1. MySQL中的**列约束**

MySQL在插入数据时，进行特定的验证；只有满足条件才允许插入，否则被认为是非法插入。 共6种

CREATE TABLE xx( nid 列类型 列约束 )

1. **主键约束**—— PRIMARY KEY

声明了主键约束列中，不允许出现重复值，且一个表仅能存在一个主键，通常加载编号列（添加了主键的数字列在查询记录时，会按照编号从小到大排序，加快查找速度，字符串类则不会排序），主键约束列不能加入NULL值。

NULL 表示空，在插入数据时，无法确定要保存的值。NULL和任何值比较都不等，包括两个NULL比较。 ※不能加引号

1. **非空约束**—— NOT NULL

指不允许约束列出现NULL值，出现则报错

1. **唯一约束**—— UNIQUE

声明了唯一约束列不允许插入重复值，但允许同时插入多个NULL（因为 NULL ≠ NULL）

1. **默认值约束**—— DEFAULT +默认值

可以使用关键字default声明默认值，2种使用方式。

INSERT INTO xx VALUES(1,DEFAULT 3 ,...)

INSERT INTO xx（A,B） VALUES (a,b); 除去插入的特定值a、b其余全为默认值

1. **检查约束**—— CHECK(数据库端基本不用，都是网页JS便可以实现)

可以对插入的数据进行自定义验证

score INT CHECK(score>=0 AND score<=100)

※MySQL不支持检查约束，会降低数据插入速度

1. **外键约束**—— FOREIGN KEY

声明了外键约束的列上，取值必须在另一个表的主键列上出现过，两者的列类型保持一致，允许A列存在NULL

外键约束在create table （）内要另起一行

FOREIGN KEY(A列) REFERENCES B表(主键列)；

#设置此表中A 列为外键约束列，关联B表中主键列

且A列与B表主键列 列类型（INT SMALLINT等）要保持一致

1. 自增列

AUTO\_INCREMENT 自动增长，假如一个列上声明了自增列，无需手动赋值，直接赋值为NULL，会获取当前的最大值，然后加1插入。

只能在主键使用，且主键要为整型列。（null会自动在你上一条最大数字上加1，不会识别后面更大数字）

1. 简单查询
2. 查询特定的列.

例：查询所有员工姓名性别

SELECT ename,sex FROM emp;

1. 查询所有的列

SELECT \* FROM emp;

1. 给列起别名（在列名称后 AS 后接别名， 为空格，甚至AS\_\_可省略）

例：查询所有员工姓名和工资，使用汉字别名

SELECT ename AS 姓名,salary AS 工资 FROM emp;

SELECT ename 姓名,salary 工资 FROM emp;

1. 显示不同的记录/**合并相同的项**

在要查询列前加入DISTINCT\_\_，

SELECT DISTINCT deptId FROM emp;

1. 查询时执行计算

例：2+3-5\*4.2/8.8+13

SELECT 2+3-5\*4.2/8.8+13;

例：若每个员工的工资增加500，年终奖5000，查询所有员工姓名及其年薪，使用汉字别名

SELECT ename 姓名,(salary+500)\*12+5000 年薪 FROM emp;

1. 对查询结果排序（ORDER BY ASC/DESC）

例：查询所有部门，按部门编号升序排列

SELECT \* FROM dept ORDER BY did ASC/DESC;

#ascendant升序 #descendent 降序

|  |
| --- |
| ORDER BY 可以按数值、日期、字符串排序，默认是升序排列 |

SELECT \* FROM emp ORDER BY salary DESC,ename;（order只需要写一次，第一个识别词并列时按第二个排列）

1. 条件查询

例：查询编号大于等于5的员工

SELECT \* FROM emp WHERE eid>=5;

|  |
| --- |
| 比较运算符：> < >= <= = !=(不等于) |

例：查询出1991-1-1后出生的员工有哪些

SELECT \* FROM emp WHERE birthday > '1991-1-1';

例：查询有无明确部门的员工（仅此一种方法查NULL）

SELECT \* FROM emp WHERE deptId IS NULL/IS NOT NULL;

例：查询工资大于等于6000的男员工

SELECT \* FROM emp WHERE sex=1 AND salary>=6000;

例：工资在6000到8000的员工（>=6000 AND <=8000）

SELECT \* FROM emp WHERE salary>=6000 AND salary<=8000;

SELECT \* FROM emp WHERE salary BETWEEN 6000 AND 8000;

例：工资小于6000或8000以上的员工（<6000 OR >8000）

SELECT \* FROM emp WHERE salary<6000 OR salary>8000;

SELECT \* FROM emp WHERE salary NOT BETWEEN 6000 AND 8000;

例：查询出20和30号部门员工有哪些（反面加not）

SELECT \* FROM emp WHERE deptld IN(20,30);

|  |
| --- |
| WHERE  AND/OR  (NOT) BETWEEN AND  IS (NOT) NULL  (NOT) IN( , ) |

1. 模糊条件查询

例：查询出姓名中含有字母e的员工（任意位置）

SELECT \* FROM emp WHERE ename LIKE '%e%';

例：查询出姓名中倒数第二位为e的员工

SELECT \* FROM emp WHERE ename LIKE '%e\_';

|  |
| --- |
| % 可以匹配任意多个字符 >=0  \_ 可以匹配任意1个字符 =1  以上两个匹配符号必须使用LIKE关键字 |

1. 分页查询

加入查询的结果集有太多数据，一次显示不完，可以分页显示。

需要两个条件：当前的页码、每页的数据量

|  |
| --- |
| **开始的值**=（当前页码 - 1）\*每页的数据量 （※数据从0条开始） |

SELECT \* FROM emp LIMIT start,count;

start : 开始的值（上方方式计算）

count : 每页的数据量

select \* from emp limit 0,6;

综合：

例：查询出工资在8000以上的女员工的姓名、性别、生日、工资，结果集按照工资的降序排列，取前3条数据。

select ename,sex,birthday,salary from emp where sex=0 order by salary desc limit 0,3;

4.复杂查询： 用\*可以有效规避NULL值

|  |
| --- |
| 聚合函数 函数：一个功能体，提供若干数据，产出结果 |

（1）聚合查询 5个 （COUNT/SUM/AVG/MAX/MIN）

例：查询所有男员工数量

SELECT COUNT( \* ) FROM emp where sex = 1;

1. 分组查询 只能查询分组条件和聚合函数

例：查询男女员工的最高工资

SELECT sex,MAX(salary) FROM emp GROUP BY sex;

例：查询不同部门中年龄最大最小的人的日期

SELECT deptId,min(birthday),max(birthday) FROM emp GROUP by deptId;

1. 子查询（记得加括号）

把一个SQL语句的结果作为另一个SQL语句的查询条件

例：查询出研发部的员工

SELECT \* FROM emp WHERE deptId=(select did from dept where dname='研发部');

例：查询出工资高于Tom的男员工

select \* from emp where salary>(select salary from emp where ename='Tom') AND sex=1；

例：查询与TOM同一年出生的员工

select \* from emp where year(birthday)=(select year(birthday) from emp where ename='Tom');

1. 多表查询

例：查询出所有员工姓名及其部门名称

SELECT ename,dname FROM emp,dept WHERE deptId=did;

无法查询出没有部门的员工，也无法查询出没有员工的部门

1. 内连接 （a inner join b on条件）

SELECT ename,dname FROM emp INNER JOIN dept ON deptId=did;

与上方问题相同，只是换了一种写法

1. 左外连接 （左侧表所有数据都会显示）

SELECT ename,dname FROM emp LEFT (OUTER) JOIN dept ON deptId=did; （outer可省略，先写的表为左侧表）

1. 右外连接 （右侧表中所有记录会显示）

SELECT ename,dname FROM emp RIGHT (OUTER) JOIN dept ON deptId=did; （outer可省略，先写的表为左侧表）

1. 全连接 （显示左侧及右侧表中所有记录 FULL JOIN ON）

PS: MySQL不支持

解决方法，(左外连接) UNION (右外连接);

(select ename,dname from emp left join dept on deptId=did) union (SELECT ename,dname FROM emp RIGHT OUTER JOIN dept ON deptId=did);