

第一范式（1NF）：原子性，数据不可再分：一个表中的数据（字段值）不可再分。我们来看一个不良做法：

学生成绩表			
学生	课程	学分	成绩
罗弟华, 男, 籍贯江西	html	2	88
罗弟华, 男, 籍贯江西	javascript	5	90
韩顺平, 男, 籍贯四川	html	2	77
韩顺平, 男, 籍贯四川	php	10	95

此列将多项信息合在一起了，不具有原子性，应拆开。
当然，实际中，此原子性要求是很容易达到的，一般不会有问题。

对其进行修改后，就符合第一范式了：

学生成绩表					
学生	性别	年龄	课程	学分	成绩
罗弟华	男	江西	html	2	88
罗弟华	男	江西	javascript	5	90
韩顺平	男	四川	html	2	77
韩顺平	男	四川	php	10	95

此时，每一项信息都是不可再分的了，就符合了原子性。

第二范式（2NF）

使每一行数据具有唯一性，并消除数据之间的“部分依赖”；

实际，有 2 个方面的要求：

- 1，每一行数据具有唯一性：只要给表设计主键，就可以保证唯一性。
- 2，消除数据之间的“部分依赖”；

什么叫做“依赖”？

依赖，就是在一个表中，其中某个字段的值 B 可以由另一个字段的值 A 来“决定”，则此时我们就称为：

字段 B 依赖于字段 A，

或：

字段 A 决定字段 B；

其意思是，如果根据字段 A 的某个值，一定可以找出一个确定的字段 B 的值，就是 A 决定 B。

对一个有主键的表，这种情况就成了：

确定了主键字段的值，则其他字段肯定都确定了。

也可以这样说：

主键决定了其他字段；

其他字段依赖于主键；

数据操作语言：

插入数据（增）：

多种语法形式：

- 形式 1: insert [into] 表名 [(字段名 1, 字段名 2, ...)] values (值表达式 1, 值表达式 2, ...), (...), ...;
- 形式 2: replace [into] 表名 [(字段名 1, 字段名 2, ...)] values (值表达式 1, 值表达式 2, ...), (...), ...;
- 形式 3: insert [into] 表名 [(字段名 1, 字段名 2, ...)] select 字段名 1, 字段名 2, ... from 其他表名;
- 形式 4: insert [into] 表名 set 字段名 1=值表达式 1, 字段名 2=值表达式 2, ...;

多种语法形式：

- 形式 1: insert [into] 表名 [(字段名 1, 字段名 2, ...)] values (值表达式 1, 值表达式 2, ...), (...), ...;
 - 这是最常用的插入语句，可以一次性插入多行数据，用逗号隔开；
 - 插入语句，要理解为每次插入数据都是以“行”为单位进行插入。
 - 字段名通常不推荐省略，其后续的“值列表”应该跟字段名列表“一一对应”；
 - 其中的值表达式，可以是一个“直接”，或“函数调用结果”，或变量值；其中，如果对应字段是字符或时间类型，则直接值应该使用单引号；
- 形式 2: replace [into] 表名 [(字段名 1, 字段名 2, ...)] values (值表达式 1, 值表达式 2, ...), (...), ...;
 - 其跟 insert into 几乎一样：唯一区别是，如果插入的数据
- 形式 3: insert [into] 表名 [(字段名 1, 字段名 2, ...)] select 字段名 1, 字段名 2, ... from 其他表名;
- 形式 4: insert [into] 表名 set 字段名 1=值表达式 1, 字段名 2=值表达式 2, ...;

删除数据（删）：

语法形式：

delete from 表名 【where 条件】 【order by 排序字段】 【limit 限定行数】；

说明：

- 1，删除数据仍然是以“行”为单位进行；
- 2，通常删除数据都需要带 where 条件，否则就会删除所有数据（这很不常见）；
- 3，where 条件的语法跟 select 中的语法一样；
- 4，order by 排序设定，用于指定这些数据的删除顺序；它通常跟 limit 配合使用才有意义；
- 5，limit 限定用于设定删除多少行（按 order by 设定的顺序）；
- 6，实际应用中，通常很少用到 order by 和 limit

修改数据（改）：

语法形式：

update 表名 set 字段 1=值 1, 字段 2=值 2, 【where 条件】 【order by 排序字段】 【limit 限定行数】；

说明：

- 1，通常，update 语句，也都需要 where 条件，否则：就会修改所有数据（这很少见）；
- 2，where 条件的语法跟 select 中的语法一样；
- 3，order by 用于设定修改的顺序，limit 用于设定修改的行数，他们通常也是结合使用（虽然都很少用）
- 4，实际应用中，通常很少用到 order by 和 limit，则修改数据的常规使用形式就成为：
update 表名 set 字段 1=值 1, 字段 2=值 2, where 条件
- 5，注意：其中的“数据值”，如果是字符串或时间类型的“直接值”，就应该用单引号；

基本查询

基本语法形式

`select [all | distinct] 字段或表达式列表 [from 子句] [where 子句] [group by 子句] [having 子句] [order by 子句] [limit 子句];`

```
1, 字段, 自然是来源于“表”, 则其必然依赖于 from 子句;
2, 表达式是类似这样一个内容: 8, 8+3, now().

mysql> select 8;
+----+
| 8 |
+----+

mysql> select 8+3;
+----+
| 8+3 |
+----+
| 11 |
+----+

mysql> select 8, 8+3, now();
+----+-----+-----+
| 8 | 8+3 | now() |
+----+-----+-----+
| 8 | 11 | 2015-07-24 14:59:49 |
+----+-----+-----+
1 row in set (0.00 sec)

mysql> select 8, 8+3, concat('aa',
+----+-----+-----+
| 8 | 8+3 | concat('aa', 'bb') |
+----+-----+-----+
```

all 和 distinct

用于设定 select 出来的数据, 是否消除“重复行”, 可以不写, 那就是默认值 all:

all: 表示不消除, 即所有都出来, 默认值;

distinct: 表示会消除;

使用 all (跟不使用结果是一样的):

from 子句

from 子句表示 select 部分从中“取得”数据的数据源——其实就是表。

通常, 其后面就是表名; 比如: from tab1,

但, 也可能是其他一些数据来源; from tab1, tab2 (连接表);

select 输出 (取出) 部分, 如果给定的是字段名, 则其必然是来源于这里的“数据源”的字段;

```
行进行判断”, 其作用, 几乎就跟各种语言中 if 语句的
2, 则, 可见, where 子句依赖于 from 子句;
3, where 子句中, 通常都需要使用各种“运算符”;
  算术运算符: + - * / %
  比较运算符: > >= < <= = (等于)
  逻辑运算符: and or not

mysql> select * from tab_load_data where not (sex='女');
+----+-----+-----+-----+-----+
| id | name | sex | jiguan | f5 |
+----+-----+-----+-----+-----+
| 1 | 罗弟华 | 男 | 江西 | 1 |
| 2 | 韩顺平 | 男 | 四川 | 2 |
| 3 | 吴苗雷 | 男 | 黑龙江 | 1 |
+----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> select * from tab_load_data where not (sex='女');
```

is 运算符：空值和布尔值的判断

有 4 种情况的使用：

- XX is null：判断某个字段是“null”值——就是没有值；
- XX is not null：判断某个字段不是“null”值
- XX is true：判断某个字段为“真”（true）
- XX is false：判断某个字段为“假”（false）：0， 0.0， “”， null

所谓布尔值，其实是 tinyint(1)这个类型的一个“别名”，本质上，只是判断一个数字是否为 0；

in 运算符：给定确定数据的范围判断

语法：

XX in (值 1， 值 2， 值 3，);

含义：

表示字段 XX 的值为所列出的这些值中的一个，就算是满足了条件；这些值，通常是零散无规律的。

```
mysql> select * from tab_load_data where id in (1,3, 4);
```

id	name	sex	jiguan	f5
1	罗弟华	男	江西	1
3	吴英雷	男	黑龙江	1
4	王玉虹	女	河北	3

它罗列出的数据，如果有一定的规律，则其实可以使用逻辑运算符或 between 运算符来代替。

