系。

例如：为每年招收的 **1000** 名新生建立一张查找表，其关键字为学号，其值的范围为 **xx000～xx999**（前两位为年份）。

若以下标为**000～999** 的有序顺序表表示之，则查找过程可以简单进行：取给定值（学号）的后三位，不需要经过比较便可直接从顺序表中找到待查关键字。

上例表明，记录的关键字与记录在表中的存储位置之间存在一种对应（函数）关系。若记录的关键字为 key，记录在表中的位置（称为哈希地址）为 f(key)，则称此函数 f(key) 为哈希函数（散列函数）。

Hash表

例如：对于如下 9 个关键字：

{Zhao, Qian, Sun, Li, Wu, Chen, Han, Ye, Dai}

设哈希函数 f(key) = ⌊(Ord(关键字首字母) - Ord('A') + 1) / 2⌋

0	1	2	3	4	5	6	7	8	9	10	11	12	13
	Chen	Dai		Han		Li		Qian	Sun		Wu	Ye	Zhao

问题：若添加关键字 Zhou，会出现什么情况？ Zhou

从这个例子可见：

1) 哈希函数是一个映像，即：将关键字的集合映射到某个地址集合上。它的设置很灵活，只要使得关键字的哈希函数值都落在表长允许的范围之内即可；

2) 由于哈希函数是一个压缩映像，因此，在一般情况下，很容易产生“冲突”现象，即：key1 ≠ key2，而 f(key1) = f(key2)。这种具有相同函数值的关键字称为同义词。

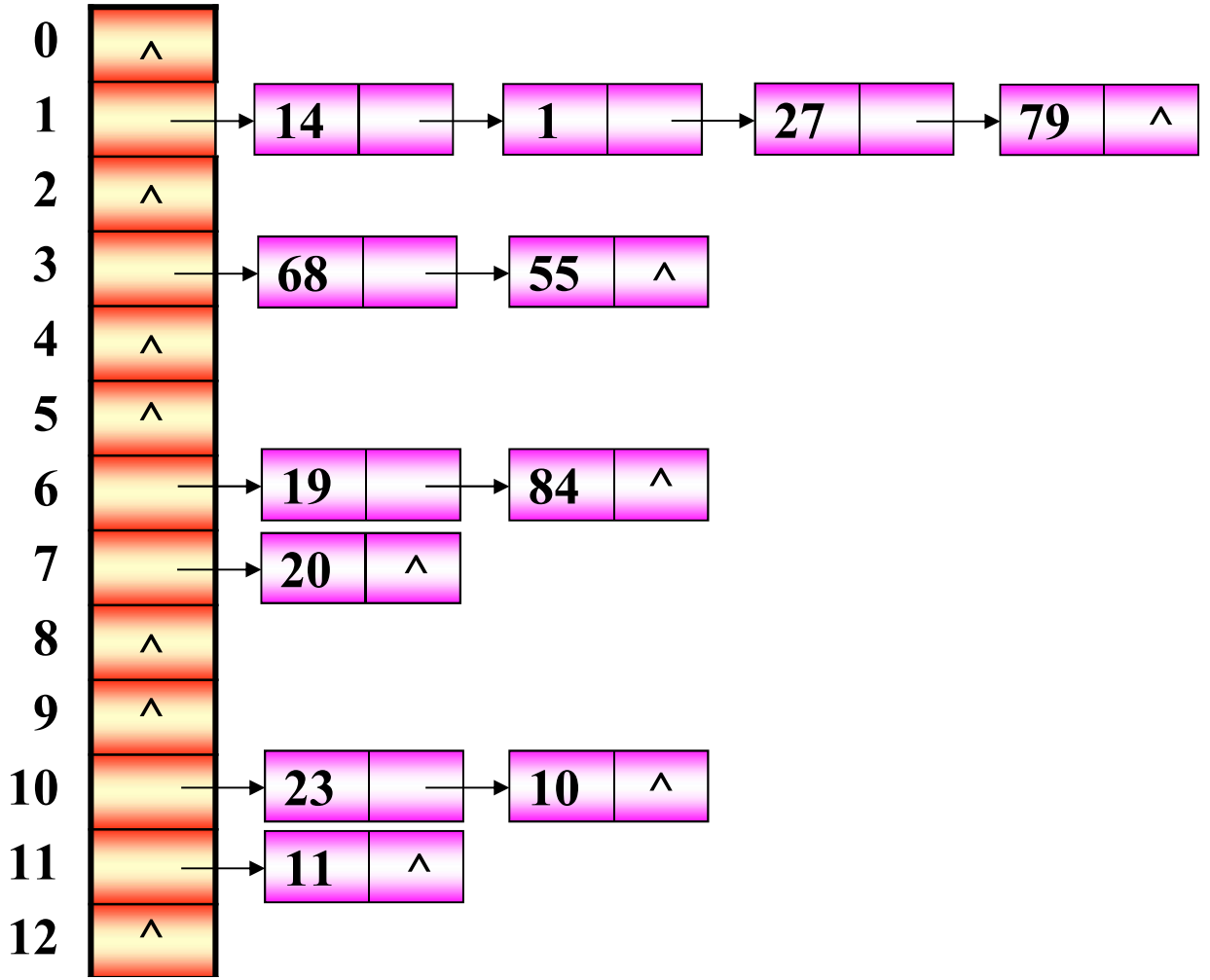
Hash表

- “处理冲突”的实际含义是：为产生冲突的地址寻找另一个哈希地址。

例：已知一组关键字 (19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79)

哈希函数为：

H(key)=key MOD 13，用链地址法处理冲突。



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>]]...)；

mysql中没有cluster index

聚簇索引：元组按照索引键值的顺序存储

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
  - 通常物理顺序结构只有一种，那么一个表的聚簇索引也只能有一个，设置了主码，系统默认就为表加上了聚簇索引。若需要用其他字段作为索引，可在设置主码之前自己手动的先添加上唯一的聚簇索引，然后再设置主码。
  - **B+Tree的叶子节点上的data就是数据本身（主码或其它字段）**
- **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
  - **B+Tree的叶子节点上的data，并不是数据本身，而是数据存放的地址**

细节在第7.5.2节

非聚簇索引比聚簇索引多了一次读取数据的IO操作，所以查找性能上会差。

建立索引（续）

[例13]在Student表的Sname（姓名）列上建立一个聚簇索引

```
CREATE CLUSTER INDEX Stusname
 ON   Student(Sname);
```

- 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。

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);
```

二、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15] 删除Student表的Stusname索引

  DROP INDEX Stusname;

  DROP INDEX Stusname on student;

3.5 数 据 更 新

3.5.1 插入数据

3.5.2 修改数据

3.5.3 删除数据

3.5.1 插入数据

- 两种插入数据方式

  1. 插入元组

  2. 插入子查询结果

  - 可以一次插入多个元组

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）

- **INTO子句**
  - 属性列的顺序可与表定义中的顺序不一致
  - 没有指定属性列
  - 指定部分属性列
- **VALUES子句**
  - 提供的值必须与**INTO**子句匹配
    - 值的个数
    - 值的类型

Into子句没有出现的属性列，新元组在这些列上取Null;

在表定义时说明了Not Null的属性列不能取空，否则报错;

插入元组（续）

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

```
1 • INSERT
2       INTO  Student (Sno, Sname, Ssex, Sdept, Sage)
3       VALUES ("200215129", "陈春", "男", "IS", 18);
4 • SELECT * FROM student:
```

esult Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Sno	Sname	Ssex	Sage	Sdept
200215128	陈冬	男	18	IS
200215129	陈春	男	18	IS

注意单引号或者双引号的使用

插入元组（续）

［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');
```

```
1 • INSERT
2   INTO  Student
3   VALUES ("200215126", "张成民", "男", 18, "CS");
4 • SELECT * FROM student;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Sno	Sname	Ssex	Sage	Sdept
200215126	张成民	男	18	CS
200215128	陈冬	男	18	IS

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);
```

查询表中有哪些属性列

- SHOW COLUMNS FROM 表名;

```
2 • SHOW COLUMNS FROM student;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
Sno	char(9)	NO	PRI	NULL	
Sname	char(20)	YES	UNI	NULL	
Ssex	char(2)	YES		NULL	
Sage	int(11)	NO		NULL	
Sdept	char(20)	YES		NULL	

二、插入子查询结果（select学过以后再介绍）

- 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）

- **INTO子句(与插入元组类似)**

- 子查询
  - **SELECT子句目标列必须与INTO子句匹配**
    - 值的个数
    - 值的类型

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*/
```

插入子查询结果（续）

第二步：插入数据

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

插入子查询结果（续）

RDBMS在执行插入语句时会检查所插元组是否破坏表上已定义的完整性规则

- 实体完整性
- 参照完整性
- 用户定义的完整性
  - **NOT NULL**约束
  - **UNIQUE**约束
  - 值域约束

3.5 数 据 更 新

3.5.1 插入数据

3.5.2 修改数据

3.5.3 删除数据

3.4.2 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

修改数据（续）

- **SET子句**
  - 指定修改方式
  - 要修改的列
  - 修改后取值
- **WHERE子句**
  - 指定要修改的元组
  - 缺省表示要修改表中的所有元组

修改数据（续）

- 三种修改方式

1. 修改某一个元组的值
2. 修改多个元组的值
3. 带子查询的修改语句

1. 修改某一个元组的值

[例5] 将学生200215121的年龄改为22岁

```
1 •     UPDATE  Student
2       SET Sage=22
3       WHERE  Sno="200215126";
4 •  SELECT * FROM student:
```

ult Grid | Filter Rows: | Edit: | Export/Import:

Sno	Sname	Ssex	Sage	Sdept
200215126	张成民	男	22	CS
200215128	陈冬	男	18	IS
200215129	陈春	男	18	IS

2. 修改多个元组的值

[例6] 将某个学生的年龄增加1岁

```
1 •  UPDATE Student
2     SET Sage= Sage+1
3      WHERE  Sno="200215126";
4 •   SELECT * FROM student;
```

ult Grid | Filter Rows: | Edit: | Export/Import: | Wr

Sno	Sname	Ssex	Sage	Sdept
200215126	张成民	男	23	CS

3. 带子查询的修改语句

[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);
```

修改数据（续）

RDBMS在执行修改语句时会检查修改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

3.5 数 据 更 新

3.5.1 插入数据

3.5.2 修改数据

3.5.3 删除数据

3.5.3 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）

- 三种删除方式

1. 删除某一个元组的值

2. 删除多个元组的值

3. 带子查询的删除语句

1. 删除某一个元组的值

[例8] 删除学号为200215128的学生记录。

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

```
1 • DELETE
2   FROM Student
3   WHERE Sno="200215128";
4 • SELECT * FROM student;
```

esult Grid | Filter Rows: | Edit: | Export/Import:

Sno	Sname	Ssex	Sage	Sdept
200215126	张成民	男	23	CS
200215129	陈春	男	18	IS
NULL	NULL	NULL	NULL	NULL

2. 删除多个元组的值

[例9] 删除所有的学生选课记录。

```
DELETE
FROM SC;
```

3. 带子查询的删除语句

[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);
```