优极限

“极限教育，挑战极限”

www.yjxxt.com

极限教育，挑战极限。优极限是一个让 95%的学生年薪过 18 万的岗前培训公司，让我们的学员具备优秀的互联网技术和职业素养，勇攀高薪，挑战极限。公司位于上海浦东，拥有两大校区，共万余平。累计培训学员超 3 万名。我们的训练营就业平均月薪 19000，最高年薪 50 万。

核心理念：让学员学会学习，拥有解决问题的能力，拿到高薪职场的钥匙。

项目驱动式团队协作、一对一服务、前瞻性思维、教练式培养模型-培养你成为就业明星。首创的老学员项目联盟给学员充分的项目、技术支撑，利用优极限平台这根杠杆，不断挑战极限，勇攀高薪，开挂人生。

扫码关注优极限微信公众号：

（获取最新技术相关资讯及更多源码笔记）

Oracle 数据库

第一章 数据库的介绍

主要内容

- 认识数据库
- 数据库安装

核心目标

- 初步认识数据库
- 数据库安装

1.1 数据库的介绍

1.1.1 数据库

简单地说，数据库（英文 Database）就是一个存放数据的仓库。在日常工作中，常常需要把某些相关的数据放进这样的“**仓库**”，并根据管理的需要进行相应的处理。

例如，企业或事业单位的人事部门常常要把本单位职工的基本情况（职工号、姓名、年龄、性别、籍贯、工资、简历等）存放在表中，这张表就可以看成是一个数据仓库（数据库）。有了这个"数据仓库"我们就可以根据需要随时查询某职工的基本情况，也可以查询工资在某个范围内的职工人数等等。这些工作如果都能在计算机上自动进行，那我们的人事管理就可以达到极高的水平。此外，在财务管理、仓库管理、生产管理中也需要建立众多的这种"数据库"，使其可以利用计算机实现财务、仓库、生产的自动化管理。

生活中的数据库

什么是数据库呢？我们可以从其名字来说，数据库的意思是数据的集合，如果这样来理解的话，在电脑上我们把照片放到同一个文件夹下，那么这个文件夹就是一个照片数据库；把文档资料放到一个文件夹，那么这个文件夹也是一个数据库。这样理解并没错，这种数据库是我们生活中常见的数据库。

计算机中的数据库

数据库（英文 Database）就是一个存放数据的仓库，这个仓库是按照一定的数据结构（数据结构是指数据的组织形式或数据之间的联系）来组织、存储的、我们可以通过数据库提供的多种方法来管理数据库里的数据。更简单的形象理解，数据库和我们生活中存放杂物的仓库性质一样，区别只是存放的东西不同。

数据库的定义和生活中有一定的区别。同样是数据的集合这没有变，但是多了一些条件限定，每一种类型数据集合里面的数据都有固定的内容结构。

1. 数据库中的数据都有一定规律结构，相同类型的数据放在一起，不同类型的数据之间相互隔离
2. 数据库有统一的规则来读写，由 **SQL 语言**专门用来读写数据库，一般都是用程序来读写数据库的内容。

每一个表里面的数据的结构都是一样的，这类似我们常用的 Excel 表格，在标题栏固定后，下面的数据都是按照标题栏的结构来写入的。

数据库现在已经成为**数据管理的重要技术**，也是计算机的重要分支。由于数据库具有数据结构化，最低冗余度、较高的程序与数据独立性，易于扩展、易于编制应用程序等优点，较大的信息系统都是建立在数据库设计之上的。数据库的运用从一般管理扩大到计算机辅助技术、人工智能以及科技计算等领域。

随着数据库技术的发展，计算机技术也随着得到了很大的发展，数据库为我们提供了可以快速存储以及检索的便利，它也为近几年软件可以如此普及贡献不小的力量。

常用的数据库有 DB2 、Oracle、Mysql、SQL Server、SQLite 等。

这些数据库的使用都需要安装相应的软件，启动数据库后我们才可以访问数据库里面的内容。而访问其内容的方式并不是用鼠标直接打开数据库文件查看，一般都是用数据库管理工具或者编写程序来访问数据库。当然，数据库和访问数据库可以不在同一个位置，也就是说数据库在北京，你在深圳写一段代码可以去访问它。

1.1.2 数据库分类

根据存储模型可将数据库划分为关系型数据库和非关系型数据库。关系型数据库，是建立在关系模型基础上的数据库，借助于集合代数等数学概念和方法来处理数据库中的数据。简单来说，关系模型指的就是二维表格模型，而一个关系型数据库就是由二维表及其之间的联系所组成的一个数据组织。标准数据查询语言 **SQL** 就是一种基于关系数据库的语言，这种语言执行对关系数据库中数据的检索和操作。

关系型数据库（表与表之间有关系：体现为主外建）

分类	产品	特点
小型	access、foxbase	负载量小，用户大概 100 人以内 (留言板、信息管理系统)；成本在千元之内 ，对安全性要求不高
中型	sqlservler、mysql	负载量，日访问在 5000~10000 ；成本在万元以内 (商务网站)；满足日常安全需求
大型	sybase、db2、oracle	海量负载，可以处理海量数据(sybase<oracle<db2 海量处理能力)；安全性高，相对贵

1.1.3 DBMS

数据库管理系统(Database Management System)是一种**操纵和管理数据库**的大型软件，用于建立、使用和维护数据库，简称 DBMS。它对数据库进行统一的管理和控制，以保证数据库的安全性和完整性。用户通过 DBMS 访问数据库中的数据，数据库管理员也通过 dbms 进行数据库的维护工作。它可使多个应用程序和用户用不同的方法在同时或不同时刻去建立，修改和询问数据库。大部分 DBMS 提供**数据定义语言** DDL（Data Definition Language）和**数据操作语言** DML（Data Manipulation Language），供用户定义数据库的模式结构与权限约束，实现对数据的追加、删除等操作，以及**数据库控制语言** DCL（Data Control Language），用来设置或更改数据库用户或角色权限的语句。

1.2 数据库的环境搭建

oracle 对 mac 支持不是很好

https://www.oracle.com/technetwork/database/enterprise-edition/downloads/index.html 下载地址

下载需要账号密码

1.3 Oracle 介绍

1.3.1 概念介绍

Oracle Database，又名 Oracle RDBMS，或简称 Oracle。是甲骨文公司的一款关系数据库管理系统。它是在数据库领域一直处于领先地位的产品。可以说 Oracle 数据库系统是目前世界上流行的关系数据库管理系统，系统可移植性好、使用方便、功能强，适用于各类大、中、小、微机环境。它是一种高效率、可靠性好的适应高吞吐量的数据库解决方案。

平常所说的 Oracle 可以指 Oracle 数据库管理系统。Oracle 数据库管理系统是管理数据库访问的计算机软件（Oracle database manager system）。它由 Oracle 数据库和 Oracle 实例（instance）构成（区分 mysql, mysql 没有实例的概念）。

数据库（database）：物理操作系统文件或磁盘的集合。

Oracle 实例：位于物理内存的数据结构，它由操作系统的多个后台进程和一个共享的内存池所组成，共享的内存可以被所有进程访问。Oracle 用它们来管理数据库访问。用户如果要存取数据库（也就是硬盘上的文件）里的数据，必须通过 Oracle 实例才能实现，不能直接读取硬盘上的文件。实际上，Oracle 实例就是平常所说的数据库服务（service）。在任何时刻，一个实例只能与一个数据库关联，访问一个数据库；而同一个数据库可由多个实例访问（RAC）

- 数据库

  Oracle 数据库是数据的物理存储。这就包括（数据文件 ORA 或者 DBF、控制文件、联机日志、参数文件）。其实 Oracle 数据库的概念和其它数据库不一样，这里的数据库是一个操作系统只有一个库。可以看作是 Oracle 就只有一个大数据库。

- 数据库实例

  一个 Oracle 实例（Oracle Instance）有一系列的后台进程（Backguound Processes)和内存结构（Memory Structures)组成。一个数据库可以有 n 个实例。

  一组 Oracle 后台进程/线程以及一个共享内存区，这些内存由同一个计算机上运行的线程/进程所共享。这里可以维护易失的、非持久性内容（有些可以刷新输出到磁盘）。就算没有磁盘存储，数据库实例也能存在。

- 表空间

  表空间是一个用来管理数据存储逻辑概念，表空间只是和数据文件（ORA 或者 DBF 文件）发生关系，数据文件是物理的，一个表空间可以包含多个数据文件，而一个数据文件只能隶属一个表空间。

- 用户

  用户是在实例下建立的。不同实例可以建相同名字的用户。

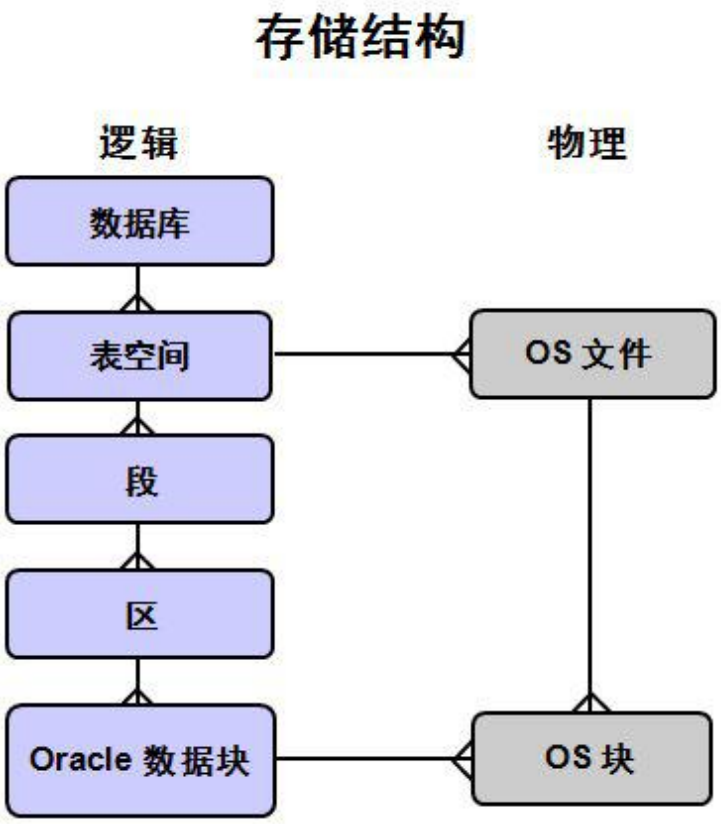
  Oracle 数据库建好后，要想在数据库里建表，必须先为数据库建立用户，并为用户指定表空间。

- 表

  有了数据库，表空间和用户，就可以用自定义的用户在自己的表空间中创建表了。有了表就可以存储数据了。

- 数据文件

  数据文件时数据库的物理存储单位。数据库的数据是存储在表空间中的，真正是在某一个或者多个数据文件中。而一个表空间可以由一个或多个数据文件组成，一个数据文件只能属于一个表空间。一旦数据文件被加入到某个表空间后，就不能删除这个文件，如果要删除某个数据文件，指有删除其所属的表空间才行。



注意：

表的数据，是由用户放入某一个表空间的，而这个表空间会随机把这些表数据放到一个或者多个数据文件中。由于 oracle 的数据库不是普通的概念，oracle 是由用户和表空间对数据进行管理和存放的。但是表不是由表空间去查询的，而是由用户去查的。因为不同用户可以在同一个表空间建立同一个名字的表！这里的区分就是用户了！！！！

1.3.2 创建用户和表空间

- 以超级管理员身份登录
- 创建表空间
- 创建用户
- 给用户授权
- 查询测试

```
sqlplus /nolog;                              // 启动 sqlplus 不登录
conn sys/root as sysdba;// 通过超级管理员以 dba 的身份登录
create tablespace 表空间名 datafile '文件路径\文件名.dbf' size 空间大小;
      //创建表空间
create user 用户名 identified by 密码 default tablespace 表空间;     //创建
用户并指定表空间
grant dba to 用户;    // 给用户授予 dba 的权限
```

第二章 表设计

第一节

表的概念

表设计的原则

2.1 表

数据库中以表为组织单位存储数据。表用来存储一些事物的信息，首先需要有一个表名，以及存储的信息。

2.2 设计原则

好的数据库表设计会影响数据库操作效率。特别是数据特别多的时候，如果表结构不好的话，操作的时候条件会变得非常复杂。所以为了以后操作简单，表的关系（内部和外部）要尽量合理的设计。

步骤：

1）找出表要描述的东西

2）列出你想通过这个表得到的相关信息的列表

3）通过上面的信息列表，将信息划分成一块块的小部分，通过这个小块来建表中的字段；

所以为了建立冗余较小、结构合理的数据库，设计数据库时必须遵循一定的规则。在关系型数据库中这种规则就称为范式。范式是符合某一种设计要求的总结。要想设计一个结构合理的关系型数据库，必须满足一定的范式。在实际开发中最为常见的设计范式有三个：

第一范式（确保每列保持原子性），属性不可再分

第二范式（确保表中的每列都和主键相关），实体唯一

第三范式（确保每列都和主键列值直接相关，而不是间接相关），字段没有冗余

基本表及其字段之间的关系，应尽量满足第三范式。但是，满足第三范式的数据库设计，往往不是最好的设计。为了提高数据库的运行效率，常常需要降低范式标准：适当增加冗余，达到以空间换时间的目的。

更合理的表设计会给每条记录加上一个唯一的识别，就是加上**主键**。 1）将一个表字段设为主键要求在表创建的时候就进行设置。 2) 一个表里被设为主键的字段的值必须是唯一的，也就是说如果一个字段被设为主键，这个表所有的数据列表里这个字段的值不可能有重复的。 3) 被设为主键的字段不能插入空值。 4) 被设为主键的字段的值是不能更改的。 5) 如果字段被设为是自增长的,主键只能设置一个且它必须是主键。如果表中没有自增长的字段,则可以设多个字段为主键. 6) 主键最好是一个和表里数据无关的值。比如说另建一个字段：id;而不要设在：name等这些字段上。

两个表关联.两个表之间数据的关系有三种: 1)一对一;两个表里数据唯一对应; 2)一对多;表 A 在表 B 里对应多条数据,但表 B 里的一条数据绝对只对就 A 中的一条数据; 3)多对多;A 里的一条数据对应 B 里的多条数据,B 里一条数据也对应 A 中的多条数据.

一对一 的表设计用的不多.可能用到的情况有: a)对一个表中大多数时候不查的字段,放到另一个表中对应起来.这样可以提高大多数时候查询的效率; b)若表中记录还有些字段的值未知,可以将这些字段分出来放.这样可以让主表中不存在 NULL; c)不想轻易就查出来的数据,比如一个人的工资详情,等.可以在主另一表中放着; d)大文本,通过一个外键关联,这样可以提高查询效率;

一对多 的情况可以如下: 有一个人员信息表 info,里面包括一个外键:email;这个字段里存的是邮箱表 emailBox 里的主键:id;因为一个人可以对应多个邮箱,但一个邮箱只能属于一个人(他自己要共用木有办法)

多对多 对优化表设计的用处最大,效果最显著;一个多对多的关系是由一个连接表有两个一对多的表关系组成的;

另外,同一个表里的各字段之间不要有复杂的**依赖关系**.各字段只能和主键有依赖关系.如果非主键和非主键间有依赖关系,就要将它们从主表分离出去,放在另一个表中,并通过外键进行关联

2.3 约束

对于存储的每一个信息，都应该使用相应的数据进行表示，并且这些数据应该是合法的（包括实际和业务逻辑），在数据库中我们通过约束来对每个字段中的数据的合法性进行规范。

- **主键约束（PRIMARY KEY）**
- **唯一性约束（UNIQUE)**
- **非空约束（NOT NULL)**
- **外键约束（FOREIGN KEY)**
- **检查约束（CHECK)**

主键

主键是定位表中单个行的方式，可唯一确定表中的某一行，关系型数据库要求所有表都应该有主键，不过 Oracle 没有遵循此范例要求，Oracle 中的表可以没有主键（这种情况不多见）。关于主键有几个需要注意的点：

1. 键列必须必须具有唯一性，且不能为空，其实主键约束 相当于 UNIQUE+NOT NULL
2. 一个表只允许有一个主键

3. 主键所在列必须具有索引（主键的唯一约束通过索引来实现），如果不存在，将会在索引添加的时候自动创建

唯一键

唯一性约束可作用在单列或多列上，对于这些列或列组合，唯一性约束保证每一行的唯一性。 UNIQUE 允许 null 值，UNIQUE 约束的列可存在多个 null。

非空约束

非空约束作用的列也叫强制列。顾名思义，强制键列中必须有值，当然建表时候若使用 default 关键字指定了默认值，则可不输入。

外键约束

外键约束定义在具有父子关系的子表中，外键约束使得子表中的列对应父表的主键列，用以维护数据库的完整性。不过出于性能和后期的业务系统的扩展的考虑，很多时候，外键约束仅出现在数据库的设计中，实际会放在业务程序中进行处理。外键约束注意以下几点：

1. 外键约束的子表中的列和对应父表中的列数据类型必须相同，列名可以不同
2. 对应的父表列必须存在主键约束（PRIMARY KEY）或唯一约束（UNIQUE）
3. 外键约束列允许 NULL 值，对应的行就成了孤行了

检查约束

检查约束可用来实施一些简单的规则，比如列值必须在某个范围内。

2.4 客户端的安装使用

2.5 准备表和数据

2.6 SQL 语言介绍

SQL(Structured Query Language)为数据库的语言，在 1974 年由 Boyce【博伊斯】和 Chamberlin【钱伯林】提出的一种介于关系代数与关系演算之间的结构化查询语言，是一个通用的、功能极强的关系型数据库语言。它包含三部分:

.DDL(Data Definition Languages)语句：**数据定义语言**，这些语句定义了不同的数据段、数据库、表、列、索引等数据库对象。常用的语句关键字主要包括 create、drop、alter 等。

.DML(Data Manipulation Languages)语句：**数据操纵语句**，用于添加、删除、更新和查询数据库记录，并检查数据完整性。常用的语句关键字主要包括 insert、delete、update 和 select 等。

.DCL（Data Control Language）语句：**数据控制语句**，用于控制不同数据段直接的许可和访问级别的语句，这些语句定义了数据库、表、字段、用户的访问权限和安全级别，主要的语句关键字包括 grant、revoke 等。

分类	命令
DDL	create:创建；drop:删除；alter:修改；rename:重命名；truncate:截断
DML	insert:插入；delete:删除；update:更新；select:查询
DCL	grant:授权；revoke:回收权利；commit:提交事务；rollback:回滚事务

Oracle 命令不区分大小写，但是数据|内容 是区分大小写的

第三章 select 语法

3.1 select 基本结构和简单查询

3.1.1 select 结构

结构：

```
select  *|colname [,...] from table [alias]
```

分析：

select 关键字 后面跟要查询的内容 from 关键字后面跟数据的来源

解析步骤：1)from 找来源 2) select 挑数据

语法结构需记忆

3.1.2 简单查询

1）查询所有

查询员工的所有信息

select * from emp

查询部门的所有信息

select * from dept

查询工资等级信息

select * from salgrade

查询班里同学的信息

select * from stu

2）查询部分字段

查询员工的姓名（我要看看公司有些什么人）

查询员工的姓名和年龄

查询员工的年龄和工种以及工资

查询学生的姓名和年龄和性别

3）按顺序查询

查询部门的所有信息并且按照，部门编号，地址，部门名称的顺序来显示

3.2 去重、别名和排序

3.2.1 去重

去除重复记录

结构：select distinct colName from tableName

查询工种名称

查询有员工所在的部门名称

3.2.2 列别名

给列取名字

结构：

```
select colName n from tableName ;
```

```
select colName as n from tableName;
```

查询员工的名称并以“姓名”作为字段名显示

查员工资信息中的工资金额和对应的等级以“金额”和“等级”作为字段名称显示

查员工资信息中的工资金额和对应的等级以“金 额”和“等 级”作为字段名称显示

3.2.3 排序

将查询出来的结果按照指定顺序排序

结构：

```
select colName1, colName2 from tableName order by colName;
```

查询员工信息并按照工资升序排序

查询员工信息并按照工资降序排序

查询员工姓名，工种，所属部门编号以及工资并且按照所属部门编号进升序排序，当所属部门相同时按照工资升序排序

查询员工姓名，工种，所属部门编号以及工资并且按照所属部门编号进升序排序，当所属部门相同时按照工资降序排序

查询员工姓名的奖金并按奖金升序排序（没有奖金的 null 放在最前）

查询员工姓名的奖金并按奖金升序排序（没有奖金的 null 放在最后）

3.3 伪列和虚表

3.3.1 伪列和表达式

说明：查询不存在的列即伪列，当需要的结果不能直接从表中得到需要经过计算来展示则可以使用伪列+表达式实现

```
select 1  from emp;
select ename, 1 from emp;
```

查询员工的名称，工种，月工资，以及年薪（年薪=12 个月的工资）

查询每个工资等级的平均工资（平均工资=（最低工资+最高工资）/2）

1）null 处理

nvl()

查询员工的名称，工种，工资，奖金，以及月收入（月收入=工资+奖金）

2）字符串拼接

||

查询员工的员工编号，姓名，以及将员工姓名和工种进行拼接后的字段（员工姓名-工种）

3.3.2 虚表

dual 是一个虚表，虚拟表，是用来构成 select 的语法规则，oracle 保证 dual 里面永远只有一条记录。该表只有一行一列，它和其他表一样，可以执行插入、更新、删除操作，还可以执行 drop 操作。但是不要去执行 drop 表的操作，否则会使系统不能用，起不了数据库。

dual 主要用来选择系统变量或是求一个表达式的值。如果我们不需要从具体的表来取得表中数据，而是单纯地味了得到一些我们想得到的信息，并要通过 select 完成时，就要借助一个对象，这个对象就是 dual。

计算 999*666

获取系统当前时间

3.4 条件查询

当我们查询的数据需要经过筛选时，我们会给出一些条件，只有当表中的记录满足我们所给的条件，才会成为我们的目标数据，这就需要借助我们的条件查询。

```
select 查询内容 from 数据来源 where 行记录条件
```

3.4.1 比较运算

= 、>、<、>=、<=、<> 、!=、^=、between and 、in

查询 10 部门的员工信息

查询 10 部门以外的员工信息

查询工资在 2000 以上的员工名称，工种，所属部门编号

查询工资在 1800 到 2500 之间到员工信息

查询 10、30、40 部门的员工信息

3.4.2 条件连接运算

and 、or 、not

当查询条件有多个时，可能需要同时满足，或者只满足其中一个，或者不满足某个条件，则需要用到 或。且。非

查询工资在 1500 以上并且是 20 部门的员工信息

查询岗位为 'CLERK' 或部门编号为 20 的员工名称，部门编号和工资

查询岗位不是'SALES'的员工名称，部门编号和工资

3.4.3 null 运算 nvl()

null 比较特殊需要单独处理

is null 、is not null、not ... is null

查询所有有可能拿奖金的员工信息

查询所有不可能拿奖金的员工信息

查询所有真实获得奖金的员工信息

3.3.4 模糊查询 like

% 、 _

当查询的条件为比较模糊时，可以使用模糊查询

查询名称以'A'开头的员工姓名，以及工资

查询名称第二个字母为'M'的员工姓名以及工资

查询名称中含有'C'的员工姓名以及工资

查询名称中含有%的员工信息 escape()

查询名称中含有_的员工信息

3.3.5 exists

in 是把外表和那表作 hash join，而 exists 是对外表作 loop，每次 loop 再对那表进行查询。这样的话，in 适合内外表都很大的情况，exists 适合外表结果集很小的情况。

3.5 子查询

3.5.1 from 子句...没定

数据来源是经过过滤的

FROM 子句指定 SELECT 语句查询及与查询相关的表或视图。在 FROM 子句中最多可指定 256 个表或视图， 它们之间用逗号分隔。 在 FROM 子句同时指定多个表或视图时，如果选择列表中存在同名列，这时应使用对象名限定这些列 所属的表或视图。

select * from (select)

3.5.2 where 子句

判断条件不能直接得到，需要经过计算和过滤的

查询'XXX'部门的员工信息

select * from emp where deptno=(select deptno from dept where dname='XXX')

查询所有工资在第三等级的员工姓名，工资，所属部门

查询练习

第四章 函数

4.1 函数介绍

Oracle SQL 提供了用于执行特定操作的专用函数，这些函数大大增强了 SQL 语言的功能。函数可以接受零个或者多个输入参数，并且返回一个输出结果。这种由 Oracle 提供给我们的函数我们称为内置函数，除了有内置函数外还可以自定义函数。

根据函数的返回结果，我们将函数分为**单行函数**和**多行函数**

1、单行函数:对应在表记录时，一条记录返回一个结果。例如 lower（x），将参数转换为小写

2、多行函数:也称 组函数 或 聚合函数 (重点) :此类函数可同时对多条记录进行操作，并返回一个结果 (重点) 。 例如 max(x) 求最大值。

4.2 常用单行函数

4.2.1 字符函数

concat(x,y) 连接字符串 x 和 y

instr(x,str,start,n). 在 x 中查找 str，可以指定从 start 开始，也可以指定从第 n 次开始

length(x) 返回 x 的长度

lower(x) x 转换为小写

upper(x) x 转换为大写

ltrim(x,trim*str)* *把 x 左边截去 trim*str 字符串，缺省截去空格

rtrim(x,trim*str)* *把 x 右边截去 trim*str 字符串，缺省截去空格

replace(x, old,new) 在 x 中查找 old，并替换为 new

substr(x, start,length) 返回 x 的字符窜，从 start 处开始，截取 length 个字符，缺省 length，默认到结尾

4.2.2 数学函数

abs(x) x 的绝对值

ceil(x) 向上取整

floor(x) 向下取整

mod(x,y) 对 x 求 y 的余数

poer(x,y) x 的 y 次幂

sqrt(x) x 的平方根

4.2.3 日期函数

sysdate 当前系统时间

current_date 返回当前系统日期

add_months(d1,n1) 返回在日期 d1 基础上再加 n1 个月后新的日期

last_day(d1) 返回日期 d1 所在月份最后一天的日期

months_between(d1,d2) 返回日期 d1 到日期 d2 之间的月数

next_day(d1[,c1]) 返回日期 d1 在下周，星期几(参数 c1)的日期

4.2.4 转换函数

to_char(x,c) 将日期或数据 x 按照 c 的格式转换为 char 数据类型

to_date(x,c) 将字符串 x 按照 c 的格式转换为日期

to_number(x) 将字符串 x 转化为数字型

4.3 常用组函数

组函数同时对多条记录进行操作，并返回一个结果

avg() 平均值

sum() 求和

min() 最小值

max() 最大值

count() 统计

注意：null 不参与运算

4.4 group by 分组

group by 分组，将满足条件的记录进一步按照某特性进行分组。提取每组记录中的共性

结构：select ... from ...where ...group by ...

按部门分组，求平均工资（查询每个部门的平均工资）

按工种分组，统计人数（查询每个工种的员工数）

4.5 having 过滤组信息

要获取的组信息也许要满足一定条件时，我们通过 hving 来过滤组的条件。

结构：select ... from ...where ...group by ... having...

4.6 过滤行记录和组信息

行记录的过滤是针对每条记录的筛选，组信息的过滤是针对组的筛选，是可以同时出现的，先筛选行，再过滤组。

where 筛选行，只能出现行信息

having 过滤组，只能出现组信息

结构：select ... from ...where ...group by ... having...

执行顺序

from

where

group by

having

select

第五章 分页和去重

5.1 分页介绍

为什么会用到分页呢，因为列表内容太多了，所以使用分页进行显示。数据过多单页面无法显示所有内容，则每一次只显示一部分的数据。

分页,是一种将所有数据分段展示给用户的技术。用户每次看到的不是全部数据,而是其中的一部分,如果在其中没有找到自习自己想要的内容,用户可以通过制定页码

或是翻页的方式转换可见内容,直到找到自己想要的内容为止。其实这和我们阅读书籍很类似。

实现分页的解决方案有两种：

（1）一次查询出数据库中的所有记录，然后在每页中显示指定的记录。

（2）对数据库进行多次查询，每次只获得本页的数据并显示

如今网站建设中的数据都是海量的，若按方案 1 执行：无疑会加大服务器内存的负载，降低系统运行速度；若使用方案 2 执行，则可能回频繁操作数据库，也会影响响应效率；因而大家都会使用方案 1+方案 2 来实现。

分页的核心就是计算每页多少记录和总页数以及第几页。每一页的数据则只需计算起始的记录和结束记录即可。

5.1.1 rownum

rownum 不是一个真实存在的列，它是用于从查询返回的行的编号，返回的第一行分配的是 1，第二行是 2，依此类推，这个伪字段可以用于限制查询返回的总行数。

由于 rownum 总是结果集的编号，所以无法直接查询 rownum>1 的任何记录，因为总是从 1 开始的嘛。

5.1.2 oracle 分页实现

虽然 ronum 不能直接查询大于 1 的记录，但是我们可以自己添加伪列，将查询的结果集中的 rownum 作为查询的来源，则此时来源中的 rownum 变成了普通字段，再通过这个 rownum 来进行某段记录的选取即可。

5.2 去除重复记录

5.2.1 rowid 介绍

ROWID 是 ORACLE 中的一个重要的概念。用于定位数据库中一条记录的一个相对唯一地址值。通常情况下，该值在该行数据插入到数据库表时即被确定且唯一。ROWID 它是一个伪列，它并不实际存在于表中。它是 ORACLE 在读取表中数据行时，根据每一行数据的物理地址信息编码而成的一个伪列。所以根据一行数据的 ROWID 能找到一行数据的物理地址信息。从而快速地定位到数据行。数据库的大多数操作都是通过 ROWID 来完成的，而且使用 ROWID 来进行单记录定位速度是最快的。

有时繁杂的数据检索时，普通检索条件不能达到要求，可以利用 rowid 来精确检索的结果

oracle 中如果要查询某张表中多个字段，又只对某个字段去重的时候用 distinct 或者 group by 都不行。distinct 和 group by 会对要查询的字段一起进行去重，也就是当查询的所有字段都相同，oracle 才认为是重复的。这时用 rowid 是个不错的选择。

5.2.2 重复记录的查找

题目场景：当我们表里面出现了许多重复记录时，我们需要将重复的记录找出来

实现步骤：1、按照重复内容分组 2、取出每一组中的一条并记录保留（注意具有唯一性）3、删除未在保留范围的数据

第六章 表连接

6.1 表连接介绍

当我们获取的数据不是来自于同一张表而是来自于多张表时就需要使用到表连接。表连接就是一个表的行根据指定的条件跟另一个表的行连接起来形成新的行的过程。

简单来讲，我们将数据存在不同的表中，而不同的表有着它们自身的表结构，不同表之间可以是有关联的，大部分实际使用中，不会仅仅只需要一张表的信息，比如需要从一个班级表中找出北京地区的学生，再用这个信息去检索成绩表中他们的数学成绩，如果没有多表连接，那只能手动将第一个表的信息查询出来作为第二个表的检索信息去查询最终的结果，可想而知这将会是多么繁琐。

连接查询：

1) 即查询的时候同时需要多张表（特别是存在外键关系的），此时需要多张表之间的值进行连接；

2) 目前 SQL 标准提出过两种连接查询，第一种是较早的 SQL92 标准，第二种是目前使用广泛的较新的 SQL99 标准；

3) 92 形式简单，但编写较为冗长，99 不仅在底层得到优化，而且形式看上去更加一目了然，逻辑性更强，一般建议使用 99 标准；

6.2 92 语法

多张表需要全部放在 from 之后，所有的连接条件都放在 where 当中，因此 SQL92 中的等值连接、非等值链接、外连接等等其实只是 where 条件的筛选

结构：select ... from table1, table2 ,table3 ... where...

很多时候需要为表取别名（1、简化表名 2、可能存在自连接的情况）

连接的原理：按照 from 后面表的出现顺序，前面的表作为内存的 for 循环，后出现的表作为外层的 for 循环

6.2.1 笛卡尔积

通过线性代数的人都知道，笛卡尔乘积通俗的说，就是两个集合中的每一个成员，都与对方集合中的任意一个成员有关联。

例如有个考勤记录表，记录着 100 个人的 2011 年 4 月的考勤信息，理论上这些人应该每天都有记录的。 但是实际上某些人在某些天上面的数据，缺少了。 一天一天的查询，还是一个人一个人的查询，都有些麻烦。 这种情况下，可以针对 每个人 与 每一天 做一个 笛卡尔积 的处理。 然后与实际的表去关联。就很容易查询出结果了。

select ... from t1,t2

6.2.2 等值连接

在笛卡尔积的基础上取条件列相同的值

查询员工信息和部门信息

查询员工姓名，部门名称

6.2.3 非等值连接

!=、>、<、<>、between and

查询员工姓名，工资及等级

6.2.4 自连接

特殊的等值连接（来自于同一张表）

找出存在上级的员工姓名

6.2.5 外连接

内联接使用比较运算符根据每个表共有的列的值匹配两个表中的行。

外联接可以是左向外联接、右向外联接或完整外部联接。

左向外联接的结果集包括 LEFT OUTER 子句中指定的左表的所有行，而不仅仅是联接列所匹配的行。如果左表的某行在右表中没有匹配行，则在相关联的结果集行中右表的所有选择列表列均为空值。

右向外联接是左向外联接的反向联接。将返回右表的所有行。如果右表的某行在左表中没有匹配行，则将为左表返回空值。

看‘+’，带‘+’的表为从表，对立面的表为主表。

找出所有员工的姓名 以及他上级的名称

找出所有上级的名称以及其手下员工的名称

找出所有部门的员工数 及部门名称

找出所有有员工的部门名称及员工数

6.3 99 语法

6.3.1 cross join

交叉连接，实现笛卡尔积

6.3.2 natural join

需要有（同名列、主外建）

自然连接，做等值连接

查询所有员工姓名及所在部门的名称

6.3.3 join using （同名列）

using 连接 ，等值连接

查询所有员工姓名及所在部门的名称

6.3.4 join on

on 连接，可做等值连接、非等值连接、自连接，可以解决一切连接，关系列必须要区分

查询所有员工姓名及所在部门的名称

查询所有员工的姓名，工资以及工资等级

6.3.5 outer join

外连接，有主表和从表一说

left [outer] join on

left [outer] join using

right [outer] join on

right [outer] join using

6.3.6 full join on| using

全连接，满足则直接匹配，不满足的相互补充 null，确保所有的表记录都至少出现一次。

6.4 集合操作

Union、Union All、Intersect、Minus

Union，并集(去重) 对两个结果集进行并集操作，不包括重复行同时进行默认规则的排序；

Union All，全集(不去重) 对两个结果集进行并集操作，包括重复行，不进行排序 ；

Intersect，交集(找出重复) 对两个结果集进行交集操作，不包括重复行，同时进行默认规则的排序；

Minus，差集(减去重复) 对两个结果集进行差操作，不包括重复行，同时进行默认规则的排序

https://blog.csdn.net/GreenHandCGL/article/details/50170743 oracle 练习题

第七章 DDL 语法

7.1 DDL 介绍

SQL 语句主要可以划分为以下 3 个类别

.DDL(Data Definition Languages)语句：数据定义语言，这些语句定义了不同的数据段、数据库、表、列、索引等数据库对象。常用的语句关键字主要包括 create、drop、alter 等。

.DML(Data Manipulation Languages)语句：数据操纵语句，用于添加、删除、更新和查询数据库记录，并检查数据完整性。常用的语句关键字主要包括 insert、delete、update 和 select 等。

.DCL（Data Control Language）语句：数据控制语句，用于控制不同数据段直接的许可和访问级别的语句，这些语句定义了数据库、表、字段、用户的访问权限和安全级别，主要的语句关键字包括 grant、revoke 等。

DDL 是数据定于语言的缩写，简单来说，就是对数据库内部的对象进行创建、删除、修改等操作的语言。它和 DML 语言的最大区别是 DML 只是对表内部数据操作，而不涉及表的定义，结构的修改，更不会涉及其他对象。DDL 语句更多的由数据库管理员（DBA）使用，开发人员一般很少使用。

DDL（data definition language）：DDL 比 DML 要多，主要的命令有 CREATE、ALTER、DROP 等，DDL 主要是用在定义或改变表（TABLE）的结构，数据类型，表之间的链接和约束等初始化工作上，他们大多在建立表时使用。

DDL（Data Definition Language，数据定义语言）： 用于定义数据的结构，比如 创建、修改或者删除数据库对象。 DDL 包括：DDL 语句可以用于创建用户和重建数据库对象。下面是 DDL 命令： CREATE TABLE：创建表 ALTER TABLE DROP TABLE：删除表 CREATE INDEX DROP INDEX

7.1 表操作

7.1.1 创建表

1）创建新表

```
create table 表名(
  字段名 类型(长度),
  ...其他字段...
);
```

2）从其他表拷贝结构

create table 表名 as select 字段列表 from 已有表 where 1!=1;

7.1.2 修改表结构

1）修改表名

rename 原表名 to 新表名

```
rename tb_txt to tb_txt_new;
```

2）修改列名

alter table 表名 rename column 列名 to

```
alter tb_txt_new rename column txtid to tid;
```

3）修改字段类型

alter table 表名 modify(字段 类型)

```
alter table tb_txt_new modify(tid vachar2(20));
```

4）添加列

alter table 表名 add 字段 类型

```
alter table tb_txt_new add col_test_name varchar2(30);
```

5）删除列

alter table 表名 drop column 字段

```
alter table tb_txt_new drop column col_test_name;
```

7.1.3 删除表

```
drop table ;
```

7.2 约束

约束是数据库用来确保数据满足业务规则的手段，不过在真正的企业开发中，除了主键约束这类具有强需求的约束，像外键约束，检查约束更多时候仅仅出现在数据库设计阶段，真实环境却很少应用，更多是放到程序逻辑中去进行处理。这也比较容易理解，约束会一定程度上较低数据库性能，有些规则直接在程序逻辑中处理就可以了，同时，也有可能在面对业务变更或是系统扩展时，数据库约束会使得处理不够方便。不过在我看来，数据库约束是保证数据准确性的最后一道防线，对于设计合理的系统，处于性能考虑数据库约束自然可有可无；不过若是面对关联关系较为复杂的系统，且对系统而言，数据的准确性完整性要高于性能要求，那么这些

约束还是有必要的（否则，就会出现各种相对业务规则来说莫名其妙的脏数据，本人可是深有体会的。。）。总之，对于约束的选择无所谓合不合理，需要根据业务系统对于准确性和性能要求的侧重度来决定。

数据库约束有五种：

- **主键约束（PRIMARY KEY）**
- **唯一性约束（UNIQUE)**
- **非空约束（NOT NULL)**
- **外键约束（FOREIGN KEY)**
- **检查约束（CHECK)**

1）主键约束（PRIMARY KEY)

主键是定位表中单个行的方式，可唯一确定表中的某一行，关系型数据库要求所有表都应该有主键，不过 Oracle 没有遵循此范例要求，Oracle 中的表可以没有主键（这种情况不多见）。关于主键有几个需要注意的点：

1.键　列必须必须具有唯一性，且不能为空，其实主键约束 相当于 UNIQUE+NOT NULL

2.一　个表只允许有一个主键

3.主　键所在列必须具有索引（主键的唯一约束通过索引来实现），如果不存在，将会在索引添加的时候自动创建

2）唯一性约束（UNIQUE)

唯一性约束可作用在单列或多列上，对于这些列或列组合，唯一性约束保证每一行的唯一性。

UNIQUE 需要注意：

1.对　于 UNIQUE 约束来讲，索引是必须的。如果不存在，就自动创建一个（UNIQUE 的唯一性本质上是通过索引来保证的）

2.U　NIQUE 允许 null 值，UNIQUE 约束的列可存在多个 null。这是因为，Unique 唯一性通过 btree 索引来实现，而 btree 索引中不包含 null。当然，这也造成了在 where 语句中用 null 值进行过滤会造成全表扫描。

3）非空约束（NOT NULL)

非空约束作用的列也叫强制列。顾名思义，强制键列中必须有值，当然建表时候若使用 default 关键字指定了默认值，则可不输入。

4）外键约束（FOREIGN KEY）

外键约束定义在具有父子关系的子表中，外键约束使得子表中的列对应父表的主键列，用以维护数据库的完整性。不过出于性能和后期的业务系统的扩展的考虑，很多时候，外键约束仅出现在数据库的设计中，实际会放在业务程序中进行处理。外键约束注意以下几点：

1. 外键约束的子表中的列和对应父表中的列数据类型必须相同，列名可以不同
2. 对应的父表列必须存在主键约束（PRIMARY KEY）或唯一约束（UNIQUE）
3. 外键约束列允许 NULL 值，对应的行就成了孤行了

其实很多时候不使用外键，很多人认为会让删除操作比较麻烦，比如要删除父表中的某条数据，但某个子表中又有对该条数据的引用，这时就会导致删除失败。我们有两种方式来优化这种场景：

第一种方式简单粗暴，删除的时候，级联删除掉子表中的所有匹配行，在创建外键时，通过 **on delete cascade** 子句指定该外键列可级联删除：

第二种方式，删除父表中的对应行，会将对应子表中的所有匹配行的外键约束列置为 NULL，通过 **on delete set null** 子句实施

第三种方式，默认，强制不让删

5）检查约束（CHECK)

检查约束可用来实施一些简单的规则，比如列值必须在某个范围内。检查的规则必须是一个结果为 true 或 false 的表达式，

7.2.1 创建表和约束

表名：tb_user（用户表）

编号	字段名	字段类型	说明
1	userid	number(5)	用户 id,主键
2	username	varchar2(30)	用户名，非空，4～20 个字符
3	userpwd	varchar2(20)	密码，非空，4～18 个字符
4	age	number(3)	年龄，默认 18，值大于等于 18
5	gender	char(3)	性别，默认‘男’，只能是‘男’或‘女’
6	email	varchar2(30)	邮箱，唯一
7	regtime	date	注册日期，默认当前日期

表名：tb_txt（文章表）

编号	字段名	字段类型	说明
1	txtid	number(5)	文章编号，主键
2	title	varchar2(32)	文章标题，非空，长度为 4～20 字符
3	txt	varchar2(1024)	内容，最大长度 1024
4	pubtime	date	发布日期，默认当前日期
5	userid	number(5)	作者，外键，参考用户表的用户 id，删除时，自设为 null

1）约束没有名字

```
create tble tb_user(
   userid number(5) primary key,
   username varchar2(30) check(length(username) between 4 and 20) not nu
ll ,
   userpwd varchar2(20) not null  check(length(userpwd) between 4 and 1
8),
   age  number(3) default(18) check(age>=18),
   gender char(3) default('男') check(gender in('男','女')),
   email  varchar2(30) unique,
   regtime date default(sysdate)
);



create table tb_txt(
   txtid number(10) primary key,
   title varchar2(32) not null check(length(title)>=4 and length(title)<
=30),
   txt varchar2(1024),
   pubtime date default(sysdate),
   userid number(5)  references tb_user(userid) on delete set null);
```

2）带名字的约束

```
create table  tb_user(
      userid number(5),
      username varchar2(30)  constraint nn_user_name not null ,
      userpwd varchar2(20)  constraint nn_user_pwd not null  ,
      age  number(3) default(18) ,
      gender char(3) default('男'),
      email  varchar2(30),
      regtime date default(sysdate),
      constraint pk_user_id  primary key (userid),
```

```
	constraint ck_user_name check(length(username)between 4 and 20) ,
	constraint ck_user_pwd check(length(userpwd) between 4 and 18),
	constraint ck_user_age check(age>=18),
	constraint ck_user_gender  check(gender in('男','女')),
	constraint uq_user_email unique(email)
);


create table tb_txt(
	txtid number(10),
	title varchar2(32) constraint nn_txt_title not null,
	txt varchar2(1024),
	pubtime date default(sysdate),
	userid number(5) ,
	constraint pk_txt_id  primary key(txtid),
	constraint ck_txt_id  check(length(title)>=4 and length(title)<=3
0),
	constraint fk_txt_ref_user_id foreign key(userid) references tb_u
ser(userid) on delete  cascade
);
```

7.2.2 创建并追加约束

```
create table  tb_user(
	userid number(5),
	username varchar2(30)  ,
	userpwd varchar2(20)  ,
	age  number(3)  ,
	gender char(3) ,
	email  varchar2(30),
	regtime date default(sysdate)
);
--追加约束
alter table tb_user add constraint pk_user_id  primary key (userid);
alter table tb_user add constraint ck_user_name check(length(username)b
etween 4 and 20) ;
alter table tb_user add constraint ck_user_pwd check(length(userpwd) bet
ween 4 and 18);
alter table tb_user add constraint ck_user_age check(age>=18);
alter table tb_user add constraint ck_user_gender  check(gender in('男',
'女'));
alter table tb_user add constraint uq_user_email unique(email);
--非空与默认
alter table tb_user  modify (username  constraint nn_user_name not null);
alter table tb_user  modify (userpwd  constraint nn_user_pwd not null);
```

```
alter table tb_user  modify (age default(18));
alter table tb_user  modify (gender default('男'));


create table tb_txt(
       txtid number(10),
       title varchar2(32),
       txt varchar2(1024),
       pubtime date,
       userid number(5)
);
--追加约束
alter table tb_txt add constraint pk_txt_id  primary key(txtid);
alter table tb_txt add constraint ck_txt_id  check(length(title)>=4 and
length(title)<=30);
--三种级联删除规则
alter table tb_txt add constraint fk_txt_ref_user_id foreign key(userid)
 references tb_user(userid);
alter table tb_txt add constraint fk_txt_ref_user_id foreign key(userid)
 references tb_user(userid) on delete cascade  ;
alter table tb_txt add constraint fk_txt_ref_user_id foreign key(userid)
 references tb_user(userid) on delete set null;
--注意非空 默认
alter table tb_txt modify (title constraint nn_txt_title not null)  ;
alter table tb_txt modify (pubtime default(sysdate));
```

7.2.3 查看约束

1）查看某个用户的约束

```
select constrat_name , constraint_type
  from user_constraints
where  owner = upper('scott');
```

2）查看表的约束

```
select constraint_name ,constraint_type
  from user_constraints
where table_name=upper('tb_user');
```

3）查看字段名上的约束

```
select constraint_name, column_name
  from user_cons_columns
 where table_name = upper('tb_user');
```

7.2.3 禁用和启用约束

很多时候由于业务需要，比如我们有大量的历史数据，需要和现有数据合并，当前表存在数据库约束（如非空约束），而这些历史数据又包含违背非空约束的数据

行，为了避免导入时由于违反约束而导入失败，我们通过调整约束状态来达到目的。

数据库约束有两类状态

启用/禁用（enable/disable）：是否对新变更的数据启用约束验证

验证/非验证 (validate/novalidate) ：是否对表中已客观存在的数据进行约束验证

这两类四种状态从语法角度讲可以随意组合，默认是 enable validate

下面我们来看着四类组合会分别出现什么样的效果：

enable validate：默认的约束组合状态，无法添加违反约束的数据行，数据表中也不能存在违反约束的数据行；

enable novalidate：无法添加违反约束的数据行，但对已存在的违反约束的数据行不做验证；

disable validate：可以添加违反约束的数据行，但对已存在的违反约束的数据行会做约束验证（从描述中可以看出来，这本来就是一种相互矛盾的约束组合，只不过是语法上支持这种组合罢了，造成的结果就是会导致 DML 失败）

disable novalidate：可以添加违法约束的数据行，对已存在的违反约束的数据行也不做验证。

拿上面的例子来说，我们需要上传大量违反非空约束的历史数据（从业务角度讲这些数据不会造成系统功能异常），可以临时将约束状态转为 disable novalidate，以保证这些不合要求的数据导入表中

7.2.4 删除约束

```
alter table tb_user drop constraint uq_user_email cascade;
```

第八章 DML

8.1 DML 介绍

DML（Data ManipulationLanguage 数据操控语言）用于操作**数据库对象中包含的数据**，也就是说操作的单位是**记录**。

Oracle 数据库的 DML 表数据的操作有三种：

① insert （插入）；② update（更新）；③ delete（删除）。

语句	作用
insert	向数据表张插入一条记录
delete	删除数据表中的一条或多条记录，也可以删除数据表中的所有记录，但是，它的操作对象仍是记录
update	用于修改已存在表中的记录的内容

表名：tb_user（用户表）

编号	字段名	字段类型	说明
1	userid	number(5)	用户 id,主键
2	username	varchar2(60)	用户名，非空，4～20 个字符
3	userpwd	varchar2(60)	密码，非空，4～18 个字符
4	age	number(3)	年龄，默认 18，值大于等于 18
5	gender	char(3)	性别，默认‘男’，只能是‘男’或‘女’
6	email	varchar2(60)	邮箱，唯一
7	regtime	date	注册日期，默认当前日期

表名：tb_txt（文章表）

编号	字段名	字段类型	说明
1	txtid	number(10)	文章编号，主键
2	title	varchar2(60)	文章标题，非空，长度为 4～20 字符
3	txt	varchar2(1024)	内容，最大长度 1024
4	pubtime	date	发布日期，默认当前日期
5	userid	number(5)	作者，外键，参考用户表的用户 id，删除时，自设为 null

####

```
create table  tb_user(
       userid number(5),
       username varchar2(60)  constraint nn_user_name not null ,
       userpwd varchar2(60)  constraint nn_user_pwd not null  ,
       age  number(3) default(18) ,
       gender char(3) default('男'),
```

```
        email  varchar2(30),
        regtime date default(sysdate),
        constraint pk_user_id  primary key (userid),
        constraint ck_user_name check(length(username)between 4 and 20) ,
        constraint ck_user_pwd check(length(userpwd) between 4 and 18),
        constraint ck_user_age check(age>=18),
        constraint ck_user_gender  check(gender in('男','女')),
        constraint uq_user_email unique(email)
);


create table tb_txt(
        txtid number(10),
        title varchar2(60) constraint nn_txt_title not null,
        txt varchar2(1024),
        pubtime date default(sysdate),
        userid number(5) ,
        constraint pk_txt_id  primary key(txtid),
        constraint ck_txt_id  check(length(title)>=4 and length(title)<=3
0),
        constraint fk_txt_ref_user_id foreign key(userid) references tb_u
ser(userid) on delete  set null
);
```

8.2 DML 之 insert

通过 insert 语句向指定的表中添加记录，添加记录时需要满足以下条件，类型和长度要**兼容**，（字段 兼容值）；值**满足约束**，主键 (唯一+非空) ，非空(必填)，唯一(不重复)，默认(没有填写使用默认值)，检查(满足条件)，外键(参考主表主键列的值)；个数必须相同，指定列，个**数顺序与列相同**；没有指定，个数与表结构的列个数和顺序相同 (null 也得占位，没有默认值)。

```
insert into 表名 [(字段列表)] values(值列表);
```

8.2.1 默认方式添加

insert into 表名 values（值列表）

此时的值列表顺序和个数以及类型需要和标结构一致，默认的，可以为空的列也都必须填上值。可以手写也可以从别的表中获取。

```
insert into tb_user values(1001,'test','test123',null,'女',null, sysdat
e);

insert into tb_user_copy values(select * from tb_user);
```

8.2.2 添加时指定列和顺序

insert into 表名（指定列） values(值列表)

此时的值列表要和指定的列个数、顺序、类型保持一致

```
insert into tb_user(username, userid, userpwd) values('tom',1002,'tompwd');

insert into tb_user_copy(username, userid, userpwd) (select username, userid, userpwd from tb_user);
```

当添加的记录中，存在外键关联时需要注意，可以采取先查询后添加的方式

```
insert into tb_txt(txtid, title,userid)  (1001,'title01',(select userid from tb_user where username='随笔大师'));
```

约束状态的演示

alter table 表名 disable novalidate constraint 约束名;

disable enable 是否启用（对新添加的记录做约束）

validate novalidate 石否校验（对已存在的记录做约束）

8.2 DML 之 update

通过 update 语句可以更新（修改）表中的记录值。

update 表名 set 字段 1=值 1[,字段 2=值 2,...] where 过滤行记录;

```
update tb_user set userpwd=8888 where 1=1;

update tb_user set userpwd='good',age='29' where username='zzz' and pwd='123';

update tb_user set(username, userpwd) = (select 'god','block' from dual) where userid=1;
```

8.3 DML 之 delete

通过 delete 语句可以删除表中的记录。（注意存在主外键约束的记录）

delete from 表名 where 条件；

```
delete from tb_user where userid<10;

delete from tb_user;
```

第九章 事务

9.1 事务介绍

9.1.1 什么是事务

在数据库中事务是工作的逻辑单元，一个事务是由一个或多个完成一组的相关行为的 SQL 语句组成，通过事务机制确保这一组 SQL 语句所作的操作要么都成功执行，完成整个工作单元操作，要么一个也不执行。

使用事务是为了保证数据的安全有效。

9.1.2 事务的开始与结束

事务是用来分割数据库活动的逻辑工作单元，事务即有起点，也有终点；事务的处理就是保证数据操作的完整性，所有的操作要么成功要么同时失败。

事务开启

Oracle 的事务默认开始于一个 DML 语句(insert update delete)

事务结束

事务会以成功或失败结束，当下列事件之一发生时，事务就结束了：

执行 COMMIT（提交） 或 ROLLBACK（回滚）语句；

执行一条 DDL 语句，例如 CREATE TABLE 语句；在这种情况下， 会自动执行 COMMIT 语句；

执行一条 DCL 语句，例如 GRANT 语句；在这种情况下，会自动执行 COMMIT 语句；

断开与数据库的连接。在退出 SQL Plus 时，通常会输入 EXIT 命令，此时会自动执行 COMMIT 语句。如果 SQL Plus 被意外终止了（例如运行 SQL Plus 的计算机崩溃了），那么就会自动执行 ROLLBACK 语句；

使用了隐士的事务，SET AUTOCOMMIT ON （只针对一条 SQL 语句有效）

9.2 事务特性

SQL92 标准定义了数据库事务的四个特点：

- 原子性(Atomicity)：一个事务里面所有包含的 SQL 语句是一个执行整体，不可分割，要么都做，要么都不做。
- 一致性(Consistency)：事务开始时，数据库中的数据是一致的，事务结束时，数据库的数据也应该是一致的。
- 隔离性(Isolation)：是指数据库允许多个并发事务同时对其中的数据进行读写和修改的能力，隔离性可以防止事务的并发执行时，由于他们的操作命令交叉执行而导致的数据不一致状态。
- 持久性 (Durability)：是指当事务结束后，它对数据库中的影响是永久的，即便系统遇到故障的情况下，数据也不会丢失。

一组 SQL 语句操作要成为事务，数据库管理系统必须保证这组操作的原子性（Atomicity）、一致性（consistency）、隔离性（Isolation）和持久性（Durability），这就是 ACID 特性。

针对读取数据时可能产生的不一致现象，在 SQL92 标准中定义了 4 个事务的隔离级别：

隔离级别	脏读	不可重复读	幻读
Read uncommitted(读未提交)	是	是	是
Read committed（读已提交）	否	是	是
Repeatable read（可重复读）	否	否	是
Serializable（串行读）	否	否	否

Oracle 默认的隔离级别是 read committed。

Oracle 支持上述四种隔离级别中的两种:read committed 和 serializable。除此之外，Oralce 中还定义 Read only 和 Read write 隔离级别。

Read only：事务中不能有任何修改数据库中数据的操作语句，是 Serializable 的一个子集。

Read write：它是默认设置，该选项表示在事务中可以有访问语句、修改语句，但不经常使用。

两个客户（事务）并发访问数据库数据时可能存在的问题

1.幻　读：

事务 T1 读取一条指定 where 条件的语句，返回结果集。此时事务 T2 插入一行新记录并 commit，恰好满足 T1 的 where 条件。然后 T1 使用相同的条件再次查询，结果集中可以看到 T2 插入的记录，这条新纪录就是幻想。

1.不　可重复读取：

事务 T1 读取一行记录，紧接着事务 T2 修改了 T1 刚刚读取的记录并 commit，然后 T1 再次查询，发现与第一次读取的记录不同，这称为不可重复读。

1.脏　读：

事务 T1 更新了一行记录，还未提交所做的修改，这个 T2 读取了更新后的数据，然后 T1 执行回滚操作，取消刚才的修改，所以 T2 所读取的行就无效，也就是脏数据。

9.3 事务控制命令

9.3.2 commit

在执行使用 COMMIT 语句可以提交事务，当执行了 COMMIT 语句后，会确认事务的变化，结束事务，删除保存点，释放锁。当使用 COMMIT 语句结束事务之后，其他会话将可以查看到事务变化后的新数据。

```
commit;
```

9.3.2 rollback

当执行 ROLLBACK 时表示要取消刚刚的事务中的操作。ROLLBACK 语句结束事务后，其他会话将不会察觉变化。

```
rollback;
```

9.3.3 事务保存点

保存点（savepoint）：是事务中的一点，用于取消部分事务，当结束事务时，会自动的删除该事务所定义的所有保存点。当执行 ROLLBACK 时，通过指定保存点可以回退到指定的点。

设置保存点：

```
savepoint a;
```

删除保存点：

```
release Savepoint a;
```

回滚部分事务

```
rollback to a;
```

9.3 建立事务

```
SET TRANSACTION READ ONLY--事务中不能有任何修改数据库中数据的操作语句，这包括 insert、update、delete、create 语句

SET TRANSACTION READ WRITE--默认设置,该选项表示在事务中可以有访问语句、修改语句

SET TRANSACTION ISOLATION LEVEL READ COMMITTED

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE--serialzable 可以执行 DML 操作
```

注意：这些语句是互斥的，不能够同时设置两个或者两个以上的选项

第十章 视图、索引、序列

10.1 视图

视图可以看作一张虚拟表。视图正如其名字的含义一样，是另一种查看数据的入口。

常规视图本身并不存储实际的数据，而仅仅是由 SELECT 语句组成的查询定义的虚拟表 。

从数据库系统内部来看，视图是由一张或多张表中的数据组成的，从数据库系统外部来看，视图就如同一张表一样，而且视图还可以被嵌套，一个视图中可以嵌套另一个视图。

在 SQL 中，视图是基于 SQL 语句的结果集的可视化的表。视图包含行和列，就像一个真实的表。视图中的字段就是来自一个或多个数据库中的真实的表中的字段。视图总是显示最近的数据。每当用户查询视图时，数据库引擎通过使用 SQL 语句来重建数据。

视图是建立在表、结果集或视图上的虚拟表，视图隐藏了底层的表结构，简化了数据访问操作，客户端不再需要知道底层表的结构及其之间的关系。视图提供了一个统一访问数据的接口。（即可以允许用户通过视图访问数据的安全机制，而不授予用户直接访问底层表的权限）。从而加强了安全性，使用户只能看到视图所显示的数据。

10.1.1 视图的特点

视图中的数据并不属于视图本身，而是属于基本的表，对视图可以像表一样进行 insert,update,delete 操作。

视图不能被修改，表修改或者删除后应该删除视图再重建。

视图的数量没有限制，但是命名不能和视图以及表重复，具有唯一性。

视图可以被嵌套，一个视图中可以嵌套另一个视图

10.1.2 视图的功能

1.简化用户操作

2.能以不同的角度观察同一个数据库

3.对重构数据库提供了逻辑独立性：

利用视图将需要的数据合并或者筛选，但是不影响原表的数据和结构

4.对机密数据提供安全保护：

可以建立不同的视图对用不同的用户，以达到安全的目的。

10.1.3 创建视图

注意需要有权限

create or replace view 视图名 as select 语句 [with read only]

10.1.4 视图使用

求每个部门中薪水低于平均薪水的员工姓名和薪水

求部门经理人中平均薪水最低的部门名称

10.2 索引

SQL 索引有两种，聚集索引和非聚集索引，索引主要目的是提高了数据库系统的性能，加快数据的查询速度与减少系统的响应时间

索引（index）被创建于已有的表中，它可使对行的定位更快速更有效。可以在表格的一个或者多个列上创建索引，每个索引都会被起个名字。用户无法看到索引，它们只能被用来加速查询。

更新一个包含索引的表需要比更新一个没有索引的表更多的时间，这是由于索引本身也需要更新。因此，理想的做法是仅仅在常常用于搜索的列上面创建索引。

下面举两个简单的例子：

图书馆的例子：一个图书馆那么多书，怎么管理呢？建立一个字母开头的目录，例如：a 开头的书，在第一排，b 开头的在第二排，这样在找什么书就好说了，这个就是一个索引（聚集索引），可是很多人借书找某某作者的，不知道书名怎么办？图书管理员在写一个目录，某某作者的书分别在第几排，第几排，这就是一个索引（非聚集索引）。

字典的例子：字典前面的目录，可以按照拼音和部首去查询，我们想查询一个字，只需要根据拼音或者部首去查询，就可以快速的定位到这个汉字了，这个就是索引的好处。

10.2.1 oracle 索引

1）索引是数据库对象之一，用于加快数据的检索，类似于书籍的索引。在数据库中索引可以减少数据库程序查询结果时需要读取的数据量，类似于在书籍中我们利用索引可以不用翻阅整本书即可找到想要的信息。

2）索引是建立在表上的可选对象；索引的关键在于通过一组排序后的索引键来取代默认的全表扫描检索方式，从而提高检索效率

3）索引在逻辑上和物理上都与相关的表和数据无关，当创建或者删除一个索引时，不会影响基本的表；

4）索引一旦建立，在表上进行 DML 操作时（例如在执行插入、修改或者删除相关操作时），oracle 会自动管理索引。

5）索引对用户是透明的，无论表上是否有索引，sql 语句的用法不变

6）oracle 创建主键时会自动在该列上创建索引

1. 若没有索引，搜索某个记录时（例如查找 name='wish'）需要搜索所有的记录，因为不能保证只有一个 wish，必须全部搜索一遍

2. 若在 name 上建立索引，oracle 会对全表进行一次搜索，将每条记录的 name 值哪找升序排列，然后构建索引条目（name 和 rowid），存储到索引段中，查询 name 为 wish 时即可直接查找对应地方

3. 创建了索引并不一定就会使用，oracle 自动统计表的信息后，决定是否使用索引，表中数据很少时使用全表扫描速度已经很快，没有必要使用索引

   唯一性较好字段适合建立索引，大数据量才有效果

10.2.2 索引创建和删除

create index indexName on tableName (colName...)

drop index indexName

查看索引

```
select index_name,index-type, tablespace_name, uniqueness from all_inde
xes where table_name ='tablename';
```

10.3 序列

序列(SEQUENCE)是序列号生成器，可以为表中的行自动生成序列号，产生一组等间隔的数值(类型为数字)。不占用磁盘空间，占用内存。

其主要用途是生成表的主键值，可以在插入语句中引用，也可以通过查询检查当前值，或使序列增至下一个值。

10.3.1 序列的使用

创建序列需要 CREATE SEQUENCE 系统权限。序列的创建语法如下

create sequence 序列名 start with 起始值 increment by 步进;

currval : 当前值

nextval: 下一个值

drop sequence 序列名;

- INCREMENT BY:序列变化的步进，负值表示递减。(默认 1)START WITH:序列的初始值 。(默认 1)MAXvalue:序列可生成的最大值。(默认不限制最大值，NOMAXVALUE)MINVALUE:序列可生成的最小值。(默认不限制最小值，NOMINVALUE)CYCLE:用于定义当序列产生的值达到限制值后是否循环(NOCYCLE:不循环，CYCLE:循环)。CACHE:表示缓存序列的个数，数据库异常终止可能会导致序列中断不连续的情况，默认值为 20，如果不使用缓存可设置 NOCACHE`
- 例 `CREATE SEQUENCE SEQ_DEMO INCREMENT BY 1 START WITH 1 NOMAXvalue NOCYCLE NOCACHE;`

currval 表示序列的当前值，新序列必须使用一次 nextval 才能获取到值，否则会报错

第十一章 JDBC

11.1 JDBC 介绍

JDBC（Java DataBase Connectivity,java 数据库连接）是一种用于执行 SQL 语句的 Java API，可以为多种关系数据库提供统一访问，它由一组用 Java 语言编写的类和接口组成。JDBC 提供了一种基准，据此可以构建更高级的工具和接口，使数据库开发人员能够编写数据库应用程序，同时，JDBC 也是个商标名。

我们安装好数据库之后，我们的应用程序也是不能直接使用数据库的，必须要通过相应的数据库驱动程序，通过驱动程序去和数据库打交道。其实也就是数据库厂商的 JDBC 接口实现，即对 Connection 等接口的实现类的 jar 文件。

11.2 JDBC 使用

11.2.1 常用接口

1）Driver 接口

Driver 接口由数据库厂家提供，作为 java 开发人员，只需要使用 Driver 接口就可以了。在编程中要连接数据库，必须先装载特定厂商的数据库驱动程序，不同的数据库有不同的装载方法。如：

装载 MySql 驱动：Class.forName("com.mysql.jdbc.Driver");

装载 Oracle 驱动：Class.forName("oracle.jdbc.driver.OracleDriver");

2）Connection 接口

Connection 与特定数据库的连接（会话），在连接上下文中执行 sql 语句并返回结果。DriverManager.getConnection(url, user, password)方法建立在 JDBC URL 中定义的数据库 Connection 连接上。

连接 MySql 数据库：Connection conn = DriverManager.getConnection("jdbc:mysql://host:port/database", "user", "password");

连接 Oracle 数据库：Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@host:port:database", "user", "password");

连接 SqlServer 数据库：Connection conn = DriverManager.getConnection("jdbc:microsoft:sqlserver://host:port; DatabaseName=database", "user", "password");

常用方法：

- createStatement()：创建向数据库发送 sql 的 statement 对象。
- prepareStatement(sql) ：创建向数据库发送预编译 sql 的 PrepareSatement 对象。
- prepareCall(sql)：创建执行存储过程的 callableStatement 对象。
- setAutoCommit(boolean autoCommit)：设置事务是否自动提交。
- commit() ：在链接上提交事务。
- rollback() ：在此链接上回滚事务。

3）Statement 接口

用于执行静态 SQL 语句并返回它所生成结果的对象。

两种 Statement 类：

- Statement：由 createStatement 创建，用于发送简单的 SQL 语句（不带参数）。
- PreparedStatement ：继承自 Statement 接口，由 preparedStatement 创建，用于发送含有一个或多个参数的 SQL 语句。PreparedStatement 对象比 Statement 对象的效率更高，并且可以防止 SQL 注入，所以我们一般都使用 PreparedStatement。

常用 Statement 方法：

- execute(String sql):运行语句，返回是否有结果集
- executeQuery(String sql)：运行 select 语句，返回 ResultSet 结果集。
- executeUpdate(String sql)：运行 insert/update/delete 操作，返回更新的行数。

- addBatch(String sql) ：把多条 sql 语句放到一个批处理中。
- executeBatch()：向数据库发送一批 sql 语句执行。

4）ResultSet 接口

ResultSet 提供检索不同类型字段的方法，常用的有：

- getString(int index)、getString(String columnName)：获得在数据库里是 varchar、char 等类型的数据对象。
- getFloat(int index)、getFloat(String columnName)：获得在数据库里是 Float 类型的数据对象。
- getDate(int index)、getDate(String columnName)：获得在数据库里是 Date 类型的数据。
- getBoolean(int index)、getBoolean(String columnName)：获得在数据库里是 Boolean 类型的数据。
- getObject(int index)、getObject(String columnName)：获取在数据库里任意类型的数据。

ResultSet 还提供了对结果集进行滚动的方法：

- next()：移动到下一行
- Previous()：移动到前一行
- absolute(int row)：移动到指定行
- beforeFirst()：移动 resultSet 的最前面。
- afterLast() ：移动到 resultSet 的最后面。

使用后依次关闭对象及连接：ResultSet → Statement → Connection

11.2.2 JDBC 使用步骤

加载 JDBC 驱动程序 → 建立数据库连接 Connection → 创建执行 SQL 的语句 Statement → 处理执行结果 ResultSet → 释放资源

1）注册驱动（只做一次)

Class.forName(“oracle.jdbc.driver.OracleDriver”);

2）建立连接

```
Connection conn = DriverManager.getConnection(url, user, password);
```

URL 用于标识数据库的位置，通过 URL 地址告诉 JDBC 程序连接哪个数据库，URL 的写法为：

3）创建执行 SQL 语句的 statement

```
//Statement
String id = "5";
String sql = "select * from tb_test where id=" +  id;
Statement st = conn.createStatement();
st.executeQuery(sql);
//存在 sql 注入的危险
//如果用户传入的 id 为“5 or 1=1”，那么将查询出表中的所有记录

//PreparedStatement 有效的防止 sql 注入(SQL 语句在程序运行前已经进行了预编译，当运行时动态地把参数传给 PreprareStatement 时，即使参数里有敏感字符如 or '1=1'也数据库会作为一个参数一个字段的属性值来处理而不会作为一个 SQL 指令)
String sql = "select * from tb_text where id=?";
PreparedStatement ps = conn.preparedStatement(sql);
ps.setString(1, 110); //占位符从 1 开始
ps.executeQuery();
```

4）处理执行结果(ResultSet)

```
 ResultSet rs = ps.executeQuery();
 While(rs.next()){
     rs.getString(“col_name”);
     rs.getInt(1);
     //…
 }
```

5）释放资源

```
//数据库连接（Connection）非常耗资源，尽量晚创建，尽量早的释放
 //都要加 try catch 以防前面关闭出错，后面的就不执行了
try {
    if (rs != null) {
        rs.close();
    }
} catch (SQLException e) {
    e.printStackTrace();
} finally {
    try {
        if (st != null) {
            st.close();
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
```

```
            if (conn != null) {
                conn.close();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

优士极限
www.yjxxt.com

优士极限
www.yjxxt.com

优士极限
www.yjxxt.com