

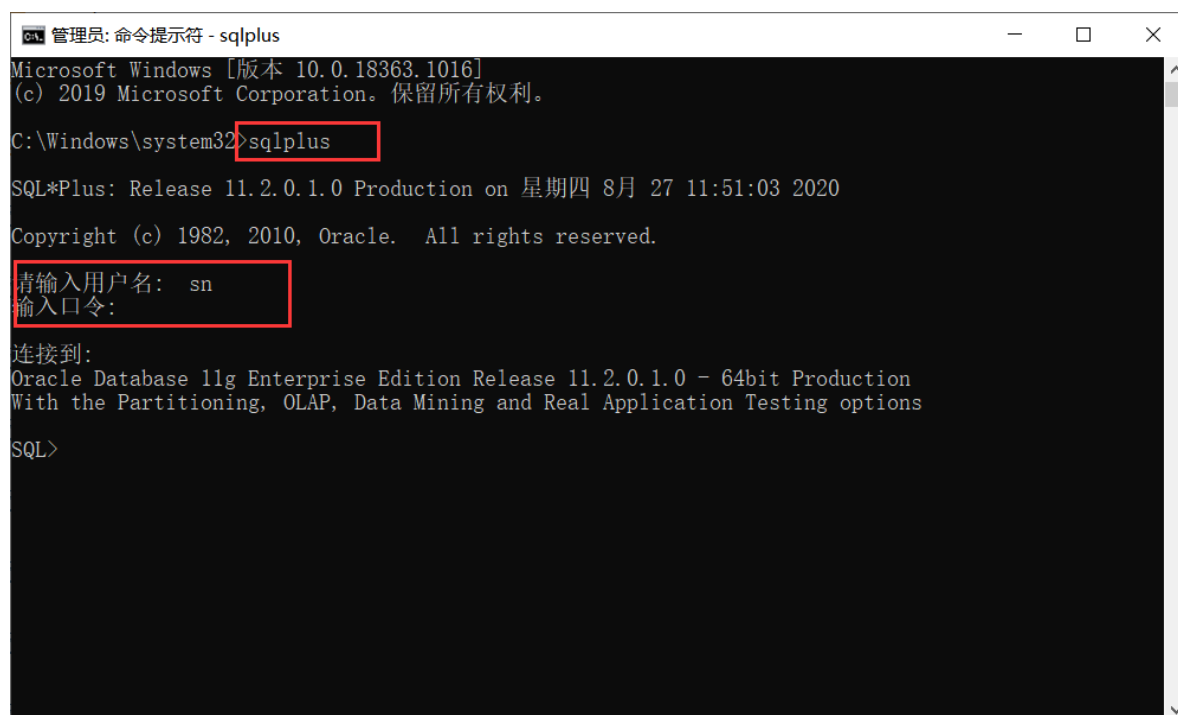
Oracle基本操作

一、SQLPLUS操作

1.1 连接sqlplus

打开命令提示符：输入sqlplus

根据提示输入用户名密码



```
管理员: 命令提示符 - sqlplus
Microsoft Windows [版本 10.0.18363.1016]
(c) 2019 Microsoft Corporation. 保留所有权利。

C:\Windows\system32>sqlplus

SQL*Plus: Release 11.2.0.1.0 Production on 星期四 8月 27 11:51:03 2020

Copyright (c) 1982, 2010, Oracle. All rights reserved.

请输入用户名: sn
输入口令:

连接到:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL>
```

用户名: system
密码: qwe123

默认用户

安装Oracle时，若没有为下列用户重设密码，则其默认密码如下：

用户名 / 密码	登录身份	说明
sys/change_on_install	SYSDBA 或 SYSOPER	不能以 NORMAL 登录，可作为默认的系统管理员
system/manager	SYSDBA 或 NORMAL	不能以 SYSOPER 登录，可作为默认的系统管理员
sysman/oem_temp	sysman	为 oms 的用户名
scott/tiger	NORMAL	普通用户
aqadm /aqadm	SYSDBA 或 NORMAL	高级队列管理员
	SYSDBA 或	

用户名/密码 dbsnmp/dbsnmp	SYSDBA 或 登录身份 SYSDBA	说明 复制管理员
-------------------------	----------------------------	-------------

使用以下三个用户登录

- sys 相当于linux root账户权限最大的用户
 - 用来维护系统信息和管理实例 动态视图
- system 默认系统管理员，管理数据库用户、权限、存储等
- scott 示范账户

```
system/qwe123
```

1.2 用户名密码直接登录

```
C:\Windows\system32>sqlplus sn/123
```

```
sqlplus system/qwe123
```

以管理员身份登陆

```
sqlplus system/qwe123 as sysdba
```

1.3 切换用户

```
connect sn/qwe123;
或
connect system/qwe123 as sysdba;
```

1.4 直接登陆

不使用用户名密码,直接登陆

```
sqlplus /nolog

connect /as sysdba;
```

1.5 修改用户密码

```
alter user 用户名 identified by 密码;

alter user 用户名 account lock/unlock;
```

忘记所有用户密码时，使用无密码以管理员身份登录，再修改用户密码

1.6 显示当前登陆用户

```
show user;
```

1.7 用户表

查看当前用户下的拥有的表

```
select table_name from user_tables;
```

1.8 查看表结构:

```
-- desc 表名  
desc dept;
```

1.9 退出:

```
exit;  
或  
quit;
```

二、表空间

使用pl/sql链接Oracle数据库

2.1 创建表空间

语法格式

```
CREATE TABLESPACE 表空间名  
DATAFILE 'E:\oracledb\文件名.dbf' -- 数据文件  
SIZE 100m -- 表空间大小  
AUTOEXTEND ON NEXT 10m MAXSIZE 200m  
  
-- 创建表空间  
CREATE TABLESPACE tabspace301  
DATAFILE 'E:\oracledb\tabspace301.dbf'  
SIZE 100m -- 'ód?  
AUTOEXTEND ON NEXT 10m MAXSIZE 200m  
  
-- 多个数据文件  
CREATE TABLESPACE A  
DATAFILE 'E:\oracledb\a01.dbf'  
SIZE 100m  
AUTOEXTEND ON NEXT 10m MAXSIZE 200m,  
'E:\oracledb\a02.dbf'  
SIZE 100m  
AUTOEXTEND ON NEXT 10m MAXSIZE 200m,  
'E:\oracledb\a03.dbf'  
SIZE 100m  
AUTOEXTEND ON NEXT 10m MAXSIZE 200m  
  
-- 同一个表空间得多个数据文件可以在不同位置  
CREATE TABLESPACE B  
DATAFILE 'E:\oracledb\b01.dbf'
```

```
SIZE 100m
AUTOEXTEND ON NEXT 10m MAXSIZE 200m,
'F:\- database\ a02.dbf'
SIZE 100m
AUTOEXTEND ON NEXT 10m MAXSIZE 200m
```

2.2 临时表空间

```
-- 创建临时表空间
CREATE TEMPORARY TABLESPACE C
TEMPFILE 'E:\oracledb\C01_temp.dbf'
SIZE 100m
AUTOEXTEND ON NEXT 5m MAXSIZE 150m,
'E:\oracledb\C02_temp.dbf'
SIZE 100m
AUTOEXTEND ON NEXT 5m MAXSIZE 150m
```

2.3 修改表空间

```
-- 修改表空间：添加数据文件
ALTER TABLESPACE A
ADD DATAFILE 'E:\oracledb\ a04.dbf'
SIZE 100m
AUTOEXTEND ON NEXT 10m MAXSIZE 200m

-- 修改表空间数据文件大小
ALTER DATABASE DATAFILE 'E:\oracledb\ a04.dbf' RESIZE 10m;
```

2.4 查询表空间

```
-- 查询表空间
SELECT * FROM dba_free_space WHERE tablespace_name='A';

-- 查询表空间大小
SELECT SUM(bytes) FROM dba_free_space WHERE tablespace_name='A';

-- 查询数据文件
SELECT * FROM dba_temp_files; -- 临时表空间数据文件
SELECT * FROM dba_data_files WHERE tablespace_name='A'; -- 表空间数据文件

-- 查询所有表空间
SELECT tablespace_name FROM dba_free_space GROUP BY tablespace_name;
```

2.5 删除表空间

```
-- 删除表空间不删除数据文件
DROP TABLESPACE A;

-- 删除表空间和数据文件
DROP TABLESPACE C INCLUDING CONTENTS AND DATAFILES;
```

三、用户

3.1 创建用户

```
-- 创建用户
/*
CREATE USER 用户名
IDENTIFIED BY 密码
DEFAULT TABLESPACE 默认表空间
TEMPORARY TABLESPACE 默认临时表空间
*/
CREATE USER test301
IDENTIFIED BY 123
DEFAULT TABLESPACE TEST_TABLE
TEMPORARY TABLESPACE TEST_TEMP

-- 查询用户
SELECT * FROM dba_users WHERE username='TEST301';

-- 删除用户
DROP USER TEST301;
```

3.2 授权

```
-- 授权
GRANT CONNECT TO TEST301; -- 链接权限，仅可以登录/查看
GRANT RESOURCE TO TEST301; -- 资源权限，可以操作数据库资源
GRANT DBA TO TEST301; -- 管理员权限，可以添加授权用户

GRANT CONNECT,RESOURCE,DBA TO TEST301;

-- 表授权
GRANT DELETE ON a TO test301;

-- 撤销权限
REVOKE CONNECT FROM TEST301;
REVOKE RESOURCE FROM TEST301;
REVOKE DBA FROM TEST301;
REVOKE CONNECT,RESOURCE,DBA FROM TEST301;
```

3.3 修改密码

```
-- 修改用户密码
ALTER USER TEST301 IDENTIFIED BY qwe123;
```

作业

```
-- 作业
-- 1.创建表空间test_table
-- 2.创建临时表空间test_temp
-- 3.创建用户,分配表空间,临时表空间
-- 4.授权
```

```
SELECT * FROM student;
SELECT * FROM classes;
SELECT * FROM score;
SELECT * FROM course;
SELECT * FROM teacher;
```

四、数据表

4.1 创建数据表

```
/*
CREATE TABLE 表名
(
    列名 类型 约束,
    .....
)
TABLESPACE 表空间
*/

-- 创建表: 不声明表空间, 默认使用当前用户的默认表空间
CREATE TABLE tests
(
    tid NUMBER(4), -- 没有约束不定义
    tname VARCHAR2(10)
)
```

建表声明表空间

```
-- 创建表: 声明表空间
CREATE TABLE tests
(
    tid NUMBER(4), -- 没有约束不定义
    tname VARCHAR2(10)
)
TABLESPACE TEST
```

表约束

```
-- 创建表: 约束
CREATE TABLE department
(
    -- 注释信息
    did NUMBER(4) PRIMARY KEY,
    dname VARCHAR2(10) NOT NULL,
    remark VARCHAR2(11) UNIQUE,
    times DATE
);
```

```

-- 给表添加注释信息: COMMENT ON TABLE 表名 IS '注释信息';
COMMENT ON TABLE department IS '部门表';
-- 给列添加注释信息: COMMENT ON COLUMN 表名.列名 IS '注释信息';
COMMENT ON COLUMN department.did IS '主键ID';
COMMENT ON COLUMN department.dname IS '部门名称';
COMMENT ON COLUMN department.remark IS '备注';
COMMENT ON COLUMN department.times IS '修改时间';

CREATE TABLE employee
(
    -- PRIMARY KEY主键约束: 主键默认非空, 唯一
    eid NUMBER(4) PRIMARY KEY,
    -- NOT NULL非空: 非空约束, 添加时该字段必须录入, 默认可以为null
    ename VARCHAR2(10) NOT NULL,
    -- CHECK约束: 添加时该字段的值只能是0/1
    sex NUMBER(1) CHECK(sex=0 OR sex=1),
    state NUMBER(1) CHECK(state IN (0,1,2)),
    -- UNIQUE: 唯一约束, 该字段值唯一, 不能重复
    phone VARCHAR2(11) UNIQUE,
    -- REFERENCES: 外键约束, 字段的值引用department表中的did列
    did NUMBER(4) REFERENCES department(did),
    times DATE
);
-- 给表添加注释信息: COMMENT ON TABLE 表名 IS '注释信息';
COMMENT ON TABLE employee IS '员工表';
-- 给列添加注释信息: COMMENT ON COLUMN 表名.列名 IS '注释信息';
COMMENT ON COLUMN employee.eid IS '主键ID';
COMMENT ON COLUMN employee.ename IS '员工姓名';
COMMENT ON COLUMN employee.sex IS '性别: 0女, 1男';
COMMENT ON COLUMN employee.state IS '状态: 0在职, 1离职, 2休假';
COMMENT ON COLUMN employee.phone IS '联系电话';
COMMENT ON COLUMN employee.did IS '部门编号';
COMMENT ON COLUMN employee.times IS '修改时间';

```

4.2 删除表

```

-- drop table 表名

drop table tests;

```

五、CRUD

5.1 新增insert

```

-- INSERT INTO 表名(列名) values(列值)
-- 只添加声明的列, 未声明的自动填写默认值, 一列对应一个值
INSERT INTO department(did,dname,times) VALUES(1,'开发部',SYSDATE);

-- 列名可省略, 省略后默认为表中的所有列, 顺序与表中的列顺序一致
INSERT INTO department VALUES(2,'学术部','学术部',SYSDATE);

```

5.2 修改update

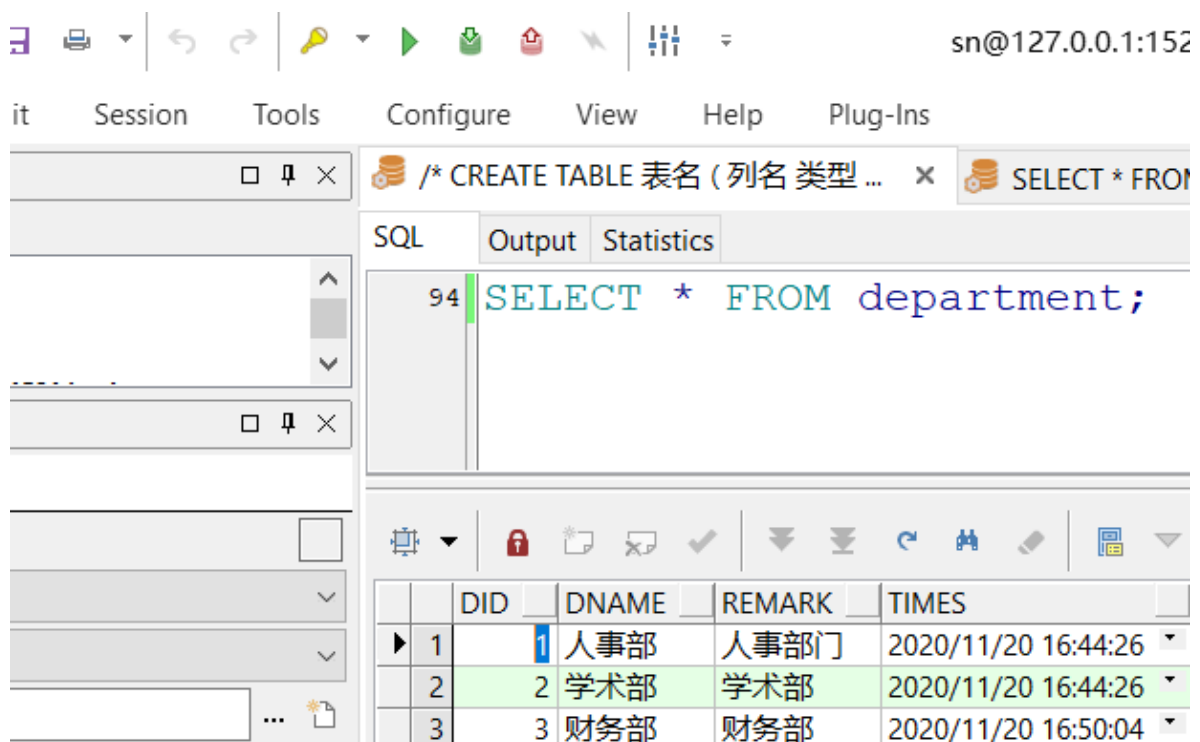
```
-- UPDATE 表名 SET 列名=列值 [WHERE 条件];  
UPDATE department SET dname='人事部' WHERE did=1;  
UPDATE department SET dname='人事部',remark='人事部门' WHERE did=1;  
UPDATE department SET times=SYSDATE;
```

5.3 删除delete

```
-- 删除表数据  
-- DELETE [FROM] 表名 [where 条件]  
DELETE FROM department;  
DELETE FROM department WHERE did=2;  
DELETE department WHERE did=2;  
  
-- 清空表数据，效果等同于delete不添加where条件  
TRUNCATE TABLE department;
```

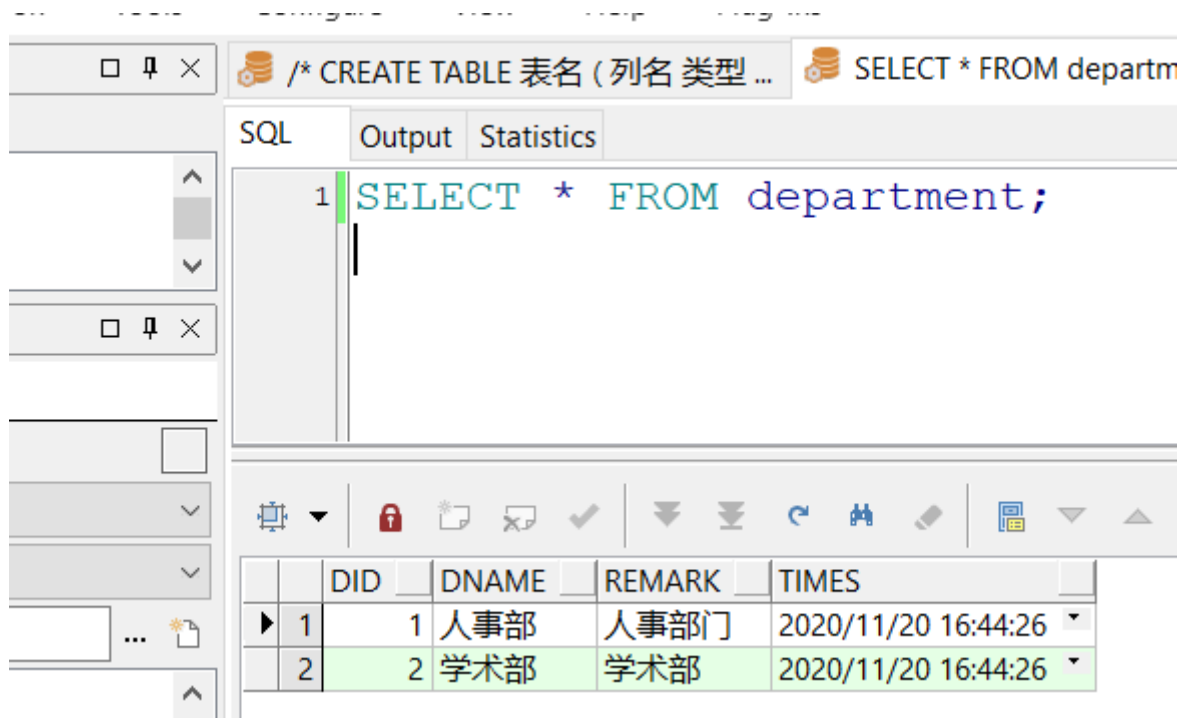
5.4 提交回滚 commit,rollback

oracle中默认事务提交方式为手动提交，执行增删改后，需要手动提交事务，数据才能永久保存。



The screenshot shows the SQL Developer interface. The top toolbar includes icons for file operations, undo/redo, and execution. The main window has tabs for Session, Tools, Configure, View, Help, and Plug-Ins. The SQL editor displays the query: `SELECT * FROM department;`. Below the editor, the Results pane shows the output of the query as a table with columns DID, DNAME, REMARK, and TIMES.

	DID	DNAME	REMARK	TIMES
1	1	人事部	人事部门	2020/11/20 16:44:26
2	2	学术部	学术部	2020/11/20 16:44:26
3	3	财务部	财务部	2020/11/20 16:50:04



第一个窗口添加数据未提交后，当前窗口查询，显示3条数据，新窗口查询只显示2条数据，新增的数据只是临时保存在数据库表，需要提交事务后，才能永久保存。



绿色提交，红色回滚。

或者使用sql命令提交/回滚事务

```
-- 提交
COMMIT;

-- 回滚
ROLLBACK;
```

提交相当于ctrl+s，回滚相当于撤销ctrl+z。

5.5 查询select

SELECT

select命令结构：

select * | 列名 | 表达式 from 表名 [别名] [where 条件] [order by 列名]

```
-- 查询学生表信息

-- *查询所有字段
SELECT * FROM student;
-- 查询部分字段
SELECT stuno,stuname,idcard FROM student;

-- 别名，添加别名后查询字段时，需要添加别名
SELECT * FROM student s;
SELECT s.* FROM student s;
```

WHERE

where 后跟查询条件

下面的运算符可在 WHERE 子句中使用：

操作符	描述
=	等于
<>	不等于
>	大于
<	小于
>=	大于等于
<=	小于等于
BETWEEN	在某个范围内
LIKE	搜索某种模式

```
-- 查询成绩等于100分的信息
SELECT * FROM score WHERE score=100;
SELECT * FROM score WHERE score>60;
SELECT * FROM score WHERE score<60;
SELECT * FROM score WHERE score>=60;
SELECT * FROM score WHERE score<=60;
SELECT * FROM score WHERE score!=60;
SELECT * FROM score WHERE score<>60;

-- 查询成绩在60-80分之间的信息
SELECT * FROM score WHERE score between 60 and 80;

-- 查询学号为S001的学生信息
SELECT * FROM student WHERE stuno='S001';

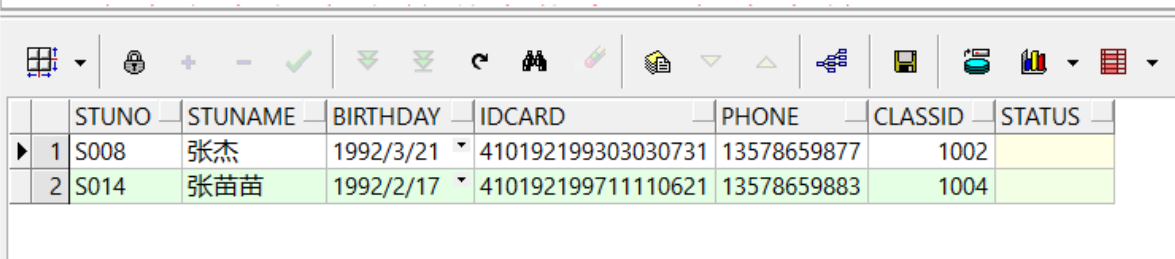
-- 查询学号不等于S001的学生信息
SELECT * FROM student WHERE stuno != 'S001';

-- 查询姓张的学生信息：%任意字符,_一个字符
SELECT * FROM student WHERE stuname LIKE '张%';
SELECT * FROM student WHERE stuname LIKE '张_';
SELECT * FROM student WHERE stuname LIKE '%张%';
```

-- 查询姓张的学生信息：%任意字符,_ 一个字符

```
SELECT * FROM student WHERE stuname LIKE '张%';
```

```
SELECT * FROM student WHERE stuname LIKE '张_';
```

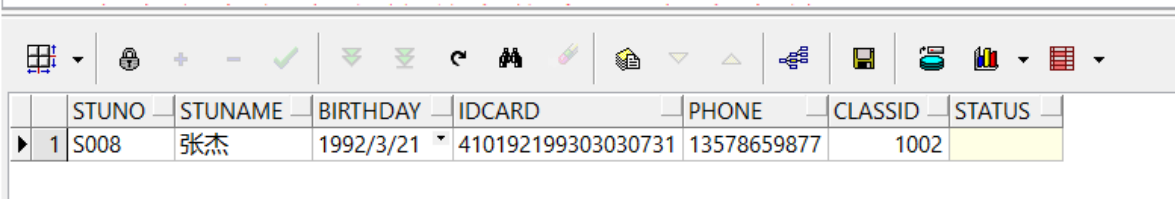


	STUNO	STUNAME	BIRTHDAY	IDCARD	PHONE	CLASSID	STATUS
▶ 1	S008	张杰	1992/3/21	410192199303030731	13578659877	1002	
2	S014	张苗苗	1992/2/17	410192199711110621	13578659883	1004	

-- 查询姓张的学生信息：%任意字符,_ 一个字符

```
SELECT * FROM student WHERE stuname LIKE '张%';
```

```
SELECT * FROM student WHERE stuname LIKE '张_';
```



	STUNO	STUNAME	BIRTHDAY	IDCARD	PHONE	CLASSID	STATUS
▶ 1	S008	张杰	1992/3/21	410192199303030731	13578659877	1002	

AND & OR

AND 和 OR 可在 WHERE 子语句中把两个或多个条件结合起来。

如果第一个条件和第二个条件都成立，则 AND 运算符显示一条记录。

如果第一个条件和第二个条件中只要有一个成立，则 OR 运算符显示一条记录。

-- 查询姓名为张杰并且班级编号为1002的学生信息

```
SELECT * FROM student WHERE stuname='张杰' AND classid=1002;
```

-- 查询姓名为张杰或者手机号为13578659877的学生信息

```
SELECT * FROM student WHERE stuname='张杰' OR phone='13578659877';
```

ORDER BY

ORDER BY 语句用于根据指定的列对结果集进行排序。

ASC升序，DESC降序，默认按照升序对记录进行排序。

-- order by 字段 [asc/desc]: 查询并按照某个字段排序,默认升序

```
SELECT * FROM student ORDER BY stuno;
```

```
SELECT * FROM student ORDER BY stuno ASC;
```

```
SELECT * FROM student ORDER BY stuno DESC;
```

IN

IN 操作符允许我们在 WHERE 子句中规定多个值。

```
SELECT * FROM student WHERE classid in (1001,1002,1003);
```

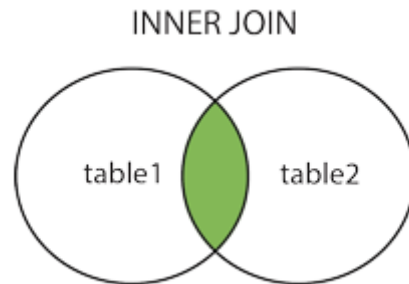
-- 等价于

```
SELECT * FROM student WHERE classid=1001 OR classid=1002 OR classid=1003;
```

JOIN

INNER JOIN

INNER JOIN产生的结果集中，是1和2的交集。



```
SELECT * FROM classes c
INNER JOIN teacher t
ON c.teacher=t.tid;
```

```
SELECT * FROM teacher;
```

```
SELECT * FROM classes c
INNER JOIN teacher t
ON c.teacher=t.tid;
```

	CLASSID	CLASSNAME	STATE	HEADMASTER	TEACHER	TID	TNAME	POSTS	TSTATE
1	1004	GS193	0	T06	T01	T01	刘璐	2	1
2	1001	GS190	0	T04	T01	T01	刘璐	2	1
3	1005	GS194	1	T05	T02	T02	陈迪	2	1
4	1003	GS192	1	T05	T02	T02	陈迪	2	1
5	1002	GS191	1	T05	T02	T02	陈迪	2	1

写Inner 和不写Inner 的话，没有区别

```
SELECT * FROM classes c
INNER JOIN teacher t
ON c.teacher=t.tid;
```

等同于

```
SELECT * FROM classes c
JOIN teacher t
ON c.teacher=t.tid;
```

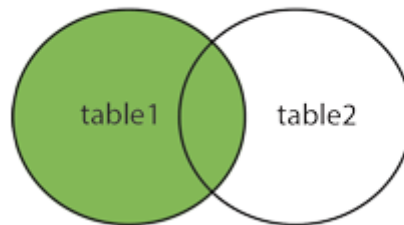
等同于

```
SELECT * FROM classes c,teacher t
ON c.teacher=t.tid;
```

LEFT JOIN

LEFT JOIN产生表1的完全集，而2表中匹配的则有值，没有匹配的则以null值取代。左表数据一定全部返回

LEFT JOIN



```
SELECT * FROM classes c
LEFT JOIN teacher t
ON c.teacher=t.tid;
```

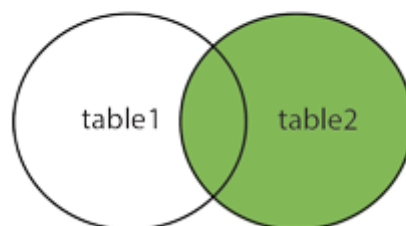
```
SELECT * FROM classes c
LEFT JOIN teacher t
ON c.teacher=t.tid;
```

	CLASSID	CLASSNAME	STATE	HEADMASTER	TEACHER	TID	TNAME	POSTS	TSTATE
1	1004	GS193	0	T06	T01	T01	刘璐	2	1
2	1001	GS190	0	T04	T01	T01	刘璐	2	1
3	1005	GS194	1	T05	T02	T02	陈迪	2	1
4	1003	GS192	1	T05	T02	T02	陈迪	2	1
5	1002	GS191	1	T05	T02	T02	陈迪	2	1

RIGHT JOIN

RIGHT JOIN产生表2的完全集，而1表中匹配的则有值，没有匹配的则以null值取代。右表数据一定全部返回

RIGHT JOIN



```
SELECT * FROM classes c
RIGHT JOIN teacher t
ON c.teacher=t.tid;
```

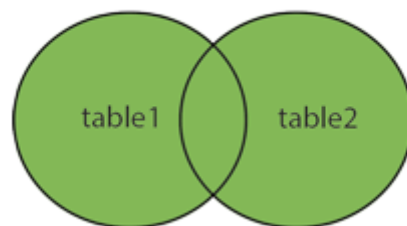
```
SELECT * FROM classes c
RIGHT JOIN teacher t
ON c.teacher=t.tid;
```

	CLASSID	CLASSNAME	STATE	HEADMASTER	TEACHER	TID	TNAME	POSTS	TSTATE
1	1001	GS190	0	T04	T01	T01	刘璐	2	1
2	1002	GS191	1	T05	T02	T02	陈迪	2	1
3	1003	GS192	1	T05	T02	T02	陈迪	2	1
4	1004	GS193	0	T06	T01	T01	刘璐	2	1
5	1005	GS194	1	T05	T02	T02	陈迪	2	1
6						T08	顾天天	2	1
7						T07	刘佳	1	0
8						T09	严岩	2	1
9						T06	陈宏	1	1
10						T04	赵明	1	1
11						T05	胡丹丹	1	1
12						T03	李欣	2	1

FULL OUTER JOIN

FULL JOIN 会从左表 和右表 那里返回所有的行。如果其中一个表的数据行在另一个表中没有匹配的行，那么对面的数据用NULL代替，所有数据都返回

FULL OUTER JOIN



```
SELECT * FROM classes c
FULL OUTER JOIN teacher t
ON c.teacher=t.tid;
```

```
SELECT * FROM classes c
FULL OUTER JOIN teacher t
ON c.teacher=t.tid;
```

	CLASSID	CLASSNAME	STATE	HEADMASTER	TEACHER	TID	TNAME	POSTS	TSTATE
1	1004	GS193	0	T06	T01	T01	刘璐	2	1
2	1001	GS190	0	T04	T01	T01	刘璐	2	1
3	1005	GS194	1	T05	T02	T02	陈迪	2	1
4	1003	GS192	1	T05	T02	T02	陈迪	2	1
5	1002	GS191	1	T05	T02	T02	陈迪	2	1
6						T03	李欣	2	1
7						T04	赵明	1	1
8						T05	胡丹丹	1	1
9						T06	陈宏	1	1
10						T07	刘佳	1	0
11						T08	顾天天	2	1
12						T09	严岩	2	1

DISTINCT

在表中，可能会包含重复值。需要仅列出不同 (distinct) 的值。DISTINCT 用于返回唯一不同的值。

```
-- SELECT DISTINCT 列名称 FROM 表名称
```

```
SELECT classid FROM student;
SELECT distinct(classid) FROM student;
```

```
SELECT classid FROM student;
SELECT distinct(classid) FROM student;
```

	CLASSID
1	1003
2	1001
3	1002
4	1004

UNION 和 UNION ALL

UNION 操作符用于合并两个或多个 SELECT 语句的结果集。UNION会自动去除重复数据，UNION ALL 不去重。

```

SELECT * FROM student WHERE stuname='张杰'
UNION
SELECT * FROM student WHERE stuname='赵敏'
UNION
SELECT * FROM student WHERE stuname='张杰'

```

```

SELECT * FROM student WHERE stuname='张杰'
UNION
SELECT * FROM student WHERE stuname='赵敏'
UNION
SELECT * FROM student WHERE stuname='张杰'

```

	STUNO	STUNAME	BIRTHDAY	IDCARD	PHONE	CLASSID	STATUS
1	S008	张杰	1992/3/21	410192199303030731	13578659877	1002	
2	S012	赵敏	1992/6/1	410192199212060901	13578659881	1003	

```

SELECT * FROM student WHERE stuname='张杰'
UNION ALL
SELECT * FROM student WHERE stuname='赵敏'
UNION ALL
SELECT * FROM student WHERE stuname='张杰'

```

```

SELECT * FROM student WHERE stuname='张杰'
UNION ALL
SELECT * FROM student WHERE stuname='赵敏'
UNION ALL
SELECT * FROM student WHERE stuname='张杰'

```

	STUNO	STUNAME	BIRTHDAY	IDCARD	PHONE	CLASSID	STATUS
1	S008	张杰	1992/3/21	410192199303030731	13578659877	1002	
2	S012	赵敏	1992/6/1	410192199212060901	13578659881	1003	
3	S008	张杰	1992/3/21	410192199303030731	13578659877	1002	