# 安装oracle

# 卸载oracle

# 本地Oracle数据库

## 普通用户

用户名：scott

密码：123

## 超级管理员

用户名：sys

密码：123

# 怎么进入命令行

https://blog.csdn.net/qq\_42282074/article/details/103603681

# 特权用户（SYSDBA）身份登录到数据库实例

sqlplus / as sysdba

# 如何修改sys用户的密码

1、以 sysdba 身份登录 Oracle 数据库

sqlplus / as sysdba

2、进入 SQL\*Plus 命令行后，使用 ALTER USER 命令修改 sys 用户的密码

ALTER USER sys IDENTIFIED BY 新密码;

# 账户被锁定的原因

1、登录失败次数过多：如果用户连续多次使用错误的密码尝试登录系统，根据安全策略，系统可能会自动锁定该账户以保护数据库安全。

2、管理员手动锁定：数据库管理员可以手动锁定特定的用户账户，可能是出于安全考虑或者进行账户维护。

3、密码过期：如果数据库设置了密码过期策略，并且用户没有及时更改过期密码，账户可能会被锁定。

4、安全策略设置：数据库可能会根据安全策略锁定长时间未使用的账户，或者根据特定条件锁定账户以保护数据库安全。

# 解锁普通用户

1、除了管理员用户以外，oracle 还提供了很多普通用户，只不过普通用户默认情况下都是锁定的，不能用来登录数据库。

2、可以通过管理员用户 对这些普通用户进行解锁，如下所示：



# 普通用户修改密码

1、超级管理员修改普通用户的密码，不需要知道原密码，可以直接修改。

2、但是普通用户自己修改自己的密码，就需要知道原密码了；使用 password关键字 修改密码 如下所示：



password 为修改密码的指令：输入密码时是不显示的，直接输入即可。

# 给用户授权dba的权限

grant dba to 用户

# 修改用户缺省的schema

alter session set current\_schema = schema名

# 怎么选择指定的数据库实例

# 查询oracle数据库的版本

SELECT \* FROM v$version;

上述查询会返回数据库的版本信息以及其他相关信息。

SELECT banner FROM v$version WHERE ROWNUM = 1;

将返回数据库的版本号作为结果。

# 怎么新建实例（数据库）

# 怎么删除数据库

## 使用DBCA删除数据库

**1、单击“开始”→“Oracle\_OraDB12Home1”→“Database Configuration Assistant”，进入删除步骤的第一步，选择“删除数据库”。**



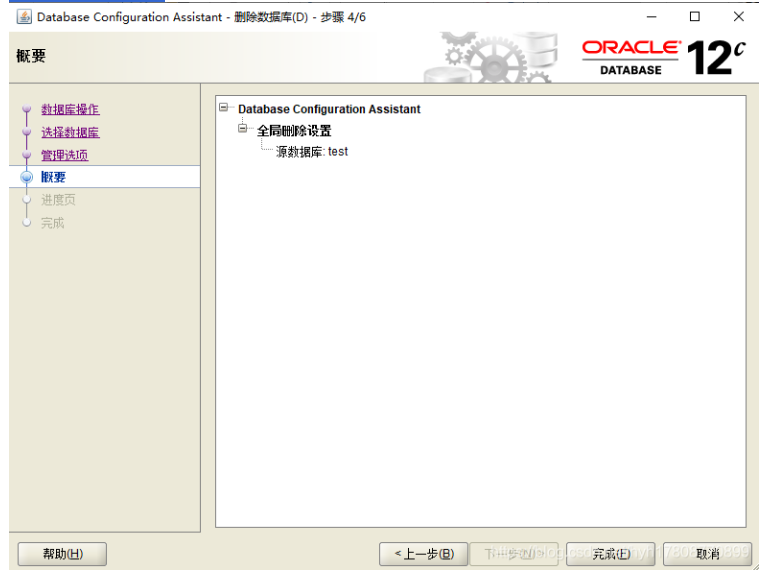
**2、单击“下一步”按钮，如图所示。在数据库列表中选择需要删除的数据，例如test，并输入用户名和口令。**



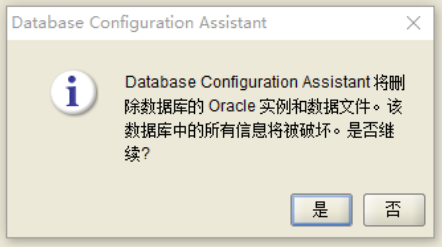
**3、单击“下一步”按钮，打开“管理选项”对话框**



**4、不要选中“取消注册Enterprise Manager(EM) Cloud Control”，单击“下一步”直接进入“删除数据库概要”窗口**



**5、设置完成后，单击“完成”按钮，弹出是否确认删除数据库消息框（如图下图所示），如果单击“否”可以重新进行删除设置，否则直接进入删除进度窗口。**



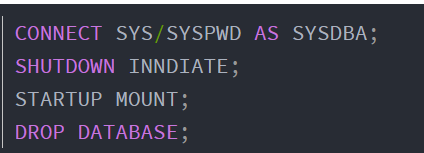
**6、打开如图所示的“删除进度”窗口。系统将连接到数据库，然后删除实例和数据文件，并更新配置网络配置文件。最后，将数据库完全删除。**



## 使用命令行删除数据库

**使用命令行删除数据库的语句是DROP DATABASE。在删除之前，需要用户以SYSDBA或SYSOPER身份登录，并且将数据库以MOUNT模式启动**

**代码如下：**



**其中，SYSPWD为SYS用户的密码，用户根据实际情况而定。**

# ****怎么删除模式及模式所有的对象****

**注意：在执行以下操作之前，请务必备份数据库，以免产生不可挽回的后果。**

**1、登录到Oracle数据库中具有足够权限的用户（通常是DBA账户）**

**2、确保没有其他会话正在使用要删除的模式，否则需要先结束这些会话。**

**3、禁用或删除与要删除的模式相关的任何作业、调度程序或其他定时任务。**

**4、确认要删除的模式不再被任何程序或应用程序使用，以免造成意外影响。**

**5、在数据库中执行以下SQL语句，以先禁用并删除与该模式关联的所有对象（表、视图、存储过程等）**

**DROP USER your\_username CASCADE;**

**其中，your\_username 是要删除的模式的用户名，CASCADE 关键字将删除用户拥有的所有对象。**

**6、如果要保留某些对象并将其转移到其他模式下，可以在删除用户之前将这些对象重新分配给其他用户。**

**7、确认删除操作执行成功，并检查数据库是否正常运行。**

**需要注意的是，删除模式是一个重要的操作，可能会对数据库产生严重影响。在执行删除操作之前，请务必仔细考虑并备份数据库，以免造成数据丢失或其他问题。**

# ****执行sql脚本文件的命令****

## ****基本概念****

**1、@符号：在 SQLPlus 工具中，@符号用于执行一个脚本文件。它告诉 SQLPlus 去读取指定路径下的脚本文件并执行其中的 SQL 语句。**

## ****示例代码****

**@D:\dropobjall.sql**

# 服务

## OracleMTSRecoveryService

## OracleDBConsoleorcl

## OracleServiceORCL

## OracleRemExecService

## OracleOraDb11g\_home2TNSListener

# 标识符是否区分大小写

在默认情况下，Oracle数据库在标识符（如表名、列名和用户名）上是不区分大小写的。这意味着，你可以使用大写、小写或混合大小写来指定标识符，而不会影响其解析或访问。

然而，对于Oracle数据库的用户身份验证（即登录名），它的默认行为是区分大小写的。这表示在创建用户时，你需要指定用户名的确切大小写，并且在进行用户身份验证时，需要提供与创建用户时相同大小写的用户名。

# 数据类型

## 字符串类型

字符型在 Oracle 11g 中有 varchar2、char、nchar、nvarchar2 和 long 五种，它们在数据库中是以 ASCII 码的格式存储的。



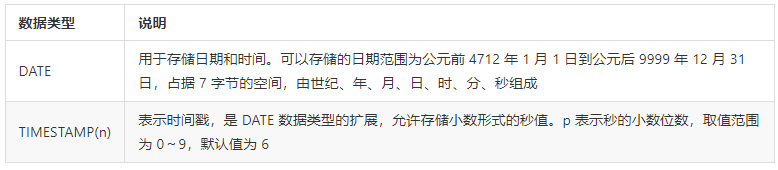
## 数字类型

数字型在 Oracle 11g 中常用的有 number 和 float 类型两种，可以用它们来表示整数和小数。



## 日期类型

日期类型在 Oracle 11g 中常用的有 date 和 timestamp 两种类型，可以用它们来存放日期和时间。



## 其他数据类型

在 Oracle 11g 中还有存放大数据的数据类型以及存放二进制文件的数据类型。



### CLOB

#### 基本概念

CLOB数据类型是用于存储大量文本数据的一种数据类型。

### BLOB

### RAW

### ROWID

# 运算符

## :=

在 Oracle 数据库中，冒号等于符号（:=）用于给变量赋值。它是一个赋值操作符，表示将右侧的值赋给左侧的变量。

例如，如果要将数字 5 赋给变量 x，可以使用冒号等于符号进行赋值操作：

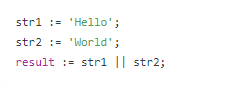


在上面的示例中，冒号等于符号将数字 5 赋给了变量 x。需要注意的是，在 Oracle 数据库中，冒号等于符号只能用于变量赋值，不能用于比较操作。在比较操作中，应使用等于符号（=）来判断两个值是否相等。

## ||

在 Oracle 数据库中，双竖线符号（||）用于字符串连接操作。它将两个字符串连接在一起，形成一个新的字符串。

例如，假设我们有两个字符串变量 str1 和 str2，我们可以使用双竖线符号将它们连接在一起：



在上面的示例中，我们将字符串 'Hello' 和 'World' 通过双竖线符号连接在一起得到了新的字符串 'HelloWorld'。该结果存储在变量 result 中。需要注意的是，双竖线符号只能用于连接字符串，不能用于其他数据类型的操作。

# 连接操作

## join与full join的区别

1、JOIN 是用于将多个表中的行连接在一起的操作。FULL JOIN 是一种特殊类型的 JOIN，它返回连接表中的所有行，包括匹配和不匹配的行。

# 单引号和双引号的区别

## 1、字符串表示

单引号：用于表示字符串常量。例如，'Hello World' 表示一个包含文本 "Hello World" 的字符串。

双引号：用于表示标识符（如列名、表名和数据库对象名）。它允许使用特殊字符或关键字作为标识符，并且区分大小写。例如，CREATE TABLE "MyTable"(...) 将创建一个名为 MyTable 的表。

## 2、字符串拼接

单引号：用于将多个字符串连接在一起。通过使用 || 运算符，可以将两个字符串拼接成一个新的字符串。例如，'Hello ' || 'World' 将返回 "Hello World"。

双引号：不能用于字符串拼接。

## 3、解析顺序

单引号：Oracle 数据库首先解析单引号内的内容作为字符串。如果出现无效字符或语法错误，可能会导致解析错误。

双引号：Oracle 数据库首先解析双引号内的内容作为标识符，然后再进行其他解析。它允许使用特殊字符或关键字作为标识符，但可能导致标识符不区分大小写。

## 4、对象引用

单引号：不能用于引用数据库对象，如表、列或索引。

双引号：允许引用数据库对象，如表、列或索引。使用双引号引用的对象名将被视为区分大小写。

需要注意的是，在大多数情况下，建议在 Oracle 数据库中使用单引号来表示字符串常量，而使用双引号来表示标识符是可选的，并且并不常见。除非有特殊要求，否则不建议过度使用双引号，以避免引起混淆和不必要的复杂性。

# 条件表达式

## CASE WHEN

### 基本概念

CASE WHEN 是 Oracle 中的条件表达式，用于根据给定的条件选择不同的结果。

### 语法

CASE

WHEN condition1 THEN result1

WHEN condition2 THEN result2

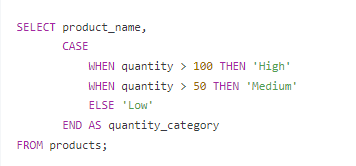
...

ELSE resultN

END

在这个语法中，condition1、condition2 等是可以进行比较的条件表达式，而 result1、result2 等则是与相应条件匹配时返回的结果。CASE WHEN 语句的执行过程是按顺序逐个检查条件，当第一个条件满足时，返回对应的结果，并终止进一步的检查。如果没有任何条件匹配，且存在 ELSE 子句，则返回 ELSE 子句中的结果。如果没有匹配的条件，且没有 ELSE 子句，则返回 NULL。

### 示例



在这个示例中，根据商品的数量（quantity 列），我们使用 CASE WHEN 语句来为每个商品选择一个数量类别。当数量大于 100 时，分类为 'High'，数量在 50 和 100 之间时，分类为 'Medium'，其他情况下分类为 'Low'。查询结果中的 quantity\_category 列将显示所属的数量类别。

### 注意事项

CASE WHEN 语句还可以在 SELECT、WHERE、ORDER BY 等子句中使用，以根据不同的条件进行灵活的数据处理和筛选。根据具体的需求和条件，你可以构建复杂的 CASE WHEN 表达式来实现更精确的逻辑判断和结果选择。

# 目录

1、

# 表空间

## 基本概念

表空间是Oracle数据库中用于存储和管理数据的逻辑存储单元。每个表空间都由一个或多个数据文件组成，数据文件是实际存储数据的物理文件。

## 作用

1、管理和控制数据库中的数据存储和分配。

2、组织和管理数据库对象（如表、索引、视图等）的存储。

3、提供了数据的物理隔离和逻辑分离，便于管理和备份恢复。

## 类型

### 系统表空间

包含系统级别的对象，如数据字典、系统表、存储过程等。

### 用户表空间

用于存储用户数据和用户创建的对象。

### 临时表空间

用于存储临时数据，比如排序、哈希表等。

### 回滚表空间

用于存储回滚段，支持事务的回滚操作。

## 表空间的管理

### 查询所有表空间物理位置

select t1.name,t2.name from v$tablespace t1, v$datafile t2 where t1.ts# = t2.ts#

### 删除空的表空间，不包含物理文件

### 删除表空间

# 模式

## 基本概念

模式（Schema）是一个逻辑上的容器，用于组织和管理数据库对象。它定义了一组相关的表、视图、索引、存储过程、函数等数据库对象，并为这些对象提供了命名空间和访问权限。

## 模式对象

我们可以在模式中创建表（Table）, 分区（Partition）, 视图（View）, 索引（Indexe）, 包（Package）,存储过程（Procedure）,函数（Function）,触发起（Trigger）,类型（Type）, 序列（Sequence）, 同义（Synonym）等数据库对象。

## 创建模式

模式隶属于某个数据库用户，模式名和数据库用户是相同的。即创建了一个数据库用户oracle就会自动的创建一个同名的模式。

## 模式和用户的区别

1、一个模式是一个数据库对象的集合（被一个用户所使用）

2、模式对象是直接引用数据库数据的逻辑结构。

3、用户是在数据库中定义的可以连接和访问对象的名称。

4、模式和用户都是为了帮助数据库管理员管理数据库的安全。

# 用户

## 删除用户

DROP USER username CASCADE;

CASCADE 关键字表示级联删除，它会删除用户拥有的所有对象，包括表、视图、索引、触发器等。如果不加 CASCADE 关键字，那么在用户还拥有对象时，将无法删除该用户。需要注意的是，执行这个删除用户的操作需要具有足够的权限，通常需要使用具有 DROP USER 权限的用户账号来执行。此外，在执行删除操作之前，请务必备份数据库或确认操作的安全性。

# 表

## 说明

字段名要大写。

## dual

### 基本概念

DUAL是Oracle数据库中的一个特殊表，它只有一行一列，并且没有实际存储的数据。

### 用途

1、DUAL表的主要作用是在没有实际表可用的情况下执行一些不需要表数据的查询操作。

2、还可以用来生成常量或者测试表达式。

### 结构

1、DUAL表只有一个列名为DUMMY的列。

2、该列的数据类型是VARCHAR2(1)，并且只有一行数据，其值为固定的X。

### 使用场景

1、查询常量：由于DUAL表只有一行数据，可以利用它来查询常量值，而不需要实际的表。

2、测试表达式：可以使用DUAL表来测试一些简单的表达式，比如日期函数、数值计算等。

### 示例

#### 1、查询常量

SELECT 'Hello, World!' FROM DUAL;

这将返回一个包含字符串常量Hello, World!的结果集。

#### 2、测试表达式

SELECT SYSDATE FROM DUAL;

这将返回当前日期时间。

# 单表备份的方法

https://www.cnblogs.com/kakaisgood/p/11388476.html#:~:text=%E5%A4%87%E4%BB%BD%E6%96%B9%E6%A1%88%E4%B8%80%EF%BC%9A1.%20%E5%A4%87%E4%BB%BDcreate%20table%20%5B%E5%A4%87%E4%BB%BD%E5%90%8D%5D%20as%20select%20%2A%20from,org\_group%3Binsert%20into%20org\_group%20select%20%2A%20from%20%5B%E5%A4%87%E4%BB%BD%E5%90%8D%5D%20%3B3.

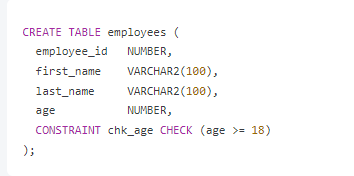
# 检查

## 基本概念

在 Oracle 数据库中，"检查"（CHECK）是一种约束（constraint）类型，用于限制表中某个列的取值范围。CHECK 约束可确保特定列中的数据满足指定的条件，如果不满足条件，将会阻止对表的修改操作。通过使用 CHECK 约束，可以在表的定义中为列添加逻辑条件，从而确保插入、更新或删除操作符合指定的规则。这些规则可以是简单的比较，也可以是复杂的表达式。

## 示例

如何在创建表时定义 CHECK 约束：



在上述示例中，employees 表中的 age 列被定义为必须大于等于 18 的整数。这是通过使用 CHECK 约束来实现的。如果试图插入年龄小于 18 的数据行，或者更新年龄为小于 18 的值，将会触发约束违反错误，阻止这些操作。CHECK 约束可以使用各种逻辑运算符、函数和表达式来定义更复杂的条件。它们提供了一种有效的方式来确保数据的完整性和一致性，以符合应用程序的业务规则和要求。

需要注意的是，CHECK 约束只能对单个表中的数据进行约束，而不能跨多个表。此外，CHECK 约束只在插入、更新和删除操作时生效，不会自动对已存在的数据进行检查和修改。因此，在设计表结构时，应仔细考虑并定义适当的 CHECK 约束来保证数据的有效性和合法性。

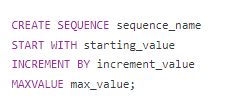
# 序列

## 基本概念

当你在Oracle数据库中需要创建一个自增的数字序列时，可以使用序列（Sequence）对象来实现。序列是一个独立的数据库对象，它生成一系列唯一的数字值。序列可以用于为表的主键列生成唯一标识符或者为其他需要自增数字的列提供值。通过序列，可以保证生成的数字值在整个数据库中是唯一的。

## 创建序列

使用CREATE SEQUENCE语句可以创建序列，指定序列的名称、起始值、递增值和最大值等属性。例如：



其中，sequence\_name是序列的名称，starting\_value是序列的初始值，increment\_value是每次递增的值，max\_value是序列的最大值。

## 使用序列

要使用序列生成新的值，可以使用NEXTVAL函数调用序列。例如：

SELECT sequence\_name.NEXTVAL FROM dual;

这将返回序列的下一个值。

## 查询序列当前的值

SELECT sequence\_name.CURRVAL FROM dual;

其中，sequence\_name是你要查询的序列名称。通过在SELECT语句中使用序列名和.CURRVAL关键字，可以获取序列的当前值。需要注意的是，在查询序列的当前值之前，必须至少调用一次该序列的NEXTVAL函数，以确保序列已经开始生成值。

## 修改序列的值

（1）如果需要将序列的当前值设置为一个特定的新值，可以使用 ALTER SEQUENCE 语句结合 INCREMENT BY 子句来实现。假设你希望将序列的当前值设置为 100，你可以执行以下 SQL 语句：

ALTER SEQUENCE sequence\_name INCREMENT BY 100;

（2）然后使用 NEXTVAL 函数调用该序列，以便它增加到你想要的新当前值：

SELECT sequence\_name.NEXTVAL FROM dual;

（3）最后，使用 ALTER SEQUENCE 语句将序列的增量修改回原始值（通常是1）

ALTER SEQUENCE sequence\_name INCREMENT BY 1;

## 应用

序列通常用于为表的主键列生成唯一标识符。在插入新记录时，可以使用序列生成下一个唯一的值，然后将其分配给主键列。这样可以确保每个记录都具有唯一的标识符。

需要注意的是，序列生成的值是预先计算的，即使在事务回滚时也不会撤销已生成的值。因此在使用序列生成的值之后，应该小心管理和处理这些值，以避免数据的混乱或重复。

# 索引

## 基本概念

1、索引是一种特殊的数据结构，用于加速对表的查询操作。它类似于书籍的目录，可以帮助数据库快速定位需要的数据。

2、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。

3、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，索引删除，不会对表产生影响。

4、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。

5、oracle创建主键时会自动在该列上创建索引。

## 数据结构

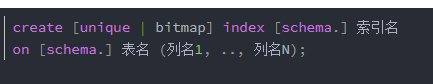
oracle 存储索引的结构是B\*数（平衡树），而索引是由根节点、分支点和叶子点组成，上级索引块包含下级索引块的索引数据，叶节点包含索引数据和确定行实际位置的rowid。

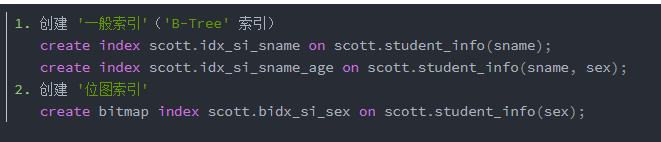
## 优点

## 索引建立的原则

1、如果有两个或者以上的索引，其中有一个唯一性索引，而其他是非唯一，这种情况下oracle将使用唯一性索引而完全忽略非唯一性索。

## 创建索引





## 修改索引

### 修改索引名称

alter index scott.idx\_si\_sname rename to idx\_si\_sname\_new;

### 修改索引为无效

### 重建索引

## 删除索引

## 查看索引

select \* from user\_indexes where table\_name = '表名'

或者

select \* from all\_indexes where table\_name = '表名'

### 查看表中索引对饮了哪些列

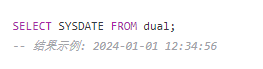
select \* from user\_ind\_columns where table\_name='表名'

# 函数

## 日期时间函数

### SYSDATE

返回当前日期和时间（包括时分秒）



### CURRENT\_DATE

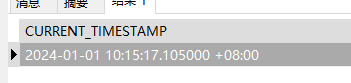
返回当前日期和时间（包括时分秒）

SELECT CURRENT\_DATE FROM dual;

### CURRENT\_TIMESTAMP

返回当前日期和时间（包括时分秒）

SELECT CURRENT\_TIMESTAMP FROM dual;



### EXTRACT()

从日期或时间中提取指定的部分，例如年、月、日、小时、分钟等。

SELECT EXTRACT(YEAR FROM SYSDATE) AS year,

EXTRACT(MONTH FROM SYSDATE) AS month,

EXTRACT(DAY FROM SYSDATE) AS day

FROM dual;

-- 结果示例: year=2024, month=1, day=1

### ADD\_MONTHS

返回一个日期加上指定月份后的日期

SELECT ADD\_MONTHS(SYSDATE, 2) FROM dual;

-- 结果示例: 2024-03-01 12:34:56

### MONTHS\_BETWEEN()

计算两个日期之间的月份数。

SELECT MONTHS\_BETWEEN(SYSDATE, TO\_DATE('2023-10-01', 'YYYY-MM-DD')) FROM dual;

### LAST\_DAY()

返回给定日期所在月份的最后一天。

SELECT LAST\_DAY(SYSDATE) FROM dual;

-- 结果示例: 2024-01-31 12:34:56

### TRUNC()

截断日期或时间的部分，例如截断到年、月、日等

SELECT TRUNC(SYSDATE, 'MM') FROM dual;

-- 结果示例: 2024-01-01 00:00:00

### ROUND()

对日期或时间进行四舍五入操作。

SELECT ROUND(SYSDATE, 'HH24') FROM dual;

-- 结果示例: 2024-01-01 13:00:00

## to\_char

### 基本概念

用于将不同的数据类型转换为字符类型。它可以将数字、日期、时间和其他数据类型转换为指定格式的字符串。

### 语法

TO\_CHAR(value, format)

### 常见用法

#### 1、将数字转换为字符串

SELECT TO\_CHAR(12345) FROM dual;

#### 2、格式化日期和时间

SELECT TO\_CHAR(SYSDATE, 'YYYY-MM-DD HH24:MI:SS') FROM dual;

这会将当前日期和时间以 "YYYY-MM-DD HH24:MI:SS" 的格式转换为字符串。例如："2023-12-20 14:54:42"。

## to\_date

### 基本概念

用于将字符串转换为日期类型

### 语法

TO\_DATE(string, format)

其中，string 是待转换的字符串，format 是格式化字符串，它告诉数据库如何解释输入的字符串。

支持的格式化字符串包括以下常用的占位符

YYYY：四位数的年份

MM：月份（01-12）

DD：日期（01-31）

HH24：24 小时制的小时数（00 - 23）

MI：分钟数（00 - 59）

SS：秒数（00 - 59）

### 示例

#### 1、将字符串 '2023-12-21' 转换为日期类型

SELECT TO\_DATE('2023-12-21', 'YYYY-MM-DD') AS converted\_date

FROM dual;

#### 2、将字符串 '2023-12-21 06:19:41' 转换为日期类型

SELECT TO\_DATE('2023-12-21 06:19:41', 'YYYY-MM-DD HH24:MI:SS') AS converted\_date FROM dual;

## to\_number

### 基本概念

用于将一个字符串转换为数值类型

### 语法

TO\_NUMBER(string, [format], [nlsparams])

参数说明：

string：要转换为数值的字符串。

format（可选）：指定输入字符串的格式。它是一个模式，用于匹配输入字符串，并确定如何进行转换。如果省略该参数，则使用默认格式。

nlsparams（可选）：指定与转换相关的国家/地区设置。它可以包含日期、货币和数值格式的信息。

### 示例

1、基本用法

SELECT TO\_NUMBER('123') FROM dual;

2、使用格式化选项

SELECT TO\_NUMBER('1,000.50', '9,999.99') FROM dual;

### 注意事项

如果无法将字符串转换为数值，TO\_NUMBER() 函数将会抛出一个异常。在使用时，可以通过适当的错误处理来捕获和处理这些异常，以确保查询的稳定性。

## SUBSTR

### 基本概念

Oracle 数据库中用于提取字符串的子串的函数。

### 语法

SUBSTR(string, start\_position, [length])

string：要截取子串的源字符串。

start\_position：指定开始截取的位置。第一个字符的位置为1。

length（可选）：指定要截取的子串的长度。如果不提供此参数，则默认截取从 start\_position 开始到字符串末尾的所有字符。

### 示例

#### 1、截取字符串的一部分

SELECT SUBSTR('Hello World', 7) FROM dual;

-- 结果: World

#### 2、指定截取子串的长度

SELECT SUBSTR('Hello World', 7, 5) FROM dual;

-- 结果: World

在上述示例中，我们使用 SUBSTR() 函数从字符串 'Hello World' 的第 7 个字符开始截取长度为 5 的子串。

#### 3、截取字符串的一部分并进行操作

SELECT SUBSTR('Hello World', 7, 5) || '!' FROM dual;

-- 结果: World!

在上述示例中，我们使用 SUBSTR() 函数从字符串 'Hello World' 的第 7 个字符开始截取长度为 5 的子串，然后使用 || 运算符将截取的子串与 '!' 字符串进行拼接。

### 注意事项

如果指定的 start\_position 大于源字符串的长度，或者提供的 length 参数超出可用字符的范围，则 SUBSTR() 函数将返回空字符串。在使用时，请确保提供正确的参数以避免意外的结果。

## INSTR

### 基本概念

用于查找一个字符串在另一个字符串中第一次出现的位置

### 语法

INSTR(string, substring, [start\_position], [occurrence])

string：要在其中查找子字符串的源字符串。

substring：要查找的子字符串。

start\_position（可选）：指定开始查找的位置。第一个字符的位置为 1。如果不提供此参数，默认从字符串的第一个字符开始查找。

nth\_appearance（可选）：指定查找子字符串的第几次出现。如果不提供此参数，默认返回第一次出现的位置。

### 示例

#### 1、查找子字符串的位置

SELECT INSTR('Hello World', 'o') FROM dual;

-- 结果: 5

#### 2、指定开始查找的位置

SELECT INSTR('Hello World', 'o', 6) FROM dual;

-- 结果: 8

在上述示例中，我们使用 INSTR() 函数从源字符串 'Hello World' 的第 6 个字符开始查找子字符串 'o' 的位置，并返回该位置。

#### 3、指定查找子字符串的第几次出现

SELECT INSTR('Hello World Hello', 'o', 1, 2) FROM dual;

-- 结果: 8

在上述示例中，我们使用 INSTR() 函数从源字符串 'Hello World Hello' 的第 2次出现的位置开始查找子字符串 'o' 的位置，并返回该位置。

## ROW\_NUMBER() OVER()

### 基本概念

用于为结果集中的行分配唯一的序号

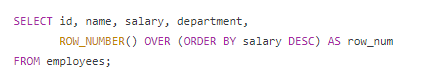
### 语法

ROW\_NUMBER() OVER (PARTITION BY column1, column2,... ORDER BY column3, column4,...)

其中，PARTITION BY 子句可选，用于指定分区列，它将结果集按照指定的列进行分组。ORDER BY 子句用于指定排序列，决定了分配序号的顺序。

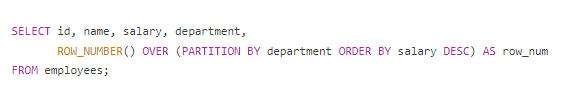
### 示例

假设我们有一个名为 employees 的表，包含以下字段：id、name、salary、department。



这个 SQL 查询将返回 employees 表中的所有行，并为每一行分配一个唯一的序号（row\_num 列）。行将按照薪资（salary）的降序排列，即薪资最高的员工将获得序号 1，其次是薪资第二高的员工，以此类推。

如果要在特定的部门内分配序号，可以使用 PARTITION BY 子句：



这个查询将根据部门（department）对结果集进行分区，并在每个分区内按照薪资的降序分配序号。每个部门内，薪资最高的员工将获得序号 1，其次是薪资第二高的员工，以此类推。使用 ROW\_NUMBER() OVER() 函数可以根据指定的条件为结果集中的行分配唯一的序号。这在需要对查询结果进行排序、分组或者筛选时非常有用。

## regexp\_substr

### 基本概念

Oracle的REGEXP\_SUBSTR函数是一个正则表达式函数，用于在字符串中查找满足指定正则表达式模式的子字符串，并返回匹配到的结果。

### 语法

REGEXP\_SUBSTR(source\_string, pattern, position, occurrence, match\_parameter)

source\_string：要进行匹配的源字符串。

pattern：要匹配的正则表达式模式

position（可选）：指定开始搜索的位置，默认为 1。

occurrence（可选）：指定匹配到的第几个结果，默认为 1。

match\_parameter（可选）：指定匹配参数，如 'i' 表示不区分大小写匹配。

### 示例

#### 提取电子邮件地址

SELECT REGEXP\_SUBSTR('john@example.com', '[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}') AS email FROM dual;

## regexp\_replace

## ABS

返回一个数的绝对值

## ASCII

返回一个字符的ASCII码

## AVG

返回一组数的平均值

## CEIL

返回一个数的上限整数

## CONCAT

连接两个字符串

## CONVERT

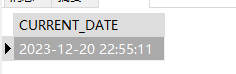
将一个字符从一种字符集转换为另一种字符集

## COUNT

返回一组数的数量

## CURRENT\_DATE

返回当前日期



## CURRENT\_TIMESTAMP

## EXTRACT

## FLOOR

## GREATEST

## INITCAP

# 自定义函数

## 基本概念

在Oracle中，函数是一种可重用的代码块，用于执行特定的任务并返回一个值。函数可以接收参数，并且可以根据输入参数进行计算和处理，然后返回一个结果。

## 语法

# 视图

## 基本概念

Oracle视图是一种虚拟表，它不存储实际数据，而是基于查询语句生成的结果集。视图可以被看作是一个预定义的查询结果集，可以方便地重复使用和共享，同时也能够保护敏感数据。在Oracle中，视图可以由单个表、多个表或其他视图构建而成，因此可以通过视图来实现数据的逻辑组织和排列。

## 语法

CREATE [OR REPLACE] [FORCE|NO\_FORCE] VIEW view\_name[(column\_list)]

AS

SELECT column1, column2, ...

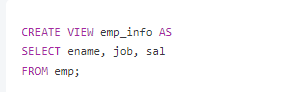
FROM table\_name

[WHERE condition];

其中，OR REPLACE表示如果视图已经存在，则替换原有视图；FORCE表示在创建视图时忽略错误，NO\_FORCE则相反；view\_name是视图名称；column\_list是可选的列名列表；table\_name是要查询的表名；condition是可选的查询条件。

## 示例

下面的语句创建了一个简单的视图，用于显示员工的姓名、工作岗位和薪水：



创建完成后，可以使用以下语句查询该视图：



# 实体化视图

# 视图的图标上有感叹号

在Navicat for Oracle中，如果视图的图标上出现感叹号，通常表示该视图的元数据信息发生了变化或存在一些问题。具体来说，可能有以下几种情况：

1、视图的定义被修改

如果您在Navicat中打开一个视图并修改了其定义，例如添加、删除或更改列，那么该视图的图标上就会出现感叹号。这表明该视图的元数据与数据库中的实际定义不同步。

2、视图所依赖的对象发生了变化

如果视图所依赖的表、视图、函数等对象发生了变化，例如被删除、重命名或修改了定义，那么该视图的图标上也会出现感叹号。这表示该视图的元数据需要更新以反映最新的依赖关系。

3、视图的元数据信息不完整或不正确

如果视图的元数据信息不完整或不正确，例如缺少必要的列或属性，那么该视图的图标上也会出现感叹号。这表明需要修复或完善视图的元数据信息。

无论是哪种情况，都应该尽快处理视图图标上的感叹号，以确保Navicat中的元数据与数据库中的实际情况保持同步，并避免潜在的错误和混淆。您可以尝试通过重新加载视图、更新元数据或手动修改视图定义等方式来解决这些问题。

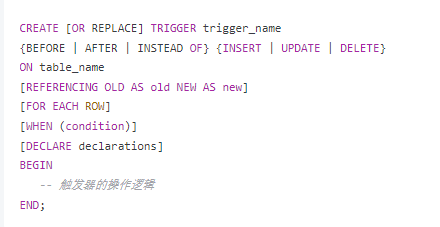
# 触发器

## 基本概念

触发器是与表关联的数据库对象，它可以在特定的操作（插入、更新或删除数据）发生时自动执行一系列操作。触发器由事件、时间点和动作组成。事件指的是触发器所响应的操作类型，时间点指的是触发器的执行时间，动作指的是触发器执行时要执行的具体操作。

## 创建触发器

使用CREATE TRIGGER语句可以在Oracle数据库中创建一个触发器。语法如下：



trigger\_name：触发器的名称

BEFORE / AFTER / INSTEAD OF：指定触发器的执行时间点。

INSERT / UPDATE / DELETE：指定触发器所响应的操作类型。

table\_name：触发器关联的表名。

REFERENCING OLD AS old NEW AS new：可选项，用于在触发器中引用旧值和新值。

## 删除触发器

# 存储过程

## 基本概念

当谈到 Oracle 数据库中的存储过程时，它是一组预编译的 SQL 语句块，可以在数据库服务器上进行定义、保存和执行。存储过程可以接受输入参数，并返回一个或多个结果。它们可以用于实现复杂的业务逻辑和数据处理操作。

## 语法

CREATE [OR REPLACE] PROCEDURE procedure\_name

[(parameter\_name [IN | OUT | IN OUT] type, ...)]

[IS | AS]

Declaration\_section

BEGIN

Executable\_section

EXCEPTION

Exception\_section

END [procedure\_name];

## 执行存储过程

要执行存储过程，可以使用 EXECUTE 或 EXEC 命令，或者直接调用存储过程的名称。

### 命令窗口

EXEC  procedure; --procedure是存储过程名

### sql窗口

用 begin  存储过程名  end;

begin

  procedure();--procedure是存储过程名

end;

说明：存储过程名字后面一定要添加()。

### 程序中

如果是程序中调用就用 call 存储过程名

hibernateDao.excuteSqlUpdate("{Call proc\_stuInfo()}");

//存储过程proc\_stuInfo

## 优点

1、**提高性能**：存储过程在数据库服务器上执行，避免了频繁的网络通信，因此可以减少数据传输延迟，并提高处理速度。此外，存储过程还可以通过在数据库中缓存执行计划来优化查询性能。

2、**提高安全性**：存储过程允许通过授权机制进行访问控制，只有授权用户才能执行特定的存储过程。这样可以确保数据的安全性，并防止未经授权的访问。

3、**简化开发**：存储过程可以将复杂的业务逻辑封装为一个单独的代码块，以便重复使用。这样可以减少开发人员编写和维护的代码量，并提高开发效率。

4、**提供事务控制**：存储过程可以作为一个原子操作执行，并且可以使用事务管理机制来确保数据的一致性和完整性。这样可以避免在应用程序中手动管理事务，并减少错误的可能性。

5、**降低网络负载**：存储过程可以在数据库服务器上执行，仅返回结果给客户端，避免了大量的数据传输，减少了网络流量和网络负载。

## 缺点

1、**学习和维护成本高**：存储过程需要使用特定的语法和编程技巧进行开发，对于开发人员来说可能需要一定的学习和培训成本。此外，存储过程的维护也需要额外的工作，包括更新和修改存储过程以满足业务需求。

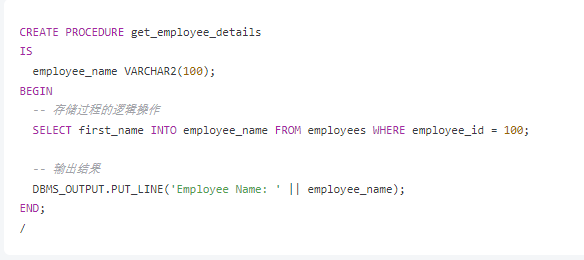
2、**难以调试**：与应用程序不同，存储过程的调试相对复杂。调试存储过程通常需要使用特定的工具和技术，这对于开发人员来说可能会增加一些困难。

3、**数据库依赖性**：存储过程是在数据库服务器上执行的，因此存储过程与特定的数据库系统紧密相关。如果需要更改数据库系统或迁移到其他数据库平台，可能需要重新编写和调整存储过程。

4、**难以版本控制**：存储过程通常部署在数据库服务器上，并与数据库紧密集成，因此在进行版本控制时可能会遇到一些挑战。在进行存储过程升级或回滚时，需要谨慎处理以避免数据丢失或不一致。

## 示例

### 1、创建无参的存储过程

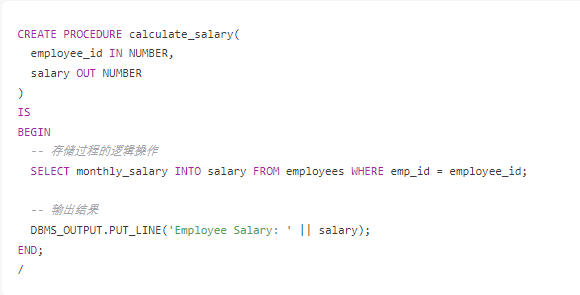


上述示例创建了一个名为 get\_employee\_details 的存储过程。它首先声明了一个变量 employee\_name，然后在逻辑部分中执行了一个 SQL 查询，并将结果赋值给该变量。最后使用 DBMS\_OUTPUT.PUT\_LINE 函数打印查询结果。

执行存储过程：

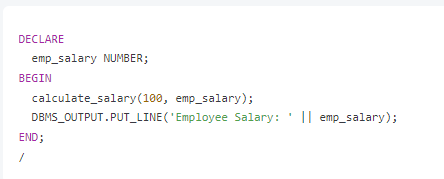
EXECUTE get\_employee\_details;

### 2、创建带参数的存储过程



以上示例中，存储过程 calculate\_salary 接受一个输入参数 employee\_id 和一个输出参数 salary。在存储过程内部，根据给定的 employee\_id 查询员工的月薪，并将结果赋值给 salary 参数。最后使用 DBMS\_OUTPUT.PUT\_LINE 函数打印查询结果。

调用存储过程：



# 数据库链接

## 基本概念

数据库链接（Database Link）是一种允许在一个数据库中访问另一个数据库的对象。它提供了跨数据库之间进行数据查询、数据传输和共享的能力。

## 创建数据库链接

使用CREATE DATABASE LINK语句可以在Oracle数据库中创建一个数据库链接。语法如下：

CREATE [PUBLIC] DATABASE LINK link\_name

CONNECT TO username IDENTIFIED BY password

USING 'connect\_string';

link\_name：数据库链接的名称。

username：目标数据库的用户名。

password：目标数据库的密码。

connect\_string：目标数据库的连接字符串。

注意：创建数据库链接的用户需要具有CREATE DATABASE LINK系统权限或者CREATE PUBLIC DATABASE LINK权限才能执行该语句。

## 访问远程数据库

一旦创建了数据库链接，就可以使用链接名称在当前数据库中访问远程数据库中的对象。在SQL语句中，可以使用“@link\_name”来引用远程数据库的对象。例如：



## 远程数据库的权限和安全性

当使用数据库链接访问远程数据库时，需要注意相应的权限和安全性设置。远程数据库中的用户需要具有适当的权限，以便通过链接进行访问，并且确保链接的用户名和密码是正确和安全的。

## 管理数据库链接

Oracle提供了多种管理数据库链接的方法。可以使用DROP DATABASE LINK语句来删除一个数据库链接。可以使用数据字典视图（如ALL\_DB\_LINKS）来查看当前数据库中存在的链接。

### 查看数据库链接

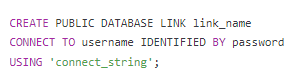
SELECT \* from ALL\_DB\_LINKS

### 删除数据库链接

DROP DATABASE LINK 链接名;

## 公共数据库链接

可以创建公共数据库链接，这意味着该链接对所有用户可见并可用。创建公共链接时，使用PUBLIC关键字，例如：



公共链接对于访问远程数据库中的共享数据或共享服务非常有用。

## 公共数据库链接和数据库链接的区别

# Java源码

## 修改了怎么保存

点击F8就可以了。

# :new

## 基本概念

在 Oracle 数据库中，:new 是一个特殊的闪回变量（flashback variable）或伪记录（pseudo-record），用于在触发器（trigger）中引用新插入或更新的数据行。当触发器被触发时，即在插入、更新或删除表中的数据时，:new 变量会包含与正在进行操作的语句相关联的新数据行。它类似于一个临时的记录，可以访问新数据行中的列和值。

:new 变量可以在触发器的代码块中使用，用于引用新插入或更新的数据行的列和值。例如，在 :new.column\_name 中，column\_name 是表中的列名，可以用于读取该列的值。

## 示例

CREATE OR REPLACE TRIGGER my\_trigger

BEFORE INSERT ON my\_table

FOR EACH ROW

BEGIN

select SEQ\_WORKFLOW\_INSTANCE.nextval, sysdate into :new.id, :new.rq from dual;

END;

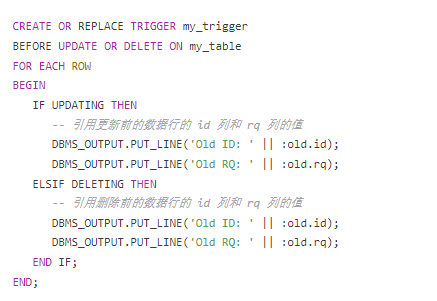
在这个示例中，my\_trigger 是一个在 my\_table 表上的插入触发器。当在表中插入新数据时，触发器会被自动触发，并通过 :new.id 和 :new.rq 引用新数据行的 id 列和 rq 列的值。然后这些值可以在触发器的代码块中进行处理或打印输出。

# :old

## 基本概念

:old 是一个特殊的闪回变量（flashback variable）或伪记录（pseudo-record），用于在触发器（trigger）中引用旧的更新或删除的数据行。当触发器被触发时，即在更新或删除表中的数据时，:old 变量会包含与正在进行操作的语句相关联的旧的数据行。它类似于一个临时的记录，可以访问旧数据行中的列和值。:old 变量可以在触发器的代码块中使用，用于引用旧的更新或删除的数据行的列和值。例如，在 :old.column\_name 中，column\_name 是表中的列名，可以用于读取该列的旧值。

## 示例



在这个示例中，my\_trigger 是一个在 my\_table 表上的更新和删除触发器。当在表中更新或删除数据时，触发器会被自动触发，并根据操作类型（更新或删除）通过 :old.id 和 :old.rq 引用旧的数据行的 id 列和 rq 列的值，然后可以在触发器的代码块中根据需要进行处理或打印输出。

# dbms\_lob.createtemporary

# PRAGMA AUTONOMOUS\_TRANSACTION

# .dbf文件

在Oracle数据库中，.dbf文件是数据文件（Data File）的一种，用于存储数据库中的表、索引等对象的数据。每个Oracle数据库都包含一个或多个数据文件，这些数据文件保存着实际的数据。当用户向表中插入一条新记录或更新一条现有记录时，数据就会被写入到数据文件中。数据文件是一个二进制文件，它们包含了由Oracle管理的数据块（Data Block）。Oracle将数据文件分成固定大小的块，通常为8KB或16KB，每个块都有一个唯一标识符。Oracle使用这些块来存储和管理数据库中的数据。数据文件通常存储在硬盘上，可以在创建数据库时指定其存储位置。如果需要扩展数据库容量，可以通过添加新的数据文件来实现。同样，如果需要缩减数据库容量，也可以通过删除不需要的数据文件来实现。需要注意的是，直接操作数据文件可能会导致数据损坏或丢失，因此应该使用Oracle提供的工具和命令来管理数据文件。例如，可以使用ALTER DATABASE命令来添加或删除数据文件，使用RENAME FILE命令来重命名数据文件等。

# .dmp文件

1、.dmp 文件是 Oracle 数据库中使用的一种备份文件格式，它包含了数据库对象、数据和元数据的二进制表示。

2、.dmp 文件是一个二进制文件，保存了 Oracle 数据库的备份数据。它不是一个可执行的程序或脚本文件，而是一种数据文件。

3、.dmp 文件包含了整个或部分 Oracle 数据库的备份内容，包括表、索引、约束、视图、存储过程、触发器等数据库对象的定义和数据。

4、.dmp 文件通常是通过 Oracle 提供的数据导出工具 expdp（数据泵导出）生成的。expdp 是 Oracle 数据库的一个实用程序，用于导出数据库对象和数据到 .dmp 文件。

5、导入工具：.dmp 文件可以使用 Oracle 提供的数据导入工具 impdp（数据泵导入）进行恢复。impdp 可以将 .dmp 文件中的备份内容导入到 Oracle 数据库中，以恢复数据库到备份时的状态。

6、.dmp 文件是 Oracle 数据库的标准备份格式，可以在不同的 Oracle 数据库实例之间进行导入和导出操作，这使得将数据库从一个环境迁移到另一个环境变得更加方便。

7、需要注意的是，.dmp 文件只是数据库的备份文件，如果要恢复数据库到备份时的状态，需要使用相应的导入工具进行操作。此外.dmp文件可能会比较大，备份和恢复过程可能需要一定的时间和系统资源。因此在使用 .dmp 文件备份和恢复数据库时，需要确保有足够的磁盘空间和系统性能来处理这些操作。

# 数据泵

## 基本概念

Oracle 数据泵（Data Pump）是 Oracle 数据库提供的一种高性能、可扩展的数据导入和导出工具。它通过使用 expdp 和 impdp 命令，提供了比传统的导入和导出工具更强大和灵活的功能。

## 数据导出

数据导出（expdp）：expdp是Oracle 数据泵的导出工具，用于将数据库对象和数据导出到 .dmp 文件中。它可以选择性地导出整个数据库、特定的表空间、用户、模式或者指定的对象类型。同时，可以使用参数来过滤导出的数据，包括表、行、列级别的过滤。

## 数据导入

数据导入（impdp）：impdp 是 Oracle 数据泵的导入工具，用于从 .dmp 文件中恢复数据库对象和数据。它可以选择性地导入整个数据库、特定的表空间、用户、模式或者指定的对象类型。与 expdp 相同，也可以使用参数来过滤导入的数据。

## 并行处理

并行处理：数据泵支持并行处理，可以利用多个进程同时导出和导入数据。这极大地提高了数据导入和导出的速度和效率，尤其在大型数据库中。

## 压缩和加密

数据泵提供了压缩和加密的选项，可以在导出和导入过程中对数据进行压缩和加密。这有助于减小备份文件的大小，并增加数据传输的安全性。

# 数据库备份和还原

## exp命令

## impdp命令

## 备份

https://www.cnblogs.com/wendj/p/9849850.html#top

## 恢复指令

impdp Cloud/Cloud@orcl dumpfile=CLOUD.dmp full=y ignore=y

# 解释data block，extent 和 segment 的区别

data block 是数据库中最小的逻辑存储单元。当数据库的对象需要更多的物理存储空间时，连续的 data block 就组成了 extent.一个数据库对象拥有的所有extents 被称为该对象的 segment。

# 解释ORACLE BASE的区别

ORACLE BASE 是 oracle 的根目录，ORACLE HOME 是 oracle 产品的目录。

# Oracle分区

分区的实质是把一张大表的数据按照某种规则使用多张子表来存储。然后这多张子表使用统一的表名对外提供服务，子表实际对用户不可见。类似于在多张子表上建立一个视图，然后用户直接使用该视图来访问数据。

# Oracle 分区在什么情况下使用

当一张表的数据量到达上亿行的时候，表的性能会严重降低，这个时候就需要用到分区了，通过划分成多个小表，并在每个小表上建立本地索引可以大大缩小索引数据文件的大小，从而更快的定位到目标数据来提升访问性能。

分区除了可以用来提升访问性能外，还因为可以指定分区所使用的表空间，因此也用来做数据的生命周期管理。当前需要频繁使用的活跃数据可以放到访问速度更快但价格也更贵的存储设备上，而 2、3 年前的历史数据，或者叫冷数据可以放到更廉价、速度更低的设备上。从而降低存储费用。

# sql性能优化

1、对查询进行优化，应尽量避免全表扫描，首先应考虑在 WHERE 及 ORDER BY 涉及的列上建立索引。

2、尽量避免在 WHERE 子句中对字段进行 NULL 值判断，创建表时 NULL 是默认值，但大多数时候应该使用 NOT NULL，或者使用一个特殊的值，如 0，-1 作为默认值。（含NULL复合索引无效）

3、尽量避免在 WHERE 子句中使用 != 或 <> 操作符。MySQL 只有对以下操作符才使用索引：<，<=，=，>，>=，BETWEEN，IN，以及某些时候的 LIKE。

4、应尽量避免在 WHERE 子句中使用 OR 来连接条件，否则将导致引擎放弃使用索引而进行全表扫描，可以使用 UNION all合并查询：select id from t where num=10 union all select id from t where num=20。

5、IN 和 NOT IN 也要慎用，否则会导致全表扫描。对于连续的数值，能用 BETWEEN 就不要用 IN：select id from t where num between 1 and 3。

6、下面的查询也将导致全表扫描：select id from t where name like‘%abc%’ 或者select id from t where name like‘%abc’若要提高效率，可以考虑全文检索。而select id from t where name like‘abc%’才用到索引。

7、如果在 WHERE 子句中使用参数，也会导致全表扫描。

8、应尽量避免在 WHERE 子句中对字段进行表达式操作，应尽量避免在 WHERE 子句中对字段进行函数操作。

9、很多时候用 EXISTS 代替 IN 是一个好的选择：select num from a where num in(select num from b)。用下面的语句替换：select num from a where exists(select 1 from b where num=a.num)。

10、索引固然可以提高相应的 SELECT 的效率，但同时也降低了 INSERT 及 UPDATE 的效率。因为 INSERT 或 UPDATE 时有可能会重建索引，所以怎样建索引需要慎重考虑，视具体情况而定。一个表的索引数最好不要超过 6 个，若太多则应考虑一些不常使用到的列上建的索引是否有必要。

11、尽可能的使用 varchar, nvarchar 代替 char, nchar。因为首先变长字段存储空间小，可以节省存储空间，其次对于查询来说，在一个相对较小的字段内搜索效率显然要高些。

12、最好不要返回所有：select \* from t ，用具体的字段列表代替 “\*”，不要返回用不到的任何字段。

13、

# select \* 和select 1的区别

select \* from 表，查询出表中所有的数据，性能比较差。

select 常量 from 表；查询的结果是所有记录数的常量，性能比较高。

selelct 常量 from ... 对应所有行，返回的永远只有一个值，即常量。所以正常只会用来判断是否有还是没有（比如exists子句）。而select \* from ... 是返回所有行的所有列。如果要判断是否有结果使用select 1，如果要返回数据，使用select \* ；

# Select Count (\*)和Select Count(1)

一般情况下，Select Count (\*)和Select Count(1)两者返回的结果是一样的，如果表中没有主键 ，使用count(1)比count(\*)快。如果有主键，那么count(主键)最快，count(\*)和count(1)的结果一样，都包括对NULL的统计，而count(字段) 不包括NULL的统计；实操中选择使用 count(1)的情况比较多。

# CONNECT BY LEVEL

## 基本概念

在 Oracle 数据库中，CONNECT BY LEVEL 是一种用于生成序列或层次结构的特殊语法。它通常与 SELECT 语句一起使用，用于创建一个包含指定数量行或特定层次结构的结果集。

# rownum

# Oracel分页查询

## 1、采用伪列rownum

（1）查询前10条记录

select \* from t\_user t where ROWNUM <10;

（2）按照学生ID排名，抓取前三条记录

SELECT \* FROM(SELECT id,realname FROM T\_USER ORDER BY id asc ) WHERE ROWNUM <=3

（3）分页SQL写法，从第10条记录开始，提取10条记录。

SELECT \* FROM (SELECT ROWNUM rn,id,realname FROM (SELECT id,realname FROM T\_USER)WHERE ROWNUM<=20) t2 WHERE T2.rn >=10;

（4）按照学生ID排名，从第10条记录开始，提取10条记录。

SELECT \* FROM (SELECT ROWNUM rn,id,realname FROM (SELECT id,realname FROM T\_USER ORDER BY id asc)WHERE ROWNUM<=20) t2 WHERE T2.rn >=10;

### 注意事项

1、 where rownum>1 不能抓取到记录，返回的是空记录。

2、where rownum between 2 and 10 也不能抓取到记录。

## 2、运用分析函数

用分析函数row\_number()over(ORDER BY 字段)

按照学生ID排名，抓取前三条记录

SELECT \* FROM(SELECT id,realname,row\_number()over(ORDER BY id asc) rn FROM T\_USER)WHERE rn <=3

按照学生ID排名，从第10条记录开始，提取10条记录。

SELECT \* FROM(SELECT id,realname,row\_number()over(ORDER BY id asc) rn FROM T\_USER)WHERE rn BETWEEN 10 AND 20

## 3、运用minus方法

从第10条记录开始，提取10条记录。

SELECT \* FROM T\_USER WHERE ROWNUM<20 MINUS SELECT \* FROM T\_USER WHERE ROWNUM<10;

按ID排序后，从第10条记录开始，提取10条记录。

(SELECT \* FROM (SELECT \* FROM T\_USER ORDER BY id asc) WHERE ROWNUM<20) MINUS( SELECT \* FROM (SELECT \* FROM T\_USER ORDER BY id asc) WHERE ROWNUM<10);

# 递归查询

## 基本概念

START WITH...CONNECT BY 是 Oracle 数据库中用于处理递归查询的关键字。它结合使用了 START WITH 和 CONNECT BY 子句来构建递归查询。

在使用 START WITH...CONNECT BY 时，需要提供两个关键组成部分：

1、START WITH 子句：指定递归查询的起始条件。可以是一个表达式、列名或常量。它用于定义起始节点或行，作为递归查询的起点。

2、CONNECT BY 子句：定义递归查询的连接条件。它指定了如何将每个节点与其父节点相连接。一般使用 PRIOR 关键字来引用父节点。

## 语法

SELECT columns

FROM table

START WITH condition

CONNECT BY [NOCYCLE] condition;

其中，关键字说明如下：

columns：要查询的列。

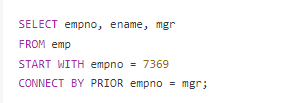
table：要查询的表。

condition：用于定义起始条件和递归关系的条件。

NOCYCLE（可选）：指定避免循环的选项。如果添加了 NOCYCLE 关键字，则会在递归查询中避免循环。

## 示例

假设有一个名为 EMP 的表，其中包含员工的 ID、姓名和经理 ID 等信息。我们想要查询员工及其直接下属的层次结构：



上述查询语句中，我们从 EMP 表中以 empno=7369 作为起始条件开始查询，然后使用 CONNECT BY PRIOR empno = mgr 来建立父节点和子节点之间的连接关系。查询结果将包含以 7369 员工为根节点的层次结构，每行都会显示员工的 empno、ename 和 mgr。通过这个查询，我们可以获取根节点员工 7369 及其所有下属员工的层次结构。

# 数据表中的结果显示（N/A）是什么意思

在 Oracle 数据表中，当某个字段的值为 (N/A) 时，通常表示该字段的数据不适用或无法提供。这是一种常见的占位符，用于表示缺失或未知的值。

1、缺失值：某个字段对于某些记录可能没有适用的值。比如，在人员信息表中，如果某个人的婚姻状态字段为空或不适用，则可以将其设为 (N/A)。

2、未知值：某个字段的值是未知的，无法确定具体的值。这可能是由于数据缺失、数据录入错误或者其他原因导致的。在这种情况下，可以使用 (N/A) 表示该字段的值未知。

3、不可用值：某个字段的值由于某种原因而无法提供。比如，在某个产品的销售记录中，如果某个字段是销售地点，但销售地点信息不适用或不可用，则可以使用 (N/A) 表示该字段的值无法提供。

# 问题

## 1、ORA-12514: TNS:listener does not currently know of service requested in connect descriptor

## 2、ORA-08002: sequence SEQ MIXCOAL APPROVAL.CURRVAL is not yet defined in this session

## 3、Result: ORA-00998: must name this expression with a column alias

ORA-00998 错误是 Oracle 数据库的一种错误类型，它表示必须为一个表达式命名一个列别名。当你在查询中使用了一个表达式（例如函数、计算或者别名）作为列，并且没有为该表达式指定一个列别名时，就会触发 ORA-00998 错误。

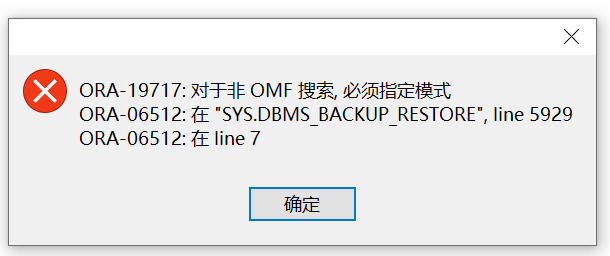
这个错误通常出现在 SELECT 语句中，例如：

## 4、ORA-01408: such column list already indexed

这个错误的原因是当前列已经创建了索引。

## 5、ORA-01031: insufficient privileges

## 6、对于非OMF搜索，必须指定模式



## 7、Cause: java.sql.SQLException: ORA-01422: 实际返回的行数超出请求的行数

## 8、Error -OCI NO DATA

在 Oracle 数据库中，OCI NO DATA 错误是指在使用 OCI（Oracle Call Interface）访问数据库时，未找到符合查询条件的数据。

OCI 是 Oracle 提供的一种用于与数据库进行通信的编程接口。当使用 OCI 进行数据库操作，例如执行查询语句时，如果查询结果为空，就会返回 OCI NO DATA 错误。

这个错误表示在执行查询操作后，没有找到满足查询条件的数据。可能存在以下几种情况导致 OCI NO DATA 错误：

1、查询条件不满足：如果查询语句中指定了某些条件，但数据库中没有符合这些条件的数据，就会触发 OCI NO DATA 错误。

2、数据已被删除或修改：在查询数据之前，如果数据已被删除或者被其他会话修改了，并且不再满足查询条件，那么也会导致 OCI NO DATA 错误。

当出现 OCI NO DATA 错误时，可以通过以下方式进行排查和解决：

1、检查查询条件：确认查询语句中的条件是否正确，并确保数据库中存在满足这些条件的数据。

2、检查数据库状态：确定查询操作之前的数据状态是否发生了变化，例如数据是否已被删除或修改。

3、错误处理：在代码中添加适当的错误处理机制，以便捕获并处理 OCI NO DATA 错误。可以根据具体需求进行处理，例如返回空结果集或抛出异常。

需要注意的是，OCI NO DATA 错误并不属于严重错误，它只是表示查询未返回任何数据。因此，在程序中遇到这个错误时，可以根据实际需求来决定如何处理。

## 9、oracle.jdbc.OracleDatabaseException: ORA-02019: 未找到远程数据库的连接说明

## 10、ORA-12514: TNS: 监听程序当前无法识别连接描述符中请求的服务

https://blog.csdn.net/oracleone1/article/details/120444609

## 11、ORA-01704:字符串文字太长

## 12、oracle.jdbc.OracleDatabaseException: ORA-04098: 触发器 'MMES.TR\_SCGL\_MGL\_LDLWKC' 无效且未通过重新验证

解决方法：

1、表示某个触发器无效或未通过重新验证。可能的原因包括：触发器依赖的对象被修改、触发器所引用的对象不存在、触发器中引用了无效或不存在的 PL/SQL 代码等。

2、根据错误信息显示，触发器名为 MMES.TR\_SCGL\_MGL\_LDLWKC，出现了无效且未通过重新验证的情况。这意味着该触发器在某个时间点被标记为无效状态，并且在此后没有通过重新验证使其恢复有效。触发器无效可能会导致数据库中某些操作无法正常执行，需要及时修复。

3、针对该错误，可以尝试重新验证触发器，或者检查触发器依赖的对象是否存在、是否被修改等。

## 13、ORA-01008:并非所有变量都已绑定