



# MySQL数据库开发技术

## —— 简单查询

# 本章内容

节	知识点	掌握程度	难易程度
结构化查询语言	结构化查询语言简介	理解	
	结构化查询语言分类	理解	
基本SELECT语句	基本SELECT语句作用	掌握	
	基本SELECT语句语法	掌握	
SQL概念和规则	SQL语句相关概念	理解	
	SQL语句的书写规则	理解	
选择列	选择所有列	掌握	
	选择特定列	掌握	
算术运算符	算术运算符	掌握	
	算术运算符优先级	掌握	
	使用括号改变优先级	掌握	
空值NULL	空值NULL	掌握	
	算术表达式中的空值NULL	掌握	
列别名	列别名	掌握	
	列别名使用	掌握	
	使用列别名的方法	掌握	
消除重复行	消除重复行	掌握	
	使用DISTINCT关键字	掌握	

# 本章内容

节	知识点	掌握程度	难易程度
选择限定数据行	WHERE子句的作用	理解	
	WHERE子句的语法	掌握	
	比较运算符	掌握	
使用WHERE子句	比较数值型数据	掌握	
	比较字符型数据	掌握	
	比较日期型数据	掌握	
特殊比较运算符	特殊比较运算符	掌握	
	BETWEEN...AND...运算符	掌握	
	IN运算符	掌握	
	LIKE运算符	掌握	
	IS NULL运算符	掌握	
逻辑运算符	逻辑运算符	理解	
	逻辑与	掌握	
	逻辑或	掌握	
	逻辑非	掌握	
	运算符的优先级	理解	

# 本章内容

节	知识点	掌握程度	难易程度
ORDER BY子句	ORDER BY子句语法	掌握	
	排序规则	掌握	
	按列名升序排序	掌握	
	按列名降序排列	掌握	
	按列别名排序	掌握	
	多列参与排序	掌握	
	按结果集列序号排序	理解	
限制记录行数	限制记录行数	掌握	

# 结构化查询语言

- 结构化查询语言简介

- 结构化查询语言 (Structured Query Language) 简称SQL, 是操作和检索关系型数据库的标准语言, 20世纪70年代由IBM公司开发, 目前应用于各种关系型数据库。
- SQL的发展
  - 1974年首次提出, 当时叫SEQUEL
  - 1980年改名为SQL
  - 1986年, ANSI定义关系数据库语言的标准, 并公布了标准SQL
  - 1992年, 通过的修改标准SQL-92
  - 1999年, 发布SQL99标准
  - 2003年, 发布SQL2003标准

# 结构化查询语言

- 结构化查询语言分类

- 结构化查询语言可分为5类：

- 数据查询语言（DQL:Data Query Language）：语句主要包括SELECT，用于从表中检索数据。
    - 数据操作语言（DML: Data Manipulation Language）：语句主要包括INSERT，UPDATE和DELETE，用于添加，修改和删除表中的行数据。
    - 事务处理语言（TPL:Transaction Process Language）：语句主要包括COMMIT和ROLLBACK，用于提交和回滚。
    - 数据控制语言（DCL:Data Control Language）：语句主要包括GRANT和REVOKE，用于进行授权和收回权限。
    - 数据定义语言（DDL:Data Definition Language）：语句主要包括CREATE、DROP、ALTER，用于定义、销毁、修改数据库对象。

# 基本SELECT语句

- 基本SELECT语句作用

选择

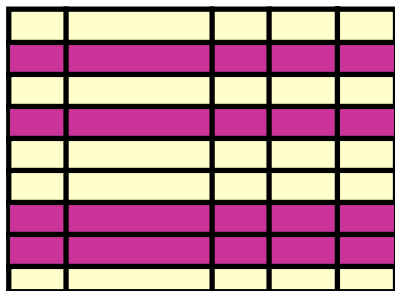



表1

投影

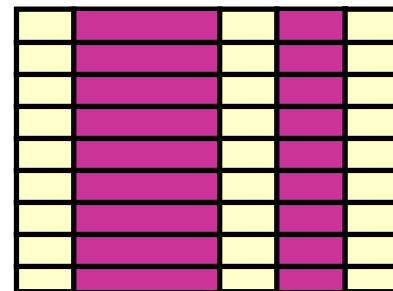



表 1

连接

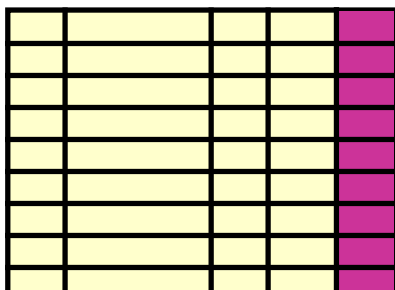



表 1

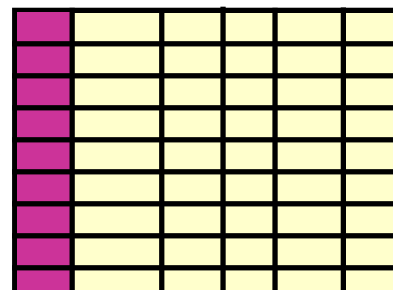



表 2

# 基本SELECT语句

- 基本SELECT语句语法

```
SELECT    [DISTINCT] { * | column | expression [alias] , ... }  
FROM      table;
```

- SELECT子句 表示所需检索的数据列。
- FROM子句 表示检索的数据来自哪个表。



# SQL概念和规则

- SQL语句相关概念

- 关键字 (Keyword) : SQL语言保留的字符串, 例如, SELECT 和FROM都是关键字。
- 语句 (statement) : 一条完整的SQL命令。例如, SELECT \* FROM dept 是一条语句。
- 子句 (clause) : 部分的SQL语句, 通常是由关键字加上其它语法元素构成, 例如, SELECT \* 是一个子句, FROM table也是一个子句。

# SQL概念和规则

- SQL语句的书写规则

- 不区分大小写，也就是说SELECT, select, Select, 执行时效果是一样的。
- 可以单行来书写，也可以书写多行，建议分多行书写，增强代码可读性，通常以子句为单位进行分行。
- 关键字不可以缩写、分开以及跨行书写，如SELECT不可以写成SEL或SELE CT等形式。
- 关键字最好使用大写，其它语法元素（如列名、表名等）小写。
- Tab和缩进的使用可以提高程序的可读性。

# 选择列

- 选择所有列

```
SQL> SELECT *  
2 FROM dept;
```

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

# 选择列

- 选择所有列

```
SQL> SELECT deptno, dname, loc  
2 FROM dept;
```

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

试比较哪条语句执行效率更高？

# 选择列

- 选择指定的列

```
SQL> SELECT deptno, loc  
2 FROM dept;
```

DEPTNO	LOC
10	NEW YORK
20	DALLAS
30	CHICAGO
40	BOSTON

# 练习1

- 1. 使用两种方式查询所有员工(EMP)信息
- 2. 查询(EMP)员工编号、员工姓名、员工职位、员工月薪、工作部门编号。

# 算术运算符

- 算术运算符

- 可以在SELECT语句中使用算术运算符，改变输出结果。

运算符	描述
+	加
-	减
*	乘
/	除

# 算术运算符

- 算术运算符

```
SQL> SELECT ename, sal, sal+300  
2 FROM emp;
```

ENAME	SAL	SAL+300
KING	5000	5300
BLAKE	2850	3150
CLARK	2450	2750
JONES	2975	3275
MARTIN	1250	1550
ALLEN	1600	1900

...

14 rows selected.



# 算术运算符

- 算术运算符优先级

- 乘除优先于加减
- 相同优先权的表达式按照从左至右的顺序依次计算
- 括弧可以提高优先权，并使表达式的描述更为清晰

```
SQL> SELECT ename, sal, 12*sal+100  
2 FROM emp;
```

ENAME	SAL	12*SAL+100
KING	5000	60100
BLAKE	2850	34300
CLARK	2450	29500
JONES	2975	35800
MARTIN	1250	15100
ALLEN	1600	19300

...

14 rows selected.

# 算术运算符

- 使用括号改变优先级

```
SQL> SELECT ename, sal, 12*(sal+100)
2 FROM emp;
```

ENAME	SAL	12*(SAL+100)
KING	5000	61200
BLAKE	2850	35400
CLARK	2450	30600
JONES	2975	36900
MARTIN	1250	16200

...

14 rows selected.

# 练习2

- 1. 员工转正后，月薪上调20%，请查询出所有员工转正后的月薪。
- 2. 员工试用期6个月，转正后月薪上调20%，请查询出所有员工工作第一年的年薪所得（不考虑奖金部分，年薪的试用期6个月的月薪+转正后6个月的月薪）

# 空值NULL

- 空值NULL
  - 空值是指一种无效的、未赋值、未知的或不可用的值。
  - 空值不同于零或者空格。

```
SQL> SELECT ename, job, sal, comm  
2 FROM emp;
```

ENAME	JOB	SAL	COMM
KING	PRESIDENT	5000	
BLAKE	MANAGER	2850	
...			
TURNER	SALESMAN	1500	0
...			

14 rows selected.

# 空值NULL

- 算术表达式中的空值NULL
  - 任何包含空值的算术表达式运算后的结果都为空值NULL。

```
SQL> select  ename, 12*sal+comm  
2   from    emp  
3  WHERE    ename='KING' ;
```

ENAME	12*SAL+COMM
-----	-----
KING	

# 列别名

- 列别名
  - 用来重新命名列的显示标题
  - 如果SELECT语句中包含计算列，通常使用列别名来重新定义列标题。
- 使用列别名的方法
  - 方式1：列名 列别名
  - 方式2：列名 AS 列别名
- 以下三种情况列别名两侧需要添加双引号
  - 列别名中包含有空格
  - 列别名中要求区分大小写
  - 列别名中包含有特殊字符

# 列别名

- 列别名使用

```
SQL> SELECT ename AS name, sal salary  
2 FROM emp;
```

NAME	SALARY
-----	
...	

```
SQL> SELECT ename "Name",  
2 sal*12 "Annual Salary"  
3 FROM emp;
```

Name	Annual Salary
-----	
...	

# 练习3

- 1. 员工试用期6个月，转正后月薪上调20%，请查询出所有员工工作第一年的所有收入（需考虑奖金部分），要求显示列标题为员工姓名，工资收入，奖金收入，总收入。



# 消除重复行

- 重复行

- 以下查询的结果默认输出所有行，其中包含了重复行

```
SQL> SELECT deptno  
2 FROM emp;
```

```
DEPTNO  
-----  
10  
30  
10  
20  
...  
14 rows selected.
```

# 消除重复行

- 消除重复行

- 在SELECT字句中使用关键字DISTINCT可消除重复行。

```
SQL> SELECT DISTINCT deptno  
2 FROM emp;
```

DEPTNO
10
20
30

-----

# 练习4

- 1. 查询员工表中一共有哪几种岗位类型。



# 显示表的结构

- 可以使用DESCRIBE 命令来查看表结构

```
DESC[RIBE] tablename
```

# 练习5

1. 分别查看员工表、部门表、薪资等级表的表结构。

# 选择限定数据行

EMP

EMPNO	ENAME	JOB	...	DEPTNO
7839	KING	PRESIDENT		10
7698	BLAKE	MANAGER		30
7782	CLARK	MANAGER		10
7566	JONES	MANAGER		20
		...		

“...检索所有  
在10部门的员工”

EMP

EMPNO	ENAME	JOB	...	DEPTNO
7839	KING	PRESIDENT		10
7782	CLARK	MANAGER		10
7934	MILLER	CLERK		10

# 选择限定数据行

- 使用WHERE子句可以返回限定的数据行, 语法如下:

```
SELECT [DISTINCT] { * | column | expression [alias], ... }  
FROM   table  
[WHERE condition(s)];
```

- WHERE子句紧跟在FROM子句之后
- condition(s) 表示条件表达式,
  - 通常格式为: 列名 比较操作符 要比较的值

# 选择限定数据行

- 比较操作符

操作符	含义
=	等于
>	大于
>=	大于或等于
<	小于
<=	小于或等于
<>	不等于



# 使用WHERE子句

- 比较数值型数据

```
SQL> SELECT ename, job, deptno  
2 FROM emp  
3 WHERE deptno=20;
```

ENAME	JOB	DEPTNO
-----	-----	-----
JAMES	CLERK	20
SMITH	MANAGER	20
ADAMS	ANALYST	20
MILLER	CLERK	20

# 使用WHERE子句

- 比较字符型数据

```
SQL> SELECT ename, job, deptno  
2 FROM emp  
3 WHERE job='CLERK';
```

ENAME	JOB	DEPTNO
-----	-----	-----
JAMES	CLERK	30
SMITH	CLERK	20
ADAMS	CLERK	20
MILLER	CLERK	10

- 字符型数据作为被比较的值时，必须用单引号引起来
- 字符型数值区分大小写

# 使用WHERE子句

- 比较日期型数据

```
SQL> SELECT ename, hiredate, deptno  
2 FROM emp  
3 WHERE hiredate > 1985-01-01';
```

ENAME	HIREDATE	DEPTNO
-----	-----	-----
SCOTT	1987-4-19	20
ADAMS	1987-5-23	20

- 日期型数值作为被比较的值时，必须用单引号引起来。

# 练习6

- 1. 查询职位为SALESMAN的员工编号、职位、入职日期。
- 2. 查询1985年12月31日之前入职的员工姓名及入职日期。
- 3. 查询部门编号不在10部门的员工姓名、部门编号。

# 特殊比较运算符

- 特殊比较运算符

运算符	含义
<b>BETWEEN...AND...</b>	判断要比较的值是否在某个范围内。
<b>IN ( 集合列表)</b>	判断要比较的值是否和集合列表中的任何一个值相等。
<b>LIKE</b>	判断要比较的值是否满足部分匹配。
<b>IS NULL</b>	判断要比较的值是否为空值NULL。

# 特殊比较运算符

- BETWEEN .. AND ..

- 使用BETWEEN .. AND .. 运算符来判断要比较的值是否在某个范围内。

```
SQL> SELECT  ename, sal  
2  FROM      emp  
3  WHERE     sal BETWEEN 1000 AND 1500;
```

ENAME	SAL
-----下限-----上限	
MARTIN	1250
TURNER	1500
WARD	1250
ADAMS	1100
MILLER	1300

# 特殊比较运算符

- IN运算符

- 使用IN运算符判断要比较的值是否和集合列表中的任何一个值相等。

```
SQL> SELECT empno, ename, sal, mgr
2 FROM emp
3 WHERE mgr IN (7902, 7566, 7788);
```

EMPNO	ENAME	SAL	MGR
7902	FORD	3000	7566
7369	SMITH	800	7902
7788	SCOTT	3000	7566
7876	ADAMS	1100	7788

# 练习7

- 1. 查询入职日期在82年至85年的员工姓名，入职日期。
- 2. 查询月薪在3000到5000的员工姓名，月薪。
- 3. 查询部门编号为10或者20的员工姓名，部门编号。
- 4. 查询经理编号为7902， 7566， 7788的员工姓名，经理编号。



# 特殊比较运算符

- LIKE运算符

- 使用LIKE运算符判断要比较的值是否满足部分匹配，也叫模糊查询。模糊查询中两个通配符：
  - % 代表零或任意更多的字符
  - \_ 代表一个字符

```
SQL> SELECT  ename
      2  FROM    emp
      3  WHERE   ename LIKE 'S%';
```

# 特殊比较运算符

- LIKE运算符

- %与\_组合使用

```
SQL> SELECT  ename
      2  FROM    emp
      3  WHERE   ename LIKE ' _L%';
```

- LIKE运算符

- 您可以使用ESCAPE标识符实现对“%”和“\_”的查找。
  - 例：查询 JOB 以“MAN\_”开头的雇员信息。

```
SQL> SELECT  ename, job
      2  FROM    emp
      3  WHERE   job LIKE 'MAN@_%' ESCAPE '@';
```

# 特殊比较运算符

- IS NULL 运算符

- 使用 IS NULL 运算符来判断要比较的值是否为空值NULL

```
SQL> SELECT  ename, mgr
      2  FROM    emp
      3  WHERE  mgr IS NULL;
```

ENAME	MGR
-----	-----
KING	

# 练习8

- 1. 查询员工姓名以W开头的员工姓名。
- 2. 查询员工姓名倒数第2个字符为T的员工姓名。
- 3. 查询奖金为空的员工姓名， 奖金。

# 逻辑运算符

- 逻辑运算符

- 当需要和多个条件表达式进行比较时，需要使用逻辑运算符把多个表达式连接起来，逻辑运算符包括AND、OR、NOT，逻辑表达式的结果为TRUE，FALSE，NULL。

运算符	含义
AND	逻辑与，用来连接多个条件表达式。如果每个条件表达式的结果都为TRUE，整个表达式的结果才为TRUE。
OR	逻辑或，用来连接多个条件表达式。只要有1个条件表达式的结果为TRUE，整个表达式的结果就为TRUE。
NOT	逻辑非，用来对条件表达式取反。TRUE取反为FALSE，FALSE取反为TRUE。

# 逻辑运算符

- 逻辑与 (AND) :
  - AND: 要求两个条件都为真, 结果才为真

```
SQL> SELECT empno, ename, job, sal  
2   FROM emp  
3   WHERE sal >= 1100  
4   AND job = 'CLERK';
```

EMPNO	ENAME	JOB	SAL
7876	ADAMS	CLERK	1100
7934	MILLER	CLERK	1300

# 逻辑运算符

- 逻辑或 (OR) :

- OR: 只需要两个条件中的一个为真, 结果就返回真

```
SQL> SELECT empno, ename, job, sal
2 FROM emp
3 WHERE sal >= 1100
4 OR job = 'CLERK';
```

EMPNO	ENAME	JOB	SAL
7839	KING	PRESIDENT	5000
7698	BLAKE	MANAGER	2850
7782	CLARK	MANAGER	2450
7566	JONES	MANAGER	2975
7654	MARTIN	SALESMAN	1250
...			
7900	JAMES	CLERK	950
...			

14 rows selected.

# 逻辑运算符

- 逻辑非 (NOT)

```
SQL> SELECT ename, job
      2 FROM emp
      3 WHERE job NOT IN ('CLERK', 'MANAGER', 'ANALYST');
```

ENAME	JOB
-----	-----
KING	PRESIDENT
MARTIN	SALESMAN
ALLEN	SALESMAN
TURNER	SALESMAN
WARD	SALESMAN



# 逻辑运算符

- 逻辑非 (NOT)

- NOT运算符还可以和BETWEEN...AND、LIKE、IS NULL一起使用

- ... WHERE deptno NOT IN (10, 20)
    - ... WHERE sal NOT BETWEEN 3000 AND 5000
    - ... WHERE ename NOT LIKE 'D%'
    - ... WHERE mgr IS NOT NULL



# 逻辑运算符

- 运算符的优先级
  - 括号'()' 优先于其他操作符。

优先级	运算分类	运算符举例
1	算术运算符	<code>*</code> , <code>\</code> , <code>+</code> , <code>-</code>
2	连接运算符	<code>  </code>
3	比较运算符	<code>=</code> , <code>&lt;&gt;</code> , <code>&lt;</code> , <code>&gt;</code> , <code>&lt;=</code> , <code>&gt;=</code>
4	特殊比较运算符	<code>BETWEEN..AND..</code> , <code>IN</code> , <code>LIKE</code> , <code>IS NULL</code>
5	逻辑非	<code>NOT</code>
6	逻辑与	<code>AND</code>
7	逻辑或	<code>OR</code>

# 逻辑运算符

- 运算符的优先级

```
SQL> SELECT ename, job, sal
2 FROM emp
3 WHERE job='SALESMAN'
4 OR job='PRESIDENT'
5 AND sal>1500;
```

ENAME	JOB	SAL
KING	PRESIDENT	5000
MARTIN	SALESMAN	1250
ALLEN	SALESMAN	1600
TURNER	SALESMAN	1500
WARD	SALESMAN	1250

# 逻辑运算符

- 运算符的优先级
  - 使用括号强制改变优先权

```
SQL> SELECT      ename, job, sal
  2  FROM        emp
  3  WHERE        (job='SALESMAN'
  4  OR          job='PRESIDENT')
  5  AND          sal>1500;
```

ENAME	JOB	SAL
-----	-----	-----
KING	PRESIDENT	5000
ALLEN	SALESMAN	1600

# 练习9

- 1. 查询工资超过2000并且职位是MANAGER, 或者职位是SALESMAN的员工姓名、职位、工资
- 2. 查询工资超过2000并且职位是 MANAGER或SALESMAN的员工姓名、职位、工资。
- 3. 查询部门在10或者20, 并且工资在3000到5000之间的员工姓名、部门、工资。
- 4. 查询入职日期在81年, 并且职位不是SALES开头的员工姓名、入职日期、职位。
- 5. 查询职位为SALESMAN或MANAGER, 部门编号为10或者20, 姓名包含A的员工姓名、职位、部门编号。

# ORDER BY子句

- 使用ORDER BY子句能对查询结果集进行排序, 语法结构如下:

```
SELECT  [DISTINCT] { * | 列名 |表达式 [别名][,...] }  
FROM    表名  
[WHERE  条件]  
[ORDER BY {列名|表达式|列别名|列序号} [ASC|DESC],...];
```

- 其中:
  - 可以按照列名、表达式、列别名、结果集的列序号排序
  - ASC: 升序, 默认值 DESC: 降序
  - ORDER BY 子句必须写在SELECT语句的最后

# ORDER BY子句

- 排序规则(以升序为例)
  - 数字升序排列小值在前，大值在后。即按照数字大小顺序由小到大排列。
  - 日期升序排列相对较早的日期在前，较晚的日期在后。
  - 字符升序排列按照字母由小到大的顺序排列。即由A-Z排列；中文升序按照字典顺序排列。
  - 空值在升序排列中排在最前面，在降序排列中排在最后。

# ORDER BY子句

- 按列名升序排序

```
SQL> SELECT      ename, job, deptno, hiredate
2  FROM          emp
3  ORDER BY hiredate;
```

ENAME	JOB	DEPTNO	HIREDATE
SMITH	CLERK	20	1980-12-17
ALLEN	SALESMAN	30	1981-02-20
...			

14 rows selected.



# ORDER BY子句

- 按列名降序排序

```
SQL> SELECT      ename, job, deptno, hiredate  
2  FROM          emp  
3  ORDER BY hiredate DESC;
```

ENAME	JOB	DEPTNO	HIREDATE
ADAMS	CLERK	20	1987-05-23
SCOTT	ANALYST	20	1987-04-19
MILLER	CLERK	10	1982-01-23
...			

14 rows selected.

# ORDER BY子句

- 按列别名排序

```
SQL> SELECT      empno, ename, sal*12 annsal  
2  FROM          emp  
3  ORDER BY      annsal;
```

EMPNO	ENAME	ANNSAL
7369	SMITH	9600
7900	JAMES	11400
7876	ADAMS	13200
7654	MARTIN	15000
7521	WARD	15000
7934	MILLER	15600
7844	TURNER	18000

...

14 rows selected.

# ORDER BY子句

- 多列参与排序

```
SQL> SELECT      ename, deptno, sal
      2  FROM      emp
      3  ORDER BY  deptno, sal DESC;
```

ENAME	DEPTNO	SAL
KING	10	5000
CLARK	10	2450
MILLER	10	1300
FORD	20	3000
...		

14 rows selected.

- 参与排序的多列都可以指定升序或者降序
- ORDER BY子句中可以写没在SELECT列表中出现的列

# ORDER BY子句

- 按结果集列序号排序
  - ORDER BY子句后列名可以用数字来代替，这个数字是SELECT语句后列的顺序号。

```
SQL> SELECT      ename, deptno, sal
      2  FROM      emp
      3  ORDER BY  2, 3 DESC;
```

ENAME	DEPTNO	SAL
-----	-----	-----
KING	10	5000
CLARK	10	2450
MILLER	10	1300
FORD	20	3000
...		

14 rows selected.

# 练习10

- 1. 查询部门在20或30的员工姓名，部门编号，并按照工资升序排序。
- 2. 查询工资在2000-3000之间，部门不在10号的员工姓名，部门编号，工资，并按照部门升序，工资降序排序。
- 3. 查询入职日期在82年至83年之间，职位以SALES或者MAN开头的员工姓名，入职日期，职位，并按照入职日期降序排序。

# 限制记录的行数

- 使用select语句时，经常要返回前几条或者中间某几行记录，可以使用关键字limit。语法格式如下：

```
select 字段列表  
from 数据源  
limit [start,]length;
```

- 说明：
  1. limit接受一个或两个整数参数。start表示从第几行记录开始输出，length表示输出的记录行数。
  2. 表中第一行记录的start值为0(不是 1)。

# 限制记录的行数

- 查询员工表的前5条记录

```
SQL> SELECT      ename, deptno, sal
  2  FROM          emp
  3  LIMIT 0,5;
```

ENAME	DEPTNO	SAL
SMITH	20	800
ALLEN	30	1600
WARD	30	1250
JONES	20	2975
MARTIN	30	1250
...		

5 rows selected.

# 练习11

- 1. 查询入职日期最早的前5名员工姓名，入职日期。
- 2. 查询工作在CHICAGO并且入职日期最早的前2名员工姓名，入职日期。
- 3. 按照每页显示5条记录，分别查询第1页，第2页，第3页信息，要求显示员工姓名、入职日期、部门名称。



# 小结

- 结构化查询语言的作用和分类
- 基本SELECT语句的作用
- 空值NULL
- 列别名
- 消除重复行的关键字Distinct
- 限制数据的作用及 WHERE 子句的语法
- 按列名、列别名、顺序号排序及升序降序排序
- 限制记录行数

# 课后作业

1. 查询入职时间在1982-7-9之后，并且不从事SALESMAN工作的员工姓名、入职时间、职位。
2. 查询员工姓名的第三个字母是a的员工姓名。
3. 查询除了10、20号部门以外的员工姓名、部门编号。
4. 查询部门号为30号员工的信息，先按工资降序排序，再按姓名升序排序。
5. 查询没有上级的员工(经理号为空)的员工姓名。
6. 查询工资大于等于4500并且部门为10或者20的员工的姓名\工资、部门编号。

# Neuedu

