创建表：

create table 表名(

列名1 数据类型 primary key,

列名2 数据类型,

……

);

删除表：

drop table 表名;

向表中插入列：

alter table 表名 add(列名 数据类型)；

alter table 表名 drop column 列名;

alter table 表名 drop (列1,列2);

插入语句：

insert into 表名 values(列值1,列值2,……);

insert into 表名(列名1,列名2,……) values (列值1,列值2,……);

修改语句：

update 表名 set 列名1=该列新值,列名2=该列新值,…… where 条件;

如果没有where条件，修改表中的所有的数据

删除语句：

delete from 表名 where 条件;

快速删除：

trun cate 表名;

优点：删除速度快，效率高

缺点：删除的数据无法恢复

查询语句：

select \* from 表名;

MySql查询、增加、修改json数据：

<https://blog.csdn.net/LuckFairyLuckBaby/article/details/89551374>

create table 'test\_user'{

'id' int primary key auto\_increment,

'name' varchar (50) not null,

'info' json

};

INSERT INTO test\_user(`name`, `info`) VALUES('xiaoming','{"sex": 1, "age": 18, "nick\_name": "小萌"}');

使用JSON\_OBJECT()函数构造json对象：

INSERT INTO test\_user(`name`, `info`) VALUES('xiaohua', JSON\_OBJECT("sex", 0, "age", 17));

使用json\_array()函数构造json数组：

INSERT INTO test\_user(`name`, `info`) VALUES('xiaozhang', JSON\_OBJECT("sex", 1, "age", 19, "tag", JSON\_ARRAY(3,5,90)));

查询：

select name, info->'$.nick\_name', info->'$.sex', info->'$.tag[0]' from test\_user;

select name, JSON\_EXTRACT(info, '$.nick\_name'), JSON\_EXTRACT(info, '$.sex'), JSON\_EXTRACT(info, '$.tag[0]') from test\_user;

可以用JSON\_UNQUOTE函数将双引号去掉：

select name, JSON\_UNQUOTE(info->'$.nick\_name') from test\_user where name='xiaoming';

也可以操作符->>：

select name, info->>'$.nick\_name' from test\_user where name='xiaoming';

select name, info->>'$.nick\_name' from test\_user where info->'$.nick\_name'='小萌';

修改json：

UPDATE test\_user SET info = JSON\_INSERT(info, '$.sex', 1, '$.nick\_name', '小花') where id=2;

使用伪劣查询：

查询前5位数据：

select \* from 表名 where rownum<=5;

取出最后3位的数据：

1）select rownum rn1,emp.\* from emp；

2）select \* from ((select rownum rn1,emp.\* from emp) order by rn1 desc) where rownum<=3;

别名：

1、as 2、空格 一般用空格

给列起别名：

select 列名1 别名1,列名2 别名2,…… from 表名;

给表起别名：

select 别名.列名1,别名.列名2,…… from 表名 别名;

简单条件查询：

>, >= , < , <= , = , !=(不等于) , <>(不等于)

null值查询：

查询列值为空的：

is null;

select 列名 from 表名 where 列名 is null;

查询列值不为空的：

select 列名 from 表名 where 列名 is not null;

模糊查询：like

%:代表零个或多个字符

\_（下划线）:代表任意的一个字符

select \* from 表名 where 列名 like'通配符';

逻辑运算符：

and(逻辑与) or(逻辑或) not(逻辑非)

between 初值 and 终值

select \* from 表名 where 列名 between 初值 and 终值 ;

查询某列中的某些值的信息：

in(list):list指某些值

select \* from 表名 where 列名 in(list);

not in(list):取出不符合的值

select \* from 表名 where 列名 not in(list);

去掉重复：distinct

select distinct 列名 from 表名;

null值的影响:

任何数字如果用“+”跟null相连接，整个结果也为null

nvl函数：

select 列名+nvl(列名,数值) from 表名;

如果为空返回的是数值，如果不为空返回的是该列的值。

聚合函数：

count(总数),sum(和),avg(平均值),max(最大值),min(最小值)

select 聚合函数（列名） from 表名;

排序：

select \* from 表名 where 条件 order by 列名 asc/desc;

asc:升序（默认） desc:降序

如有相同的：

select \* from 表名 where 条件 order by 列名1 asc/desc,列名2 asc/desc;

分组：

select 列名/聚合函数

from 表名

where 条件

group by 列名

order by 列名/聚合函数/别名;

步骤：先where->group by->查询返回的数据->order by

子句：对分组以后的数据再次进行过滤，一般跟聚合函数结合使用

select 列名/聚合函数

from 表名

where 条件

group by 列名

having 子句

order by 列名/聚合函数/别名;

步骤：先where->group by->having->查询返回的数据->order by

where和having的区别：

where是针对整张表中的所有数据

having是针对分组以后的每一组数据进行过滤，而且经常跟聚合函数结合使用

连接查询（关联查询）：所要查询的数据来源于多张表

内连接-等值连接:

格式：

select 别名1.\*/列名,别名2.\*/列名

from 表名1 别名1,表名2 别名2

where 关联条件

笛卡尔积：在表的连接查询的时候，如果没有加入关联条件，则两张表中每一条数据都会进行任意的排列组合。

内连接-非等值连接:

select 别名1.\*/列名,别名2.\*/列名

from 表1 别名1,表2 别名2

where 非等值关联条件

案例：查询emp表中员工的编号，姓名，工资以及所属工资等级，以及该等级下的最高工资

select e.empno,e.ename,e.sal,s.grade,

s.hisal

from emp e,salgrade s

where e.sal between s.losal and s.hisal

内连接-自连接：同一张表自己跟自己连接，必须保证表中两个字段之间有关联关系

eg：查询emp表中，每个员工的姓名，职位以及上级的编号，姓名

分析：把emp表当成2张表，一张表员工表，另外一张表是领导表，员工表中mgr（上级领导编号）等于领导表中empno(编号)

select e.ename,e.job,m.empno,m.ename

from emp e,emp m --e 员工表,m 领导表

where e.mgr = m.empno

外连接：内连接查询两张表中有关联关系的数据（没有关联关系的数据，就查不到了），外连接即查询有关联关系的数据，也查询没有关联关系的数据

Oracle中特有用于查询外连接专用语法，使用（+）来进行外连接查询

案例：查询dept表中所有的部门的编号，名称以及所属员工的所有信息

select d.deptno,d.dname,e.\*

from dept d,emp e

where d.deptno = e.depton(+)

有关联查询

在关联条件中，使用（+）来实现连接查询。在关联条件中，查询所有数据的表的条件对面加入（+）

Oracle中专用外连接查询，只能在条件一端加上（+），不能两端都加上（+）

SQL用于连接查询的语法（重点，可以任意匹配数据库）

内连接：用来连接两张表中有关联关系的数据

格式：inner join

select 别名1.\*/列名, 别名2.\*/列名

from 左表 别名1 inner join 右表 别名2

on 关联条件

左外连接：以左表为主，查询左表所有的数据，以及与之关联右表中的数据

格式：left join

select 别名1.\*/列名,别名2.\*/列名

from 左表 别名1 left (outer) join 右表 别名2

on 关联条件

右外连接：以右边的表为主，查询右表所有的数据，以及与之关联左表中的数据

格式：right join

select 别名1.\*/列名,别名2.\*/列名

from 左表 别名1 right (outer) join 右表 别名2

on 关联条件

全外连接查询：查询两张表中所有的数据（有关联关系的数据和没有关联关系的数据）

格式：full join

select 别名1.\*/列名,别名2.\*/列名

from 左表 别名1 full outer join 右表 别名2

on 关联条件

视图：

视图：可以把视图当成一个表去使用

视图：视图是一个虚拟表，从一个或者多个视图中查看表中的数据

1）通过视图查看表中的数据，视图中不保存任何的数据，仅仅是一个查看数据的窗口。删除一个视图对表没有任何的影响。

2）使用视图可以集中，简化，定制所需要查询的数据，而不需要再去表中查询

创建视图：

create view 视图名称

as

查询语句

修改视图：

create or replace view 视图名称

as

查询语句

总结：视图中不要出现重名的现象，可以给列起别名

删除视图：

drop view 视图名称;

只读视图：

create view 视图名称

as

查询语句

with read only;

序列：Oracle数据库中的一个数据库对象，自动帮我们生成对应列值，而且每次生成列值可以自动进行增长，经常使用序列来生成表中主键的值

1）创建序列的方式：

create sequence 序列名称

dual表：专门用来做各种测试的表格

序列两个特别的属性

nextval：用于获得序列的下一个值

currval：用户获得序列当前的值

nextval是企业开发中，必然要使用到的属性，currval用于测试

序列的其他属性：

a）序列初始值（默认初值是1）

start with 初值值

b）增长步长（每次增长的数值）

increment by 数值

c）序列最小值

minvalue

d）序列最大值

maxvalue

eg：

create sequence eid\_seq2—创建一个序列

start with 10—序列的初始值

increment by 10—序列的增长值

minvalue 1—序列的最小值

maxvalue 10000000;--序列的最大值（一般默认不写）

约束：

1）约束的概念：在设计表的时候，提前对表中某些列，加入一些限制条件，当插入数据时候，只有当数据满足这些限制条件，才可以正常插入进去，这些限制条件就是约束。保持数据的完整性。

2）约束的种类：

主键约束

外键约束

唯一约束

检查约束

默认值约束

非空约束

主键约束：用来标识表中每一条数据，避免出现重复数据，主键所在列的值，必须要非空而且唯一

1）一张表只能有一个主键（主键约束）

2）指定主键的列，可以是一个列或者是多个列的组合，而且主键所在的列，列值永远都要非空唯一。

主键如果设置到多个列组合，叫做联合主键

指定主键约束名字--在primary key前面加上constraint 约束名

create table 表名(

列名1 数据类型 constraint 主键约束名 primary key,

列名2 数据类型

);

设置联合主键（复合主键），主键又多个列组合而成

create table 表名(

列名1 数据类型,

列名2 数据类型,

列名3 数据类型,

constraint 约束名 primary key(列名1,列名2)

);

添加主键约束：

修改表的时候添加主键（固定格式）：

alter table 表名 add constraint 约束名 primary key(列名);

create table worker4(

sid number(10),

sname varchar2(50)

);

alter table worker4 add constraint pk\_w4 primary key(sid);

删除主键：

a）alter table 表名 drop primary key

仅限于删除主键约束（主键）

alter table worker4 drop primary key;

b）alter table 表名 drop constraint 约束名

适合删除各种约束

唯一约束：用来指定一个列或者多个列的组合具有唯一性，用来防止该列输入重复的数值，定义了唯一约束的列，该列的列值可以出现null值。

1）使用唯一约束修饰列，可以出现null值

2）一个表中可以出现多个唯一约束

3）唯一约束不能修饰主键所在的列

create table 表名(

列名1 数据类型 primary key,

列名2 数据类型 unique（唯一约束）,

列名3 数据类型 unique

);

添加唯一约束：

修改表的时候，添加唯一约束

alter table 表名 add constraint 约束名 unique(列名);

设置多个列用一个唯一约束

alter table 表名 add constraint 约束名 unique(列名1,列名2);

删除唯一约束：

alter table 表名 drop constraint 约束名;

唯一约束跟主键约束的区别

a) 一个表中只能定义一个主键约束，但可以定义多个唯一约束

b) 主键约束所修饰的列，列值永远都是不能为null，唯一约束修饰的列，列值可以为null

检查约束：用于限制某列的列值，必须要满足某些特定的条件，避免用户输入非法数据

创建表的时候设置检查约束：

create table 表名(

列名1 数据类型 primary key（主键）,

列名2 数据类型 unique not null,--唯一约束并且不能为空

列名3 数据类型 check(列名3 in(列值1,列值2，……)),--检查约束，只能输入相应的值

列名4 数据类型 ,

列名5 数据类型

);

添加检查约束：

修改表时候，添加检查约束

alter table customer3（表名）

add constraint chk\_email（约束名）

check(email（列名） like '%@%'（指定的条件）);

删除检查约束：

alter table 表名 drop constraint 约束名;

字符处理函数：字符函数处理字符类型，char,varchar,varchar2

dual:Oracle专门用来进行测试的表

length(列名/字符串):字符的个数

例：

查询emp表中，所有员工的名字，以及名字的字符个数

select ename,length(ename) from emp;

upper(列名/字符串)：将参数转换为大写字母

lower(列名/字符串)：将参数转换为小写字母

substr(参数1，参数2，参数3)：用于截取字符串（全拼）

第1参数：处理字符串

第2参数：如果为正数，表示从头开始进行截取，起始下标为1，如果为负数，表示从倒数第几个开始

第3参数：表示要截取字符的个数

案例：查询emp表中ename的最后两个字符

方法1：

select ename,substr(ename,length(ename)-1,2) from emp;

方法2：

select ename,substr(ename,-2,2) from emp;

instr(参数1，参数2):获得参数2在参数1中第1次出现的位置的下标值

select instr('liuyingqian','qian') from dual

trim(字符串):去掉字符串两端的空格

select trim(' abc ') from dual;

数值处理函数：处理类型是number类型为主

round(数字,位数)：四舍五入，位数如果是正数，表示小数点以后，位数如果是0，没有小数，位数如果是负数，表示小数之前。

trunc(数字，位数):截取函数，根据位数来截取数字，如果位数为正数，截取是小数点以后，如果位数为0，没有小数部分，如果位数为负数，截取小数点之前

abs(数字)：求出绝对值

ceil(数字)：向上取整

floor(数字):向下取整

sign(数字)：如果数字为0，返回值0，如果数字为正数，返回值1，反之如果数字为负数，返回值-1

sqrt(数字)：求平方根

power(数字，指数)：乘方

日期处理函数：

1）sysdate：获取当前系统时间，以天为单位

显示当前系统的时间

select sysdate from dual

2) 常见时间的格式：

yyyy-mm-dd :年月日

yyyy-mm-dd hh24:mi:ss 年月日时分秒

mi：表示分钟，Oracle中不区分大小写，导致mm只能表示月份，mi就是分钟

3）to\_char函数：可以把date转换为指定格式的char进行显示

使用年月日来显示当前时间

select to\_char(sysdate--系统时间,'yyyy-mm-dd') from dual

使用to\_char获得日期的某一部分值

4）to\_date函数：把char类型字符串（满足日期格式）转换为date对象，多用于插入操作

5）months\_between(d1,d2):求出d1和d2之间相差几个月

索引：索引是建立在表中某（几）个列上的数据库对象，用于提高查询的速度。索引一旦创建成功，就会由当前Oracle进行维护，在使用查询语句，不需要再指定具体是哪一个索引。

1）索引用来提高查询的速度，在查询中，经常被使用字段（列）上加入索引，就可以提高查询的速度

2）创建索引的格式：

create index 索引名

on 表名(列名)

3）主键所在的列，默认系统就会添加一个索引，使用主键作为查询的条件，速度很快

4）删除索引：

drop index 索引名字

5）使用unique修饰的列，默认就有一个索引

6）索引总结：

优点：能提高查询的速度（检索速度）

缺点：创建索引会占用磁盘的空间，建立索引以后会加快数据的检索的速度，但是减慢数据修改速度

非空约束：not null，被非空修改列，插入数据时候，其列值不能为null，经常跟unique结合使用

例：

create table worker10(

id number(4) primary key,

name varchar2(50) not null unique

);

默认值约束：可以在表中列上设置默认值，插入数据的时候，如果该列没有插入任何的列值，则系统会自动把默认值插入该列上。每个列只能定义一个默认值。

create table worker11(

id number(4) primary key,

name varchar2(50) not null unique,

hiredate date default sysdate,--默认值约束default

age number(3) default 25

);

创建表以后修改默认值：

使用modify

alter table 表名modify 列名 类型 default 默认值

删除默认值约束,使用modify把默认值设置为null

alter table 表名 modify 列名 数据类型 default null

外键约束：外键约束是用来维护两张表或者更多张表之间的关联关系

foreign references

alter table emp6--表名

add constraint emp6\_dept6\_deptId\_FK --约束名称

foreign key--外键(dept\_Id) references—引用 dept6(id)

alter table 表名

add constraint 约束名

foreign key(子列) references(父列)；

外键：用来维护表与表之间的关联关系，外键所在的表，称为子表，另外一张表为父表

外键所在列的值，依赖于另外一张表中主键的值

规律：如果加入外键约束以后，一般都是先往主表（dept6）中插入数据，然后再往子表（emp6）表中插入数据

子查询：一个查询语句内部又包含了一个查询语句

select ename from emp --查询最小工资的员工

where sal = (

select min(sal) from emp --查询最小工资

);

跟BLAKE是同一个工作的员工的所有信息(不包含BLAKE)

select \* from emp

where job = (

select job from emp

where ename = 'BLAKE'

) and ename != 'BLAKE'

案例：查询KING所属的部门的编号，名称

select deptno ,dname from dept

where deptno = (

select deptno from emp

where ename = 'KING'

);

案例：查询部门名称是ACCOUNTING的下属所有的员工的信息

select \* from emp

where deptno in (

select deptno from dept

where dname = 'ACCOUNTING'

);