**RDBMS：关系型数据库**

DB2、MySQL、PostgreSQL

MongoDB：文档型非关系数据库

Redis：KV键值数据库

HBase：列数据库

Hadoop：分布式数据库

**四个特性：**

原子性、持久性、隔离性、一致性

**SQL语句主要分为三类：**

DDL（Data Definition Languages）数据定义语言。用来定义数据库，数据表，列，索引等数据库对象。关键字主要是：create、drop、alter

DML（Data Manipulation Language）数据操纵语句。用于添加、删除、更新和查询数据库记录，并检查数据完整性。关键字主要是：insert、delete、update和select等

DCL（Data Control Language）数据控制语句。用于控制不同数据段直接的许可和访问级别。这些语句定义了数据库、表、字段、用户的访问权限和安全级别。关键字主要是：grant、revoke（收回权限）

**DDL语句**

1. **创建数据库**

CREATE DATABASE dbname;

1. **查询系统中存在的数据库**

SHOW DATABASES;

1. **选择要使用的数据库**

USE dbname;

1. **查看数据库中的所有数据表**

SHOW tables;

1. **删除数据库**

DROP DATABASE dbname;

1. **创建表**

CREATE TABLE tablename(

column\_name1 column\_type\_1 constraints,

column\_name2 column\_type\_2 constraints,

…

column\_namen column\_type\_n constraints,

);

1. **查看表**

（1）Show create table tableName;

（2）desc tableName;

1. **修改表**

ALTER TABLE tablename MODIFY [column] column\_definition [FIRST\AFTER col\_name]

Eg. 修改表student的sname字段定义，将varchar（10）改为varchar（20）：

ALTER TABLE student MODIFY sname varchar(20);

Eg. 在student表中增加一个age字段

ALTER TABLE student ADD [column] age int(3);

(一般情况下，增加一个字段，该字段被加入到表的最后一个位置，如果想要自定义位置可以用first和after两个关键字)

Eg. 将sname字段加在sno之后

ALTER TABLE student ADD [COLUMN] sname varchar(10) AFTER sno;

Eg. 将sname字段放在最前面

ALTER TABLE student MODIFY COLUMN sname varchar(10) FIRST;

Eg. 将age字段从表中删除

ALTER TABLE student DROP COLUMN age;

Eg. 将age改名为age1，同时修改字段类型为int(4)

ALTER TABLE student CHANGE age age1 int(4);

(change和modify都可以修改表的定义，change后面需要写两次列名，不方便。但是change可以修改列的名称，modify则不能)

Eg. 将表student改名为mystudent

ALTER TABLE student RENAME mystudent;

**DML语句**

1. **插入记录**

INSERT INTO tablename (field1, field2, … , fieldn) VALUES (value1, value2, … , valuen);

也可以不指定字段名称，但是values后面的顺序应该和数据表中的顺序排列一致，且每一个都要写。

一次插入多条记录：

INSERT INTO tablename (field1, field2, … , fieldn)

VALUES

(record1\_value1, record1\_value2, … , record1\_valuen),

(record2\_value1, record2\_value2, … , record2\_valuen),

…

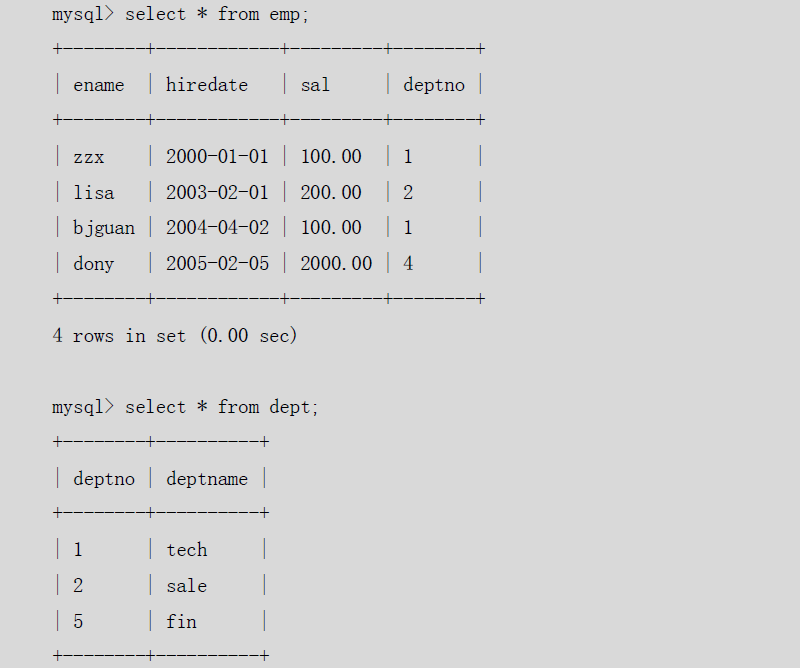
(recordn\_value1, recordn\_value2, … , recordn\_valuen);

1. **更改记录**

UPDATE tablename SET field1=value1, field2=value2, … , fieldn=valuen [WHERE CONDITION];

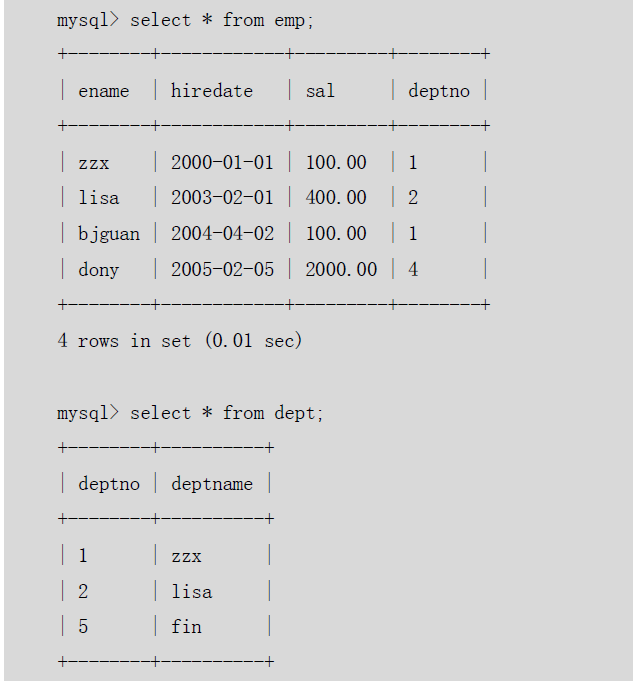
也可以同时更新多个表中的字段

UPDATE t1, t2, … , tn SET t1.field1=value1, …, tn.fieldn=valuen [WHERE CONDITION];



同时更新表emp中的字段sal和表dept中的字段deptname：

Eg. UPDATE emp a, dept b SET a.sal=a.sal\*b.deptno, b.deptname=a.ename where a.deptno=b.deptno;



1. **删除记录**

DELETE FROM tablename [WHERE CONDITION]

（如果不加where condition则会删除表中所有的数据）

一次删除多个表中的数据：

DELETE t1, t2, … ,tn FROM t1, t2, …, tn [WHERE CONDITION];

(如果from后面是表的别名，则delete后面也要用相应的别名，否则会提示语法错误)

DELETE a, b FROM emp a, dept b WHERE a.deptno=b.deptno and a.deptnp=3;

1. **查询记录**

SELECT \* FROM tablename [WHERE CONDITION];

SELECT col1,col2,… FROM tablename [WHERE CONDITION];

**（1）去掉重复的记录：**

Select distinct deptno from staff;

**（2）条件查询：**

Select \* from staff where deptno=1;

**（3）排序和限制**

SELECT \* FROM tableName [WHERE CONDITION] [ORDER BY field1 [DESC/ASC]，field2[DESC/ASC]];

desc是降序排列，asc是升序排列，如果不写，则默认为是升序排列。

（如果排序的字段的值一样，则值相同的字段按照第二个排序字段进行排序；如果只有一个排序字段，那么这些相同的值将会无序排列）。

Eg：select \* from staff order by deptno, sal desc; (按照部门号升序排列，部门号相同的按照工资降序排列)。

只希望显示一部门数据：LIMIT关键字

Eg：select \* from staff order by salary limit 3; 显示前三条记录

Eg：select \* from staff order by salary limit 1,3; 显示从第二条开始的3条记录。

**（4）多查询条件**

在学生表 Student 的系别 (Sdept) 属性中查询信息系 (IS) 、数学系 (MA) 和计算机系 (CS) 的学生姓名 (Sname) 和性别 (Ssex) :

SELECT Sname, Ssex FROM Student WHERE Sdept IN( ‘IS’ , ‘MA’ , ‘CS’ );

**（5）聚合（汇总操作）**

统计学生表中的总人数：Select count(1) from taff;

统计各个部门的总人数：Select deptno，count(1) from staff group by deptno;

既要各部门总人数，又要员工总数：

select deptno，count(1) from staff group by deptno **with rollup**;

**（6）表连接**

内连接和外连接

内连接：只选出两张表中互相匹配的记录。

外连接：会选出其他不匹配的记录。

选出学生的名字和所在院系名称：

Select student, deptname from student, studep where student.sno=studep.sno;

左连接：左表中全部记录和右表中与左表匹配的记录（P43）

右连接：右表中全部记录和左表中与右表匹配的记录

**DCL语句**

 grant 权限1,权限2,…权限n on [**数据库**](javascript:;)名称.表名称 to 用户名@用户地址 identified by ‘连接口令’;

grant select, insert on databaseName.\* to ‘user1’@’localhost’ identified by ‘dorothy’;

revoke insert on databaseName.\* from ‘user1’@’localhost’.

**视图**

是从一个或多个基本表导出的表，它是虚表。数据库中只存放视图的定义，而不是实际的数据，数据仍然保存在基本表中。如果基本表的数据变化，视图中的数据也随之变化。

视图一经定义，可以像基本表一样被查询和删除，也可以在视图上在定义一个新的视图，但对视图的更新存在一定的限制。

**建立视图：**

CREATE VIEW IS\_STUDENT

AS

SELECT Sno,Sname,Sage

FROM Student

WHERE Sdept=’IS’;

**更新时进行检查：WITH CHECK OPTION：**

CREATE VIEW IS\_STUDENT

AS

SELECT Sno,Sname,Sage

FROM Student

WHERE Sdept=’IS’

WITH CHECK OPTION；插入和修改视图时，数据库系统会自动加上Sdept=’IS’的条件。

**在视图上建立视图：**

建立一个IS\_S1视图，显示选修了课程号为1的学生信息：

CREATE VIEW IS\_S1（Sno，Sname，Grade）

AS

SELECT Student.Sno,Sname,Sage

FROM Student，SC

WHERE Sdept=’IS’ AND Student.Sno=SC.Sno AND SC.Cno=‘1’；

建立信息系选修了1号课程且成绩为90分以上的学生的视图：

CREATE VIEW IS\_S2

AS

SELECT Sno,Sname,Sage

FROM IS\_S1

WHERE Grade>90;

删除视图：

Drop view viewName [cascade]

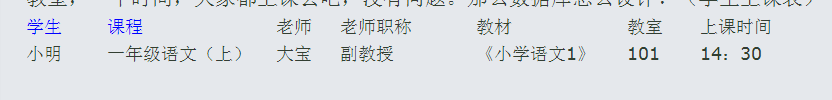
加上cascade就会将由该视图导出的所有视图全部删除

**范式**

第一范式：如果数据库表中的所有字段值都是不可分解的原子值，就说明该数据库表满足了第一范式。

e.g．：一个数据库表中有这样的字段birth（year，month，day），就不满足第一范式

第二范式：满足第一范式，并且每一个非主属性完全函数依赖于码。也就是说要确保数据库表中的每一列都与主键相关，而不能只与主键的某一部分相关（主要针对联合主键而言）。



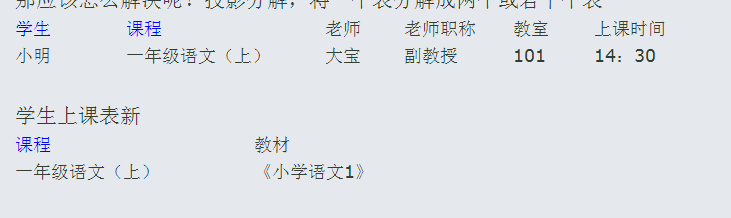
该表中的主键是学生+课程，而教材依赖于课程，属于部分依赖，这样做的坏处是：

插入异常：当学生还没选课，没有学号，而要插入一个课程信息，这时主键中的学号为空，不能插入

删除异常：学生第二学期没有语文课，将本门课删除，从此在该表中将查找不到关于语文课的信息

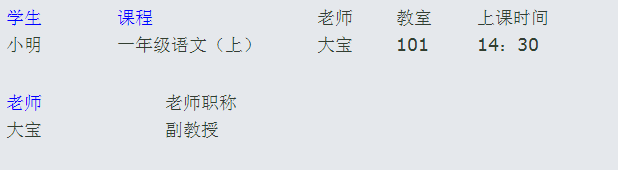
更新异常：要将语文课教材更换，如果有10000名学生选了语文课，就要更新10000条语句。

所以应该将表分解成如下的样子。



第三范式：满足2NF，并且消除传递依赖。数据库表中的每一列都要和主键直接相关，而不是间接相关。

在上表中，老师职称可直接依赖于老师，所以构成了间接依赖，不满足第三范式。应该修改为：



**触发器**

创建触发器：

CREATE TRIGGER trigger\_name trigger\_time trigger\_event ON table\_name FOR EACH ROW BEGIN trigger\_stmt;

对于同一张表，相同触发时间的相同触发时间，只能定义一个触发器。例如：对两个字段定义after update触发器，只能定义一个。

为film表创建after insert的触发器：

Create trigger ins\_film

After insert on film for each row begin

Insert into film\_text (film\_id, title, description) values (new.film\_id, new.title, new.description)

End;

删除触发器：drop trigger trigger\_name;

查看触发器：show triggers；

select foo,count(foo)from pokes where foo>10 group by foo having count (\*)>5 order by foo

执行顺序是：from -> where -> group by -> having -> select -> order by

外键必须可以找到或者为空：

