

mysql day02 课堂笔记

1、把查询结果去除重复记录【distinct】

注意：原表数据不会被修改，只是查询结果去重。

去重需要使用一个关键字：distinct

```
mysql> select distinct job from emp;
```

job
CLERK
SALESMAN
MANAGER
ANALYST
PRESIDENT

// 这样编写是错误的，语法错误。

// distinct 只能出现在所有字段的最前方。

```
mysql> select ename,distinct job from emp;
```

// distinct 出现在 job,deptno 两个字段之前，表示两个字段联合起来去重。

```
mysql> select distinct job,deptno from emp;
```

job	deptno
CLERK	20
SALESMAN	30
MANAGER	20
MANAGER	30
MANAGER	10
ANALYST	20
PRESIDENT	10
CLERK	30
CLERK	10

统计一下工作岗位的数量？

```
select count(distinct job) from emp;
```

count(distinct job)
5

2、连接查询

2.1、什么是连接查询？

从一张表中单独查询，称为单表查询。

emp 表和 dept 表联合起来查询数据，从 emp 表中取员工名字，从 dept 表中取部门名字。
这种跨表查询，多张表联合起来查询数据，被称为连接查询。

2.2、连接查询的分类？

根据语法的年代分类：

SQL92：1992 年的时候出现的语法

SQL99：1999 年的时候出现的语法

我们这里重点学习 SQL99. (这个过程中简单演示一个 SQL92 的例子)

根据表连接的方式分类：

内连接：

等值连接

非等值连接

自连接

外连接：

左外连接（左连接）

右外连接（右连接）

全连接（不讲）

2.3、当两张表进行连接查询时，没有任何条件的限制会发生什么现象？

案例：查询每个员工所在部门名称？

```
mysql> select ename,deptno from emp;
```

ename	deptno
SMITH	20
ALLEN	30
WARD	30
JONES	20
MARTIN	30
BLAKE	30
CLARK	10
SCOTT	20
KING	10
TURNER	30
ADAMS	20
JAMES	30
FORD	20
MILLER	10

```
mysql> select * from dept;
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

两张表连接没有任何条件限制：

```
select ename,dname from emp, dept;
```

ename	dname
SMITH	ACCOUNTING
SMITH	RESEARCH
SMITH	SALES
SMITH	OPERATIONS
ALLEN	ACCOUNTING
ALLEN	RESEARCH
ALLEN	SALES
ALLEN	OPERATIONS

...

56 rows in set (0.00 sec)

14 * 4 = 56

当两张表进行连接查询，没有任何条件限制的时候，最终查询结果条数，是两张表条数的乘积，这种现象被称为：笛卡尔积现象。（笛卡尔发现的，这是一个数学现象。）

2.4、怎么避免笛卡尔积现象？

连接时加条件，满足这个条件的记录被筛选出来！

```
select
    ename,dname
from
    emp, dept
where
    emp.deptno = dept.deptno;
```

```
select
    emp.ename,dept.dname
from
    emp, dept
where
    emp.deptno = dept.deptno;
```

// 表起别名。很重要。效率问题。

```

select
    e.ename, d.dname
from
    emp e, dept d
where
    e.deptno = d.deptno; //SQL92 语法。

```

ename	dname
CLARK	ACCOUNTING
KING	ACCOUNTING
MILLER	ACCOUNTING
SMITH	RESEARCH
JONES	RESEARCH
SCOTT	RESEARCH
ADAMS	RESEARCH
FORD	RESEARCH
ALLEN	SALES
WARD	SALES
MARTIN	SALES
BLAKE	SALES
TURNER	SALES
JAMES	SALES

思考：最终查询的结果条数是 14 条，但是匹配的过程中，匹配的次数减少了吗？还是 56 次，只不过进行了四选一。次数没有减少。

注意：通过笛卡尔积现象得出，表的连接次数越多效率越低，尽量避免表的连接次数。

2.5、内连接之等值连接。

案例：查询每个员工所在部门名称，显示员工名和部门名？
emp e 和 dept d 表进行连接。条件是：e.deptno = d.deptno

SQL92 语法：

```

select
    e.ename, d.dname
from
    emp e, dept d
where
    e.deptno = d.deptno;

```

sql92 的缺点：结构不清晰，表的连接条件，和后期进一步筛选的条件，都放到了 where 后面。

SQL99 语法:

```
select
    e.ename, d.dname
from
    emp e
join
    dept d
on
    e.deptno = d.deptno;
```

//inner 可以省略 (带着 inner 可读性更好!!! 一眼就能看出来是内连接)

```
select
    e.ename, d.dname
from
    emp e
inner join
    dept d
on
    e.deptno = d.deptno; // 条件是等量关系, 所以被称为等值连接。
```

sql99 优点: 表连接的条件是独立的, 连接之后, 如果还需要进一步筛选, 再往后继续添加 where

SQL99 语法:

```
select
    ...
from
    a
join
    b
on
    a 和 b 的连接条件
where
    筛选条件
```

2.6、内连接之非等值连接

案例: 找出每个员工的薪资等级, 要求显示员工名、薪资、薪资等级?

mysql> select * from emp; e

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	1980-12-17	800.00	NULL	20
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30

7521	WARD	SALESMAN	7698	1981-02-22	1250.00	500.00	30
7566	JONES	MANAGER	7839	1981-04-02	2975.00	NULL	20

....

```
mysql> select * from salgrade; s
```

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

```
select
    e.ename, e.sal, s.grade
from
    emp e
join
    salgrade s
on
    e.sal between s.losal and s.hisal; // 条件不是一个等量关系，称为非等值连接。
```

```
select
    e.ename, e.sal, s.grade
from
    emp e
inner join
    salgrade s
on
    e.sal between s.losal and s.hisal;
```

ename	sal	grade
SMITH	800.00	1
ALLEN	1600.00	3
WARD	1250.00	2
JONES	2975.00	4
MARTIN	1250.00	2
BLAKE	2850.00	4
CLARK	2450.00	4
SCOTT	3000.00	4
KING	5000.00	5

TURNER	1500.00	3
ADAMS	1100.00	1
JAMES	950.00	1
FORD	3000.00	4
MILLER	1300.00	2

2.7、内连接之自连接

案例：查询员工的上级领导，要求显示员工名和对应的领导名？

```
mysql> select empno,ename,mgr from emp;
```

empno	ename	mgr
7369	SMITH	7902
7499	ALLEN	7698
7521	WARD	7698
7566	JONES	7839
7654	MARTIN	7698
7698	BLAKE	7839
7782	CLARK	7839
7788	SCOTT	7566
7839	KING	NULL
7844	TURNER	7698
7876	ADAMS	7788
7900	JAMES	7698
7902	FORD	7566
7934	MILLER	7782

技巧：一张表看成两张表。

emp a 员工表

empno	ename	mgr
7369	SMITH	7902
7499	ALLEN	7698
7521	WARD	7698
7566	JONES	7839
7654	MARTIN	7698
7698	BLAKE	7839
7782	CLARK	7839
7788	SCOTT	7566
7839	KING	NULL
7844	TURNER	7698
7876	ADAMS	7788
7900	JAMES	7698

7902	FORD	7566
7934	MILLER	7782

emp b 领导表

empno	ename	mgr
7369	SMITH	7902
7499	ALLEN	7698
7521	WARD	7698
7566	JONES	7839
7654	MARTIN	7698
7698	BLAKE	7839
7782	CLARK	7839
7788	SCOTT	7566
7839	KING	NULL
7844	TURNER	7698
7876	ADAMS	7788
7900	JAMES	7698
7902	FORD	7566
7934	MILLER	7782

```

select
    a.ename as '员工名', b.ename as '领导名'
from
    emp a
join
    emp b
on
    a.mgr = b.empno; //员工的领导编号 = 领导的员工编号

```

员工名	领导名
SMITH	FORD
ALLEN	BLAKE
WARD	BLAKE
JONES	KING
MARTIN	BLAKE
BLAKE	KING
CLARK	KING
SCOTT	JONES
TURNER	BLAKE
ADAMS	SCOTT
JAMES	BLAKE

FORD	JONES
MILLER	CLARK

+-----+-----+

13 条记录，没有 KING。《内连接》

以上就是内连接中的：自连接，技巧：一张表看做两张表。

2.8、外连接

```
mysql> select * from emp; e
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	1980-12-17	800.00	NULL	20
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30
7521	WARD	SALESMAN	7698	1981-02-22	1250.00	500.00	30
7566	JONES	MANAGER	7839	1981-04-02	2975.00	NULL	20
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30
7698	BLAKE	MANAGER	7839	1981-05-01	2850.00	NULL	30
7782	CLARK	MANAGER	7839	1981-06-09	2450.00	NULL	10
7788	SCOTT	ANALYST	7566	1987-04-19	3000.00	NULL	20
7839	KING	PRESIDENT	NULL	1981-11-17	5000.00	NULL	10
7844	TURNER	SALESMAN	7698	1981-09-08	1500.00	0.00	30
7876	ADAMS	CLERK	7788	1987-05-23	1100.00	NULL	20
7900	JAMES	CLERK	7698	1981-12-03	950.00	NULL	30
7902	FORD	ANALYST	7566	1981-12-03	3000.00	NULL	20
7934	MILLER	CLERK	7782	1982-01-23	1300.00	NULL	10

```
mysql> select * from dept; d
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

内连接：（A 和 B 连接，AB 两张表没有主次关系。平等的。）

```
select
    e.ename, d.dname
from
    emp e
join
    dept d
on
```

e.deptno = d.deptno; //内连接的特点：完成能够匹配上这个条件的数据查询出来。

ename	dname
CLARK	ACCOUNTING
KING	ACCOUNTING
MILLER	ACCOUNTING
SMITH	RESEARCH
JONES	RESEARCH
SCOTT	RESEARCH
ADAMS	RESEARCH
FORD	RESEARCH
ALLEN	SALES
WARD	SALES
MARTIN	SALES
BLAKE	SALES
TURNER	SALES
JAMES	SALES

外连接（右外连接）：

```
select
    e.ename,d.dname
from
    emp e
right join
    dept d
on
    e.deptno = d.deptno;
```

// outer 是可以省略的，带着可读性强。

```
select
    e.ename,d.dname
from
    emp e
right outer join
    dept d
on
    e.deptno = d.deptno;
```

right 代表什么：表示将 join 关键字右边的这张表看成主表，主要是为了将这张表的数据全部查询出来，捎带着关联查询左边的表。

在外连接当中，两张表连接，产生了主次关系。

外连接（左外连接）：

```
select
```

```

        e.ename, d.dname
from
        dept d
left join
        emp e
on
        e.deptno = d.deptno;

// outer 是可以省略的，带着可读性强。
select
        e.ename, d.dname
from
        dept d
left outer join
        emp e
on
        e.deptno = d.deptno;

```

带有 right 的是右外连接，又叫做右连接。
 带有 left 的是左外连接，又叫做左连接。
 任何一个右连接都有左连接的写法。
 任何一个左连接都有右连接的写法。

ename	dname
CLARK	ACCOUNTING
KING	ACCOUNTING
MILLER	ACCOUNTING
SMITH	RESEARCH
JONES	RESEARCH
SCOTT	RESEARCH
ADAMS	RESEARCH
FORD	RESEARCH
ALLEN	SALES
WARD	SALES
MARTIN	SALES
BLAKE	SALES
TURNER	SALES
JAMES	SALES
NULL	OPERATIONS

思考：外连接的查询结果条数一定是 \geq 内连接的查询结果条数？
 正确。

案例：查询每个员工的上级领导，要求显示所有员工的名字和领导名？

```
select
    a.ename as '员工名', b.ename as '领导名'
from
    emp a
left join
    emp b
on
    a.mgr = b.empno;
```

员工名		领导名
SMITH	FORD	
ALLEN	BLAKE	
WARD	BLAKE	
JONES	KING	
MARTIN	BLAKE	
BLAKE	KING	
CLARK	KING	
SCOTT	JONES	
KING	NULL	
TURNER	BLAKE	
ADAMS	SCOTT	
JAMES	BLAKE	
FORD	JONES	
MILLER	CLARK	

2.9、三张表，四张表怎么连接？

语法：

```
select
    ...
from
    a
join
    b
on
    a 和 b 的连接条件
join
    c
on
    a 和 c 的连接条件
right join
    d
on
```

a 和 d 的连接条件

一条 SQL 中内连接和外连接可以混合。都可以出现！

案例：找出每个员工的部门名称以及工资等级，
要求显示员工名、部门名、薪资、薪资等级？

```
select
    e.ename, e.sal, d.dname, s.grade
from
    emp e
join
    dept d
on
    e.deptno = d.deptno
join
    salgrade s
on
    e.sal between s.losal and s.hisal;
```

ename	sal	dname	grade
SMITH	800.00	RESEARCH	1
ALLEN	1600.00	SALES	3
WARD	1250.00	SALES	2
JONES	2975.00	RESEARCH	4
MARTIN	1250.00	SALES	2
BLAKE	2850.00	SALES	4
CLARK	2450.00	ACCOUNTING	4
SCOTT	3000.00	RESEARCH	4
KING	5000.00	ACCOUNTING	5
TURNER	1500.00	SALES	3
ADAMS	1100.00	RESEARCH	1
JAMES	950.00	SALES	1
FORD	3000.00	RESEARCH	4
MILLER	1300.00	ACCOUNTING	2

案例：找出每个员工的部门名称以及工资等级，还有上级领导，
要求显示员工名、领导名、部门名、薪资、薪资等级？

```
select
    e.ename, e.sal, d.dname, s.grade, l.ename
from
    emp e
join
```

```

dept d
on
  e.deptno = d.deptno
join
  salgrade s
on
  e.sal between s.losal and s.hisal
left join
  emp l
on
  e.mgr = l.empno;

```

ename	sal	dname	grade	ename
SMITH	800.00	RESEARCH	1	FORD
ALLEN	1600.00	SALES	3	BLAKE
WARD	1250.00	SALES	2	BLAKE
JONES	2975.00	RESEARCH	4	KING
MARTIN	1250.00	SALES	2	BLAKE
BLAKE	2850.00	SALES	4	KING
CLARK	2450.00	ACCOUNTING	4	KING
SCOTT	3000.00	RESEARCH	4	JONES
KING	5000.00	ACCOUNTING	5	NULL
TURNER	1500.00	SALES	3	BLAKE
ADAMS	1100.00	RESEARCH	1	SCOTT
JAMES	950.00	SALES	1	BLAKE
FORD	3000.00	RESEARCH	4	JONES
MILLER	1300.00	ACCOUNTING	2	CLARK

3、子查询？

3.1、什么是子查询？

select 语句中嵌套 select 语句，被嵌套的 select 语句称为子查询。

3.2、子查询都可以出现在哪里呢？

```

select
  ..(select).
from
  ..(select).
where
  ..(select).

```

3.3、where 子句中的子查询

案例：找出比最低工资高的员工姓名和工资？

```

select
    ename, sal
from
    emp
where
    sal > min(sal);

```

ERROR 1111 (HY000): Invalid use of group function
 where 子句中不能直接使用分组函数。

实现思路：

第一步：查询最低工资是多少

```

select min(sal) from emp;
+-----+
| min(sal) |
+-----+
|   800.00 |
+-----+

```

第二步：找出>800 的

```

select ename, sal from emp where sal > 800;

```

第三步：合并

```

select ename, sal from emp where sal > (select min(sal) from emp);
+-----+-----+
| ename  | sal    |
+-----+-----+
| ALLEN  | 1600.00 |
| WARD   | 1250.00 |
| JONES  | 2975.00 |
| MARTIN | 1250.00 |
| BLAKE  | 2850.00 |
| CLARK  | 2450.00 |
| SCOTT  | 3000.00 |
| KING   | 5000.00 |
| TURNER | 1500.00 |
| ADAMS  | 1100.00 |
| JAMES  | 950.00  |
| FORD   | 3000.00 |
| MILLER | 1300.00 |
+-----+-----+

```

3.4、from 子句中的子查询

注意：from 后面的子查询，可以将子查询的查询结果当做一张临时表。（技巧）

案例：找出每个岗位的平均工资的薪资等级。

第一步：找出每个岗位的平均工资（按照岗位分组求平均值）

```
select job,avg(sal) from emp group by job;
```

job	avgsal
ANALYST	3000.000000
CLERK	1037.500000
MANAGER	2758.333333
PRESIDENT	5000.000000
SALESMAN	1400.000000

t 表

第二步：克服心理障碍，把以上的查询结果就当做一张真实存在的表 t。

```
mysql> select * from salgrade; s 表
```

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

t 表和 s 表进行表连接，条件： t 表 avg(sal) between s.losal and s.hisal;

```
select
    t.*, s.grade
from
    (select job,avg(sal) as avgsal from emp group by job) t
join
    salgrade s
on
    t.avgsal between s.losal and s.hisal;
```

job	avgsal	grade
CLERK	1037.500000	1
SALESMAN	1400.000000	2
ANALYST	3000.000000	4
MANAGER	2758.333333	4
PRESIDENT	5000.000000	5

3.5、select 后面出现的子查询（这个内容不需要掌握，了解即可!!!）

案例：找出每个员工的部门名称，要求显示员工名，部门名？

```
select
```



```

        e.ename,e.deptno,(select d.dname from dept d where e.deptno = d.deptno) as
dname
from
    emp e;

```

ename	deptno	dname
SMITH	20	RESEARCH
ALLEN	30	SALES
WARD	30	SALES
JONES	20	RESEARCH
MARTIN	30	SALES
BLAKE	30	SALES
CLARK	10	ACCOUNTING
SCOTT	20	RESEARCH
KING	10	ACCOUNTING
TURNER	30	SALES
ADAMS	20	RESEARCH
JAMES	30	SALES
FORD	20	RESEARCH
MILLER	10	ACCOUNTING

```

//错误: ERROR 1242 (21000): Subquery returns more than 1 row
select
    e.ename,e.deptno,(select dname from dept) as dname
from
    emp e;

```

注意: 对于 select 后面的子查询来说, 这个子查询只能一次返回 1 条结果, 多于 1 条, 就报错了。!

4、union 合并查询结果集

案例: 查询工作岗位是 MANAGER 和 SALESMAN 的员工?

```

select ename,job from emp where job = 'MANAGER' or job = 'SALESMAN';
select ename,job from emp where job in('MANAGER','SALESMAN');

```

ename	job
ALLEN	SALESMAN
WARD	SALESMAN
JONES	MANAGER
MARTIN	SALESMAN
BLAKE	MANAGER

CLARK	MANAGER
TURNER	SALESMAN

```
select ename,job from emp where job = 'MANAGER'
union
select ename,job from emp where job = 'SALESMAN';
```

ename	job
JONES	MANAGER
BLAKE	MANAGER
CLARK	MANAGER
ALLEN	SALESMAN
WARD	SALESMAN
MARTIN	SALESMAN
TURNER	SALESMAN

union 的效率要高一些。对于表连接来说，每连接一次新表，则匹配的次数满足笛卡尔积，成倍的翻。。。但是 union 可以减少匹配的次数。在减少匹配次数的情况下，还可以完成两个结果集的拼接。

a 连接 b 连接 c
a 10 条记录
b 10 条记录
c 10 条记录
匹配次数是：1000

a 连接 b 一个结果：10 * 10 --> 100 次
a 连接 c 一个结果：10 * 10 --> 100 次
使用 union 的话是：100 次 + 100 次 = 200 次。（union 把乘法变成了加法运算）

union 在使用的时候有注意事项吗？

```
//错误的：union 在进行结果集合并的时候，要求两个结果集的列数相同。
select ename,job from emp where job = 'MANAGER'
union
select ename from emp where job = 'SALESMAN';
```

// MYSQL 可以，oracle 语法严格，不可以，报错。要求：结果集合并时列和列的数据类型也要一致。

```
select ename,job from emp where job = 'MANAGER'
union
select ename,sal from emp where job = 'SALESMAN';
```

ename	job
JONES	MANAGER
BLAKE	MANAGER
CLARK	MANAGER
ALLEN	1600
WARD	1250
MARTIN	1250
TURNER	1500

5、limit（非常重要）

5.1、limit 作用：将查询结果集的一部分取出来。通常使用在分页查询当中。

百度默认：一页显示 10 条记录。

分页的作用是为了提高用户的体验，因为一次全部都查出来，用户体验差。

可以一页一页翻页看。

5.2、limit 怎么用呢？

完整用法：limit startIndex, length

startIndex 是起始下标，length 是长度。

起始下标从 0 开始。

缺省用法：limit 5；这是取前 5。

按照薪资降序，取出排名在前 5 名的员工？

```
select
    ename, sal
from
    emp
order by
    sal desc
limit 5; //取前 5
```

```
select
    ename, sal
from
    emp
order by
    sal desc
limit 0,5;
```

ename	sal
-------	-----

	KING		5000.00	
	SCOTT		3000.00	
	FORD		3000.00	
	JONES		2975.00	
	BLAKE		2850.00	
+-----+-----+				

5.3、注意：mysql 当中 limit 在 order by 之后执行!!!!!!

5.4、取出工资排名在[3-5]名的员工？

```
select
    ename, sal
from
    emp
order by
    sal desc
limit
    2, 3;
```

2 表示起始位置从下标 2 开始，就是第三条记录。

3 表示长度。

+-----+-----+				
	ename		sal	
+-----+-----+				
	FORD		3000.00	
	JONES		2975.00	
	BLAKE		2850.00	
+-----+-----+				

5.5、取出工资排名在[5-9]名的员工？

```
select
    ename, sal
from
    emp
order by
    sal desc
limit
    4, 5;
```

+-----+-----+				
	ename		sal	
+-----+-----+				
	BLAKE		2850.00	
	CLARK		2450.00	
	ALLEN		1600.00	
	TURNER		1500.00	

```
| MILLER | 1300.00 |  
+-----+-----+
```

5.6、分页

每页显示 3 条记录

```
第 1 页: limit 0,3      [0 1 2]  
第 2 页: limit 3,3      [3 4 5]  
第 3 页: limit 6,3      [6 7 8]  
第 4 页: limit 9,3      [9 10 11]
```

每页显示 pageSize 条记录

第 pageNo 页: limit (pageNo - 1) * pageSize , pageSize

```
public static void main(String[] args){  
    // 用户提交过来一个页码, 以及每页显示的记录条数  
    int pageNo = 5; //第 5 页  
    int pageSize = 10; //每页显示 10 条  
  
    int startIndex = (pageNo - 1) * pageSize;  
    String sql = "select ...limit " + startIndex + ", " + pageSize;  
}
```

记公式:

limit (pageNo-1)*pageSize , pageSize

6、关于 DQL 语句的大总结:

```
select  
    ...  
from  
    ...  
where  
    ...  
group by  
    ...  
having  
    ...  
order by  
    ...  
limit  
    ...
```

执行顺序?

1. from
2. where
3. group by
4. having

```
5.select
6.order by
7.limit..
```

7、表的创建（建表）

7.1、建表的语法格式：（建表属于 DDL 语句，DDL 包括：create drop alter）

```
create table 表名(字段名1 数据类型, 字段名2 数据类型, 字段名3 数据类型);
```

```
create table 表名(
    字段名1 数据类型,
    字段名2 数据类型,
    字段名3 数据类型
);
```

表名：建议以 t_ 或者 tbl_ 开始，可读性强。见名知意。

字段名：见名知意。

表名和字段名都属于标识符。

7.2、关于 mysql 中的数据类型？

很多数据类型，我们只需要掌握一些常见的数据类型即可。

```
varchar(最长 255)
    可变长度的字符串
    比较智能，节省空间。
    会根据实际的数据长度动态分配空间。
```

优点：节省空间

缺点：需要动态分配空间，速度慢。

```
char(最长 255)
    定长字符串
    不管实际的数据长度是多少。
    分配固定长度的空间去存储数据。
    使用不恰当的时候，可能会导致空间的浪费。
```

优点：不需要动态分配空间，速度快。

缺点：使用不当可能会导致空间的浪费。

varchar 和 char 我们应该怎么选择？

性别字段你选什么？因为性别是固定长度的字符串，所以选择 char。

姓名字段你选什么？每一个人的名字长度不同，所以选择 varchar。

```
int(最长 11)
    数字中的整数型。等同于 java 的 int。
```

bigint
数字中的长整型。等同于 java 中的 long。

float
单精度浮点型数据

double
双精度浮点型数据

date
短日期类型

datetime
长日期类型

clob
字符大对象
最多可以存储 4G 的字符串。
比如：存储一篇文章，存储一个说明。
超过 255 个字符的都要采用 CLOB 字符大对象来存储。
Character Large Object:CLOB

blob
二进制大对象
Binary Large Object
专门用来存储图片、声音、视频等流媒体数据。
往 BLOB 类型的字段上插入数据的时候，例如插入一个图片、视频等，
你需要使用 IO 流才行。

t_movie 电影表（专门存储电影信息的）

编号	名字	故事情节	上映日期
时长	海报	类型	
no(bigint)	name(varchar)	history(clob)	playtime(date)
time(double)	image(blob)	type(char)	

10000	哪吒	2019-10-11
2.5	'1'	
10001	林正英之娘娘	2019-11-11
1.5	'2'	
....			

7.3、创建一个学生表？

学号、姓名、年龄、性别、邮箱地址

```
create table t_student(
    no int,
    name varchar(32),
    sex char(1),
    age int(3),
    email varchar(255)
);
```

删除表:

```
drop table t_student; // 当这张表不存在的时候会报错!
```

```
// 如果这张表存在的话, 删除
```

```
drop table if exists t_student;
```

7.4、插入数据 insert (DML)

语法格式:

```
insert into 表名(字段名 1, 字段名 2, 字段名 3...) values(值 1, 值 2, 值 3);
```

注意: 字段名和值要一一对应。什么是一一对应?

数量要对应。数据类型要对应。

```
insert into t_student(no, name, sex, age, email)
values(1, 'zhangsan', 'm', 20, 'zhangsan@123.com');
```

```
insert into t_student(email, name, sex, age, no)
values('lisi@123.com', 'lisi', 'f', 20, 2);
```

```
insert into t_student(no) values(3);
```

no	name	sex	age	email
1	zhangsan	m	20	zhangsan@123.com
2	lisi	f	20	lisi@123.com
3	NULL	NULL	NULL	NULL

```
insert into t_student(name) values('wangwu');
```

no	name	sex	age	email
1	zhangsan	m	20	zhangsan@123.com
2	lisi	f	20	lisi@123.com
3	NULL	NULL	NULL	NULL
NULL	wangwu	NULL	NULL	NULL

注意: insert 语句但凡是执行成功了, 那么必然会多一条记录。
没有给其它字段指定值的话, 默认值是 NULL。


```
drop table if exists t_student;
create table t_student(
    no int,
    name varchar(32),
    sex char(1) default 'm',
    age int(3),
    email varchar(255)
);
```

Field	Type	Null	Key	Default	Extra
no	int(11)	YES		NULL	
name	varchar(32)	YES		NULL	
sex	char(1)	YES		m	
age	int(3)	YES		NULL	
email	varchar(255)	YES		NULL	

```
insert into t_student(no) values(1);
mysql> select * from t_student;
```

no	name	sex	age	email
1	NULL	m	NULL	NULL

insert 语句中的“字段名”可以省略吗？可以
 insert into t_student values(2); //错误的

// 注意：前面的字段名省略的话，等于都写上了！所以值也要都写上！
 insert into t_student values(2, 'lisi', 'f', 20, 'lisi@123.com');

no	name	sex	age	email
1	NULL	m	NULL	NULL
2	lisi	f	20	lisi@123.com

7.5、insert 插入日期

数字格式化：format

```
select ename,sal from emp;
```

ename	sal
SMITH	800.00

ALLEN	1600.00	
WARD	1250.00	
JONES	2975.00	
MARTIN	1250.00	
BLAKE	2850.00	
CLARK	2450.00	
SCOTT	3000.00	
KING	5000.00	
TURNER	1500.00	
ADAMS	1100.00	
JAMES	950.00	
FORD	3000.00	
MILLER	1300.00	
+-----+-----+		

格式化数字: format(数字, '格式')

```
select ename,format(sal, '$999,999') as sal from emp;
```

ename	sal	
+-----+-----+		
SMITH	800	
ALLEN	1,600	
WARD	1,250	
JONES	2,975	
MARTIN	1,250	
BLAKE	2,850	
CLARK	2,450	
SCOTT	3,000	
KING	5,000	
TURNER	1,500	
ADAMS	1,100	
JAMES	950	
FORD	3,000	
MILLER	1,300	
+-----+-----+		

str_to_date: 将字符串 varchar 类型转换成 date 类型

date_format: 将 date 类型转换成具有一定格式的 varchar 字符串类型。

```
drop table if exists t_user;
create table t_user(
    id int,
    name varchar(32),
    birth date // 生日也可以使用 date 日期类型
);
```

```
create table t_user(
```

```

    id int,
    name varchar(32),
    birth char(10) // 生日可以使用字符串，没问题。
);

```

生日：1990-10-11 （10 个字符）

注意：数据库中的有一条命名规范：

所有的标识符都是全部小写，单词和单词之间使用下划线进行衔接。

```
mysql> desc t_user;
```

Field	Type	Null	Key	Default	Extra
id	int(11)	YES		NULL	
name	varchar(32)	YES		NULL	
birth	date	YES		NULL	

插入数据？

```
insert into t_user(id,name,birth) values(1, 'zhangsan', '01-10-1990'); //
```

1990 年 10 月 1 日

出问题了：原因是类型不匹配。数据库 birth 是 date 类型，这里给了一个字符串 varchar。

怎么办？可以使用 str_to_date 函数进行类型转换。

str_to_date 函数可以将字符串转换成日期类型 date？

语法格式：

```
str_to_date('字符串日期', '日期格式')
```

mysql 的日期格式：

```

%Y  年
%m  月
%d  日
%h  时
%i  分
%s  秒

```

```
insert into t_user(id,name,birth) values(1, 'zhangsan',
str_to_date('01-10-1990', '%d-%m-%Y'));
```

str_to_date 函数可以把字符串 varchar 转换成日期 date 类型数据，通常使用在插入 insert 方面，因为插入的时候需要一个日期类型的数据，需要通过该函数将字符串转换成 date。

好消息？

如果你提供的日期字符串是这个格式，str_to_date 函数就不需要了!!!

```

        %Y-%m-%d
insert into t_user(id,name,birth) values(2, 'lisi', '1990-10-01');

```

查询的时候可以以某个特定的日期格式展示吗？

date_format

这个函数可以将日期类型转换成特定格式的字符串。

```
select id,name,date_format(birth, '%m/%d/%Y') as birth from t_user;
```

id	name	birth
1	zhangsan	10/01/1990
2	lisi	10/01/1990

date_format 函数怎么用？

date_format(日期类型数据, '日期格式')

这个函数通常使用在查询日期方面。设置展示的日期格式。

```
mysql> select id,name,birth from t_user;
```

id	name	birth
1	zhangsan	1990-10-01
2	lisi	1990-10-01

以上的 SQL 语句实际上是进行了默认的日期格式化，

自动将数据库中的 date 类型转换成 varchar 类型。

并且采用的格式是 mysql 默认的日期格式：'%Y-%m-%d'

```
select id,name,date_format(birth, '%Y/%m/%d') as birth from t_user;
```

java 中的日期格式？

yyyy-MM-dd HH:mm:ss SSS

7.6、date 和 datetime 两个类型的区别？

date 是短日期：只包括年月日信息。

datetime 是长日期：包括年月日时分秒信息。

```
drop table if exists t_user;
```

```
create table t_user(
```

```
    id int,
```

```
    name varchar(32),
```

```
    birth date,
```

```
    create_time datetime
```

```
);
```

id 是整数
name 是字符串
birth 是短日期
create_time 是这条记录的创建时间：长日期类型

mysql 短日期默认格式：%Y-%m-%d
mysql 长日期默认格式：%Y-%m-%d %h:%i:%s

```
insert into t_user(id, name, birth, create_time)
values(1, 'zhangsan', '1990-10-01', '2020-03-18 15:49:50');
```

在 mysql 当中怎么获取系统当前时间？
now() 函数，并且获取的时间带有：时分秒信息!!!! 是 datetime 类型的。

```
insert into t_user(id, name, birth, create_time)
values(2, 'lisi', '1991-10-01', now());
```

7.7、修改 update (DML)

语法格式：

update 表名 set 字段名 1=值 1, 字段名 2=值 2, 字段名 3=值 3... where 条件;

注意：没有条件限制会导致所有数据全部更新。

```
update t_user set name = 'jack', birth = '2000-10-11' where id = 2;
```

+-----+-----+-----+-----+			
id	name	birth	create_time
+-----+-----+-----+-----+			
1	zhangsan	1990-10-01	2020-03-18 15:49:50
2	jack	2000-10-11	2020-03-18 15:51:23
+-----+-----+-----+-----+			

```
update t_user set name = 'jack', birth = '2000-10-11', create_time = now() where
id = 2;
```

更新所有？

```
update t_user set name = 'abc';
```

7.8、删除数据 delete (DML)

语法格式？

```
delete from 表名 where 条件;
```

注意：没有条件，整张表的数据会全部删除！

```
delete from t_user where id = 2;
```

```
insert into t_user(id) values(2);
```

```
delete from t_user; // 删除所有!
```