数据库学习大纲

什么是数据库
狭义：
存储数据的仓库
广义：
可以对数据进行存储和管理的软件 以及 数据本身 统称为数据库

数据库是由表、 关系、 操作组成

为什么需要数据库
几乎所有的应用软件的后台都需要数据库
数据库存储数据占用空间小容易持久保存
存储比较安全
容易维护和升级
数据库移植比较容易
简化对数据的操作
为将来学习Oracle做准备
B/S架构里面包含数据库

数据库的安装和卸载
sql2000
解决挂起的问题

sql2005
参见视频

预备知识
学习数据库必须的学习数据库原理么
我的视频中会讲一些数据库原理的知识

学习SqlServer 2005必须的先学一门编程语言么
不需要，但是懂一门编程语言的话会有助于学习SqlServer 2005
的TL_SQL

数据结构和数据库的区别
数据库是在应用软件级别研究数据的存储和操作
数据结构是在系统软件级别研究数据的存储和操作

什么是连接【重点】

有了编程语言为什么还需要数据库
对内存数据操作是编程语言的强项，但是对硬盘数据操作却是编程语言的弱项
对硬盘数据操作是数据库的强项，是数据库研究的核心问题

建议初学者从三个方面学习数据库
数据库是如何存储数据的

字段 记录 表 约束(主键 外键 唯一键 非空 check default 触发器)

数据库是如何操作数据的
insert update delete T-SQL 存储过程 函数 触发器

数据库是如何显示数据的
select (重点的重点)

必备的一些操作
如何建数据库

如何删除数据库

如何附加和分离数据库

设置登录用户名和密码

如何创建用户

数据库是如何解决数据存储问题的【最基础内容，必须掌握】
1. 表的相关数据
字段
一个事物的某一个特征

记录
字段的组合 表示的是一个具体的事物

表
记录的组合 表示的是同一类型事物的集合

表和字段、记录的关系
字段是事物的属性
记录是事物本身
表是事物的集合

列
字段的另一种称谓

属性
字段的另一种称谓

元组
记录的另一种称谓

2. create table 命令

通过图形化界面建表
create table 最后一个字段的后面建议不要写逗号
说明：简单掌握 后面我们会再详细的介绍

3. 什么是约束

定义

对一个表中属性操作的限制叫做约束

分类

主键约束

不允许重复元素 避免了数据的冗余

外键约束

通过外键约束从语法上保证了本事物所
关联的其他事物一定是存在的

事物和事物之间的关系是通过外键来体现的

check约束

保证事物属性的取值在合法的范围之内

default约束

保证事物的属性一定会有一个值

唯一约束

保证了事物属性的取值不允许重复，
但允许其中有一列且只能有一列为空
问题：
unique键是否允许多列为空？
答案：
SqlServer2005只允许一个unique列为空
Oracle11G允许多个unique列为空

not null

要求用户必须的为该属性赋一个值,否则语法出错!

如果一个字段不写null 也不行not null
则默认是null 即默认允许为空, 用户可以不给该字段赋值
如果用户没有为该字段赋值，则该字段的值默认是null

要注意null 和 default的区别
相同点：
都允许用户不赋值
不同点：
null修饰的字段如果用户不赋值则默认是null
default修饰的字段如果用户不赋值则默认是default指定的那个值

4. 表和约束的异同

数据库是通过表来解决事物的存储问题的
数据库是通过约束来解决事物取值的有效性和合法性的问题
建表的过程就是指定事物属性及其事物属性各种约束的过程

5. 什么是关系

定义：
表和表之间的联系

实现方式

通过设置不同形式的外键来体现表和表的不同关系

分类(假设是A表和B表)

一对一

既可以把表A的主键充当表B的外键

也可以把表B的主键充当表A的外键

一对多【重点】

把表A的主键充当表B的外键

或者讲：把A表的主键添加到B表来充当B表的外键

在多的一方添加外键

多对多

多对多必须的通过单独的一张表来表示

例子

班级 和 教师

班级是一张表

教师是一张表

班级和教师的关系也是一张表

6. 主键

定义

能够唯一标示一个事物的一个字段或者多个字段的组合，被称为主键

主键的特点【重点】：

含有主键的表叫做主键表

主键通常都是整数 不建议使用字符串当主键(如果主键是用于集群式服务，才可以考虑用字符串当主键)

主键的值通常都不允许修改，除非本记录被删除

主键不要定义成id，而要定义成表名Id或者表名_id

要用代理主键，不要用业务主键

任何一张表，强烈建议不要使用有业务含义的字段充当主键

我们通常都是在表中单独添加一个整型的编号充当主键字段

主键是否连续增长不是十分重要

7. 外键

定义：

如果一个表中的若干个字段是来自另外若干个表的主键或唯一键

则这若干个字段就是外键

注意：

外键通常是来自另外表的主键而不是唯一键，因为唯一键可能为null

外键不一定是来自另外的表，也可能来自本表的主键

含有外键的表叫外建表，外键字段来自的那一张表叫做主键表

问题：

先删主键表还是外建表？

答案： 先删外建表
如果先删主键表，会报错，因为这会导致外建表中的数据引用失败

查询【最重要 难度最大，考试必考内容，强烈建议所有的学生都要熟练掌握查询的内容】

1. 计算列

```
select * from emp;
    -- * 表示所有的
    -- from emp 表示从emp表查询

select empno, ename from emp;

select ename, sal from emp;

select ename, sal*12 as "年薪" from emp;
    --as 可以省略记住："年薪" 不要写成'年薪' 也不要写成年薪

select ename, sal*12 as "年薪", sal "月薪", job from emp;

select 888 from emp;
--ok
--输出的行数是emp表的行数 每行只有一个字段，值是

select 5;  --ok
      --不推荐
```

注意：
在Oracle中字段的别名不允许用单引号括起来
但是SqlServer 2005却允许，
因此为了兼容性 最好字段别名用双引号括起来，不要用单引号

2. distinct【不允许重复的】

```
select distinct deptno from emp;
    --distince deptno 会过滤掉重复的deptno
select distinct comm from emp;
    --distinct也可以过滤掉重复的null  或者说如果有多个null 只输出一个
select distinct comm, deptno from emp;--把comm和deptno的组合进行过滤
select deptno, distinct comm from emp;--error 逻辑上有冲突
```

3. between【在某个范围】

```
--查找工资在1500到3000之间(包括和)的所有的员工的信息
            select * from emp
              where sal>=1500 and sal<=3000
            等价于
            select * from emp
              where sal between 1500 and 3000

            --查找工资小于3000或大于1500的所有的员工的信息
            select * from emp
              where sal<1500 or sal>3000
            等价于
            select * from emp
```

```
where sal not between 1500 and 3000
```

4. in【属于若干个孤立的值】

```
select * from emp where sal in (1500, 3000, 5000)
```

等价于

```
select * from emp
    where sal=1500 or sal=3000 or sal=5000

select * from emp where sal not in (1500, 3000, 5000) --把sal既不是也不是也不是的记录输出
```

等价于

```
select * from emp
    where sal<>1500 and sal<>3000 and sal<>5000
        --数据库中不等于有两种表示： !=    <>   推荐使用第二种
        --对或取反是并且 对并且取反是或
```

5. top【最前面的若干个记录 专属于SqlServer的语法，不可移植到其他数据库】

```
select top 5 * from emp;
select top 15 percent * from emp;  --输出的是3个，不是2个
select top 5  from emp; --error
```

分页查询后面会讲

6. null【没有值 空值】

零和null是不一样的，null表示空值，没有值，零表示一个确定的值

null不能参与如下运算：<> != =

null可以参与如下运算： is not is

```
select * from emp where comm is null; --输出奖金为空的员工的信息
select * from emp where comm is not null; --输出奖金不为空的员工的信息
select * from emp where comm <> null; --输出为空error
select * from emp where comm != null; --输出为空error
select * from emp where comm = null;  --输出为空error
```

任何类型的数据都允许为null

```
create table t1 (name nvarchar(20), cnt int, riqi datetime);
insert into t1 values (null, null, null); --OK
```

任何数字与null参与数学运算的结果永远是null

```
--输出每个员工的姓名年薪(包含了奖金)  comm假设是一年的奖金
select empno, ename, sal*12+comm "年薪" from emp;
    --本程序证明了：null不能参与任何数据运算否则结果永远为空
--正确的写法是：
select ename, sal*12+isnull(comm, 0) "年薪" from emp;
    --isnull(comm, 0) 如果comm是null 就返回零否则返回comm的值
```

in与null的组合使用

后期Oracle讲

7. order by 【以某个字段排序】

```
order by a, b                --a和b都是升序
order by a, b desc           --a升序 b降序
order by a desc, b       --a降序 b升序
order by a desc, b desc      --a和b都是降序
```

文字描述：

如果不指定排序的标准，则默认是升序 升序用asc表示 默认可以不写

为一个字段指定的排序标准并不会对另一个字段产生影响
强烈建议为每一个字段都指定排序的标准

例子：

—asc是升序的意思默认可以不写 desc是降序

```
select * from emp order by sal; —默认是按照升序排序
select * from emp order by deptno, sal;
  --先按照deptno升序排序，如果deptno相同，再按照sal升序排序

select * from emp order by deptno desc, sal;
  --先按deptno降序排序如果deptno相同再按照sal升序排序
  --记住sal是升序不是降序
  --order by a desc, b, c, d    desc只对a产生影响不会对后面的b  c d 产生影响

select * from emp order by deptno, sal desc
  --问题：desc是否会对deptno产生影响?
  --答案：不会
  --先按deptno升序，如果deptno相同，再按sal降序
```

8. 模糊查询 【搜索时经常使用】

格式：

select 字段的集合 from 表名 where 某个字段的名字 like 匹配的条件
匹配的条件通常含有通配符

通配符：

%

表示任意0个或多个字符

```
select * from emp  where ename like '%A%'  --ename只要含有字母A就输出
select * from emp where ename like 'A%'  —ename只要首字母是A的就输出
select *   from emp    where ename like '%A'  —ename只要尾字母是A的就输出
```

_ [这是下划线 不是减号]

表示任意单个字符

```
select * from emp where ename like '_A%' —ename只要第二个字母是A的就输出
```

[a-f]

a到f中的任意单个字符 只能是a b c d e f 中的任意一个字符

```
select * from emp where ename like '_[A-F]%'
    --把ename中第二个字符是A或B或C或D或E或F的记录输出
```

[a, f]

a或f

[^a-c]

不是a 也不是b 也不是c的任意单个字符

```
select * from emp where ename like '_[^A-F]%'
    --把ename中第二个字符不是A也不是B也不是C也不是D也不是E也不是F的记录输出
```

注意：

匹配的条件必须的用单引号括起来 不能省略 也不能改用双引号
通配符作为不同字符使用的问题

预备操作

```
create table student
                (
                    name varchar(20) null
                    ,age int
                );


                insert into student values ('张三', 88);
                insert into student values ('Tom', 66);
                insert into student values ('a_b', 22);
                insert into student values ('c%d', 44);
                insert into student values ('abc_fe', 56);
                insert into student values ('haobin', 25);
                insert into student values ('HaoBin', 88);
                insert into student values ('c%', 66)
                insert into student values ('long''s', 100)
                select * from student;

select * from student where name like '%\%%' escape '\'  --把name中包含有%的输出
select * from student where name like '%\_%' escape '\' --把name中包含有_的输出
```

9. 聚合函数【多行记录返回至一个值 通常用于统计分组的信息】

函数的分类

单行函数

每一行返回一个值

多行函数

多行返回一个值

聚合函数是多行函数

例子：

```
select lower(ename) from emp; -- 最终返回的是行lower()是单行函数
select max(sal) from emp; --返回行max()是多行函数
```

聚合函数的分类

max()

min()

avg() 平均值

count() 求个数

count(*)

返回表中所有的记录的个数

```
select count(*) from emp;  --返回emp表所有记录的个数
```

count(字段名)

返回字段值非空的记录的个数，重复的记录也会被当做有效的记录

```
select count(deptno) from emp;
    --返回值是这说明deptno重复的记录也被当做有效的记录
select count(comm) from emp;
    --返回值是这说明comm为null的记录不会被当做有效的记录
```

count(distinct 字段名)

返回字段不重复并且非空的记录的个数

```
select count(distinct deptno) from emp;
    --返回值是 统计deptno不重复的记录的个数
```

注意的问题

判断如下sql语句是否正确

```
select max(sal), min(sal), count(*) from emp; --ok
select max(sal) "最高工资", min(sal) "最低工资", count(*) "员工人数" from emp; --ok
select max(sal), lower(ename) from emp; --error 单行函数和多行函数不能混用
select max(sal) from emp; --ok 默认把所有的信息当做一组
```

10. group by 【分组 难点】

格式:

group by 字段的集合

功能:

把表中的记录按照字段分成不同的组

例子

查询不同部门的平均工资

```
select deptno, avg(sal) as "部门平均工资"  from emp group by deptno
```

注意:

理解: group by a, b , c的用法

先按a分组，如果a相同，再按b分组，如果b相同，再按c分组

最终统计的是最小分组的信息

一定要明白下列语句为什么是错误的

```
select deptno, avg(sal) as "部门平均工资", ename
    from emp
    group by deptno

select deptno, ename
    from emp
    group by deptno

select deptno , job, sal
    from emp
    group by deptno, job
```

记住：使用了group by 之后select 中只能出现分组后的整体信息，

不能出现组内的详细信息

11. having【对分组之后的信息进行过滤 难点】

1.

having子句是用来对分组之后的数据进行过滤

因此使用having时通常都会先使用group by

2.

如果没使用group by 但使用了having

则意味着having把所有的记录当做一组来进行过滤

极少用

```
select count(*)
    from emp
    having avg(sal) > 1000
```

3.

having子句出现的字段必须的是分组之后的组的整体信息
having子句不允许出现组内的详细信息

4.

尽管select字段中可以出现别名
但是having子句中不能出现字段的别名，只能使用字段最原始的名字
原因不得而知

5 having和where的异同

相同的：

都是对数据过滤，只保留有效的数据
where和having一样，都不允许出现字段的别名，
只允许出现最原始的字段的名字,本结论在SqlServer 2005和Oracle11G都成立

不同：

where是对原始的记录过滤 having是对分组之后的记录过滤
where必须的写在having的前面，顺序不可颠倒 否则运行出错

例子：

```
--把工资大于，
                --统计输出部门平均工资大于的部门的部门编号部门的平均工资
                select deptno, avg(sal) "平均工资", count(*) "部门人数",
                    max(sal) "部门的最高工资"
                    from emp
                    where sal > 2000  --where是对原始的记录过滤
                    group by deptno
                    having avg(sal) > 3000  --对分组之后的记录过滤


                  --判断入选语句是否正确
                  select deptno, avg(sal) "平均工资", count(*) "部门人数",
                        max(sal) "部门的最高工资"
                        from emp
                        group by deptno
                        having avg(sal) > 3000  --对分组之后的记录过滤
                        where sal > 2000  --where写在了having的后面  error
```

12. 连接查询

定义

将两个表或者两个以上的表以一定的连接条件连接起来
从中检索出满足条件的数据

分类

内连接【重点的重点 也是难点的难点】

1. select ... from A, B 的用法

产生的结果：

行数是A和B的乘积
列数是A和B之和

或者说
把A表的每一条记录都和B表的每一条记录组合在一起

形成的是个笛卡尔积

或者说：
把B表的每一条记录都和A表的每一条记录组合在一起
形成的是个笛卡尔积

注意：
select * from A, B
输出结果和
select * from B, A
是一模一样的

例子
--输出70行 11列
```
select * from emp, dept
```

2. select ... from A, B where ... 的用法
select ... from A, B --A和B可以互换
产生的笛卡尔积，用where中的条件进行过滤
例子：
--输出5行 11列
select *
from emp, dept --dept和emp互换 输出结果不变
where empno = 7369

3. select ... from A join B on ... 的用法
select ... from A join B on .. --A和B互换 输出结果不变
4. SQL92标准 和 SQL99标准 的区别
select ... from A, B where ... 是sql92标准
select ... from A join B on ... 是sql99标准
输出结果是一样

推荐使用SQL99标准
1. sql99更容易理解
2. 在sql99标准中，on和where可以做不同的分工
on指定连接条件
where对连接之后临时表的数据进行过滤

示例：
```
--把工资大于2000的员工的姓名和部门的名称输出 和 工资的等级
--sql99标准 明显的优于sql92
select "E".ename, "D".dname, "S".grade
  from emp "E"
  join dept "D"
  on "E".deptno = "D".deptno
  join salgrade "S"
  on "E".sal>= "S".losal and "E".sal <= "S".hisal
  where "E".sal > 2000

--把工资大于2000的员工的姓名和部门的名称输出 和 工资的等级
--sql92标准
select "E".ename, "D".dname, "S".grade
  from emp "E", dept "D", salgrade "S"
  where "E".sal > 2000  and  "E".deptno = "D".deptno and
          ("E".sal>= "S".losal and "E".sal <= "S".hisal)
```

5. select、 from、 where、 join、 on、 group、 order、 top、 having 的混合使用

查询的顺序

```
select top ....
 from A
 join B
 on ....
 join C
 on ....
 where ......
 group by ...
 having .....
 order by .....
 例子：
 --把工资大于的所有的员工按部门分组把部门平均工资大于的最
 --高前个的部门的编号部门的名称部门平均工资的等级
     —第一种写法
     select "T".*, "D".dname, "S".grade
        from dept "D"
        join (
            select top 2 "E".deptno, avg(sal) "avg_sal"
                from emp "E"
                join dept "D"
                on "E".deptno = "D".deptno
                join salgrade "S"
                on "E".sal between "S".losal and "S".hisal
                where "E".sal > 1500
                group by "E".deptno
                having avg("E".sal) > 2000
                order by avg("E".sal) desc
        ) "T"
        on "D".deptno = "T".deptno
        inner join salgrade "S"
        on "T"."avg_sal" between "S".losal and "S".hisal


     —第二种写法
     select "T".*, "D".dname, "S".grade
        from dept "D"
        join (
            select top 2 deptno, avg(sal)  as "avg_sal"
                from emp
                where sal > 1500
                group by deptno
                having avg(sal) > 2000
                order by "avg_sal" desc
        ) "T"
        on "D".deptno = "T".deptno
        join salgrade "S"
        on "T"."avg_sal" between "S".losal and "S".hisal
```

6. 习题

判断以下语句输出是几行

```
select * from emp, dept where emp.deptno = 10
select * from emp, dept where dept.deptno = 10  --过滤条件不是连接条件
```

考虑如何把

```
select * from emp, dept where dept.deptno = 10
```

以sql99标准来输出

1> 求出每个员工的姓名 部门编号 薪水 和 薪水的等级
2> 查找每个部门的编号 该部门所有员工的平均工资 平均工资的等级
3> 查找每个部门的编号 部门名称 该部门所有员工的平均工资 平均工资的等级
4> 求出emp表中所有领导的信息
5> 求出平均薪水最高的部门的编号和部门的平均工资
6> 把工资大于所有员工中工资最低的前3个人的姓名 工资 部门编号 部门名称 工资等级 输出

外连接[参见ppt]

完全连接[参见ppt]

交叉连接[参见ppt]

自连接

定义

一张表自己和自己连接起来查询数据

例子

不准用聚合函数 求薪水最高的员工的信息

联合

定义

表和表之间的数据以纵向的方式连接在一起

注意：我们以前讲的所有的连接是以横向的方式连接在一起的

例子：

输出每个员工的姓名 工资 上司的姓名

```
select "E1".ename, "E1".sal,  "E2".ename "上司的姓名"
        from emp "E1"
        join emp "E2"
        on "E1".mgr = "E2".empno
      union
      select ename, sal, '已是最大老板' from emp where mgr is null
```

注意：

若干个select子句要联合成功的话，必须的满足两个条件

1。 这若干个select子句输出的列数必须是相等的
2. 这若干个select子句输出列的数据类型至少是兼容的

13. 嵌套查询
orale中讲

identity【主键自动增长，用户不需要为identity修饰的主键赋值】
用户如何手动给被identity修饰的主键赋值

不重要
具体解决办法参见ppt

表中删除数据后又插入数据会导致主键不连续递增 怎么办?
主键是否连续增长不十分重要
具体解决办法参见ppt

视图
为什么需要视图
示例
求出平均工资最高的部门的编号和部门的平均工资
总结:
简化查询
避免了代码的冗余
避免了书写大量重复的sql语句

什么是视图
视图从代码上看是一个select语句
视图从逻辑上看被当做一个虚拟表看待

如何创建视图

```
create view 视图的名字
    as
        —select的前面不能添加begin
      select语句
        —select的后面不能添加end
```

注意的问题
创建视图的select语句必须的为所有的计算列指定别名

```
        —error
        create view v$_a
            as
                select avg(sal) from emp;

        —ok
        create view v$_a
            as
                select avg(sal) as "avg_sal" from emp;
```

视图不是物理表，是虚拟表
不建议通过视图更新视图所依附的原始表的数据或结构

视图的优点
简化查询
增加数据的保密性

视图的缺点
增加了数据库维护的成本
视图只是简化了查询，但是并不能加快查询的速度 这也是视图使用不足的地方

事务【重要 参见ppt】

初学者必须要理解的三个概念

事务是用来研究什么的

1。 避免数据处于不合理的中间状态

转账

2. 怎样保证多用户同时访问同一个数据时呈现给用户的数据是合理的

很复杂，现在人类仍然没有设计出很好的解决办法！

事务和线程的关系

事务是通过锁来解决并发访问的

线程同步也是通过锁来解决并发访问的 synchronized

所谓并发访问是指：多用户同时访问同一个数据

事务和第三方插件的关系

直接使用事务库技术难度很大 很多人是借助第三放插件来实现

因此我们一般人不需要细细的研究数据库中事务的语法细节

第三方插件要想完成预期的功能，一般必须的借助数据库中的事物机制来实现

索引

存储过程

游标

TL_SQL

触发器

分页查询

总结

假设每页显示n条记录，当前要显示的是第m页

表名是A 主键是A_id

```
select top n *
    from A
    where A_id not in (select top  (m-1)*n   A_id from emp)
```

郝斌

2010年12月16日