创建表时需要考虑的基本问题：

1. 表包含那些数据类型
2. 名称
3. 哪个字段做主键
4. 每一列的数据类型
5. 每一列的长度
6. 哪些列可以是null

表有专门的命名规范

PHONE\_TBL ENPLOEE\_TBL

对CREATE TABLE 与SELECT TABLE 联合用可以复制现有的表，新表具有同样的列定义如：

CREATE TABLE NEW\_TABLE\_NAME AS

SELECT [ \*/COLUMN1] FROM OLD\_TABLE WHERE 条件

数据库的规格化

把原始数据库分解为表的过程称为规格化。

是去除数据库中冗余数据的过程，在设计数据库时使用

数据库设计时应该考虑的因素：

1. 数据库应该保存什么数据？
2. 用户如何访问数据库？
3. 用户需要什么权限？
4. 数据库中的数据如何分组
5. 哪些数据经常被访问
6. 全部数据与数据库如何关联
7. 采用什么措施保证数据正确
8. 怎样减少数据冗余
9. 如何易用

三种设计范式：

第一范式：没有冗余数据，主键必须能够唯一定义表中的记录。否则查询结果可能会返回重复的记录。

第二范式：有部分冗余数据，目标是提取与主键只有部分关系的数据

第三范式：目标是删除表里不依赖于主键的数据。

格式化缺点：降低数据性能，需要更多的cpu，io

格式化优点：更好的组织，数据一致性好，数据设计更灵活，更安全

去格式化：增加性能，在格式化中进行调整，但维护成本高，保证数据一致性难度大

#############################################################

Sql控制语句：

从另一个表中插入数据

INSERT INTO TABLE\_NAME [（字段1，字段2]] SELECT [ \*] FROM TABLE\_NAME [WHERE ….]

注意：在UPDATE 命令中一定要有 where语句，否则会将表中所有字段都更新。

################################################################

事务

控制事务命令：

COMMIT

ROLLBACK

SAVEPOINT

控制事务命令只与DML(IINSERT UPDATE DELETE)配合使用

格式均为：COMMIT；

SAVEPOINT savepoint name

ROLLBACK TO SAVEPOINT\_NAME

比如：

DELETE FROM CLASS\_TBL WHERE id = ‘1234’;

SAVEPOINT SP1;

DELETE FROM CLASS\_TBL WHERE id = ‘90’;

SAVEPOINT SP2 名字必须唯一

RELEASE SAVEPOINT SP1;

删除创建的SP1节点

SET TRANSACTION READ ONLY (设置表的事务是什么权限)

SELECT DISTINCT(pro\_name) FROM CANDY\_TBL

不重复的显示pro\_name字段

ORDER BY:

SELECT \* FROM TABLE\_TBL WHERE cost < 20 ORDER BY Id（1） ASC（升序）/DESC（降序）id处也可以写为数字位置表示

更改别名，将PROD更改为product

SELECT PROD,product FROM …

逻辑判断：

=

<>不等于!=

逻辑操作符号：对SQL关键字的比较

IS NULL ------------- WHERE SALARY IS NULL

BETWEEN ------------- WHERE SALARY BETWEEN ‘2W’ AND ‘3.5W’

IN --------- WHERE SALARY IN (‘20000’,’3000000’)至少有一个匹配即可

LIKE --------- ‘%’表示0，1，n个字符‘\_’表示一个数字字符

EXISTS ------WHERE EXISTS （select cost from ASD\_TBL where cost > 100）

UNIQUE

ALL ---把一个值与另一个集合中的所有值进行比较

WHERE SALARY > ALL (select cost from ASD\_TBL where cost >100)

ANY --- 把一个值与。。。任意一个值比较

汇总函数：

Max，min，avg，count

#####################################

数据排序与分组

用SELECT + GROUP BY 实现子句的分组。

SELECT man\_id ,city FROM student\_TBL WHERE SALARY > 20000 GOURP BY city ，man\_id ORDER BY SALARY

根据city，man\_id将salary > 20000 的man\_id ，city查找出来，每个以SALARY升序

**GROUP BY 与ORDER BY 的区别：**

order by 从英文里理解就是行的排序方式，默认的为升序。 order by 后面必须列出排序的字段名，可以是多个字段名。

group by 从英文里理解就是分组。必须有“聚合函数”来配合才能使用，使用时至少需要一个分组标志字段。

什么是“聚合函数”？

像sum()、count()、avg()等都是“聚合函数”

使用group by 的目的就是要将数据分类汇总。

一般如：

select 单位名称,count(职工id),sum(职工工资) form [某表]

group by 单位名称

这样的运行结果就是以“单位名称”为分类标志统计各单位的职工人数和工资总额。

在sql命令格式使用的先后顺序上，group by 先于 order by。

注意：使用group by 需要将所有非汇总函数放入，一般用在汇总函数上

Group by 可以用在create view 上而order by 不行

Having：之于Group by = Where 之于 Select

限制其条件用。

Group by city Having SALARY > 20000

**ROLLUP CUBE**

如果是ROLLUP(A, B, C)的话，首先会对(A、B、C)进行GROUP BY，然后对(A、B)进行GROUP BY，然后是(A)进行GROUPBY，最后对全表进行GROUP BY操作。如果是GROUP BY CUBE(A, B, C)，则首先会对(A、B、C)进行GROUP BY，然后依次是(A、B)，(A、C)，(A)，(B、C)，(B)，(C)，最后对全表进行GROUP BY操作。 grouping\_id()可以美化效果：

SQL> select index\_type, status, count(\*) from t group by index\_type, status;

INDEX\_TYPE STATUS COUNT(\*)

--------------------------- -------- ----------

LOB VALID 51

NORMAL N/A 25

NORMAL VALID 479

CLUSTER VALID 11

Grouping() 显示前面的字窜

SQL> select grouping(index\_type) g\_ind, grouping(status) g\_st, index\_type,status, count(\*)

2 from t group by rollup(index\_type, status) order by 1, 2;

G\_IND G\_ST INDEX\_TYPE STATUS COUNT(\*)

---------- ---------- --------------------------- -------- ----------

0 0 LOB VALID 51

0 0 NORMAL N/A 25

0 0 NORMAL VALID 479

0 0 CLUSTER VALID 11

0 1 LOB 51

0 1 NORMAL 504

0 1 CLUSTER 11

1 1 566

下面看看CUBE语句。

SQL> select grouping(index\_type) g\_ind, grouping(status) g\_st, index\_type,status, count(\*)

2 from t group by cube(index\_type, status) order by 1, 2;

G\_IND G\_ST INDEX\_TYPE STATUS COUNT(\*)

---------- ---------- --------------------------- -------- ----------

0 0 LOB VALID 51

0 0 NORMAL N/A 25

0 0 NORMAL VALID 479

0 0 CLUSTER VALID 11

0 1 LOB 51

0 1 NORMAL 504

0 1 CLUSTER 11

1 0 N/A 25

1 0 VALID 541

1 1 566

调整数据的外观：

常用：字符函数：

串接函数：

SELECT CONCAT(‘John’,’.’,’Son’)----mysql中用concat（‘A’，‘B’）

TANSLATE 找到对应的字符并替代：

SELECT CITY TRANSLATE (CITY,’IND’,’ABC’) 将凡是CITY中出现的IND替换为ABC，预定I换为A N---B D---C

REPLACE 将制定的字符串换为另外的字符串：

‘IND’ --- ‘ABC’

**UPPER/LOWER**

UPPER（CITY）把CITY中的所有信息换成大写

**INSTR SUBSTR**

**INSTR**

**在一个字符串中查找指定的字符,返回被查找到的指定的字符的位置**

instr(sourceString,destString,start,appearPosition)

instr（'源字符串' , '目标字符串' ,'开始位置','第几次出现'）

SQL> select instr('qinyinglianqin','qin', 1, 2) position from dual;

POSITION

**SUBSTR()函数**

1.用处:是从给定的字符表达式或备注字段中返回一个子字符串。

2.语法格式:SUBSTR(cExpression，nStartPosition [，nCharactersReturned])

其中，

cExpression指定要从其中返回字符串的字符表达式或备注字段；

nStartPosition用于指定返回的字符串在字符表达式或备注字段中的位置，

nCharactersReturned用于指定返回的字符数目，缺省时返回字符表达式的值结束前的全部字符

3.举例:STORE'abcdefghijlkm' To mystring

SUBSTR(mystring ,1,5) 显示 "abcde" 　1 从第一个字符开始 截取的字符中，包括第一个字符

SQL 中的 TRIM 函数是用来移除掉一个字串中的字头或字尾。最常见的用途是移除字首或字尾的空白。这个函数在不同的资料库中有不同的名称：

。

　TRIM ( [ [位置] [要移除的字串] FROM ] 字串): [位置] 的可能值为 LEADING (起头), TRAILING (结尾), or BOTH (起头及结尾)。 这个函数将把 [要移除的字串] 从字串的起头、结尾，或是起头及结尾移除。如果我们没有列出 [要移除的字串] 是什么的话，那空白就会被移除。

LTRIM(字串): 将所有字串起头的空白移除。

RTRIM(字串): 将所有字串结尾的空白移除。

ITRIM/RTRIM

SQL ltrim()函数实例代码教程 - 返回字符串str的前导(左边)空格字符去掉。

返回字符串str的前导(左边)空格字符去掉

IFNULL

用于在表达式是NULL的时候将NULL换成IFNULL（）中的值

SELECT PAGE IFNULL（PAGE，999999）将PAGE中的NULL显示为9999999

L/RPAD左右填充

SELECT LPAD(PRE,10,’>’) PRODUCT FROM PRODUCTION\_TBL

>>>>>>>>>>12

>>>>>>>>>>13

算术函数：

ABS绝对值

SQRT平方根

EXP指数。。。

ABS（-11）---11

时间函数：

DAYNAME/DAYWEEK（date） 星期几

DAYMONTH（date）几日

DAYYEAR（）一年中的第几天

时间相加：

DATETIME中的值

DATE ‘2010-10-31’ + INTERVAL ‘1’ MONTH

把日期化为字符串：

与把字符串化为日期》

SELECT STR\_TO\_DATE(‘01/01/2010 12:00:00 AM’) AS FOR\_DATE