

# 数据库相关概念

## ▼ 一、事务处理和锁定

### ▼ （一）事务

#### ▼ 1、事务（Transaction）的定义

- （1）事务是由多条SQL语句组成的工作逻辑单元（没有比事务更小的逻辑），这些语句要么全部执行成功，要么全部不执行，只有有一条SQL执行失败，则已执行的SQL就会回滚到执行之前的状态，保证了数据库数据的一致性

#### ▼ （2）事务的结束

- 完成所有操作后COMMIT进行提交---用COMMIT区分事务
- 使用ROLL BACK回滚
- 退出ORACLE工具

#### ▼ 2、事务的特点

- （1）原子性：事务必须是原子工作单元（没有比事务更小的逻辑），在修改数据时，要么全部执行，要么全部不执行
- （2）一致性：事务在完成时，必须使所有的数据都保持一致状态，即所有的数据都要发生更改，以保证数据的完整性
- （3）隔离性：两个事务的执行是互不干扰的，一个事务不可能看到另一个事务运行时、运行中间某一时刻的数据
- （4）永久性：一旦事务被提交后，数据库数据的变化会永久保留，不会受外界因素影响

### ▼ （二）锁定

#### ▼ 1、锁定的定义

- 是数据库管理系统为确保数据一致性和完整性的重要机制
- Oracle提供了自动的锁定机制，来控制并发访问数据，使得A用户更新账户时，会锁定当前记录，不允许其他用户更改，保证数据的稳定性和一致性

#### ▼ 2、锁定的种类

- 表锁定（互斥锁、排它锁）：锁定整个表，确保当前事务不会被其他会话或事务造成冲突，用于保护整张表的数据，当执行INSERT、DELETE、SELECT FOR UPDATE语句时，将进行记录锁定
- 行锁定（记录锁定）：对当前操作的一行进行锁定
- 注：锁定总是以独占的方式进行，在一个事务结束之前，其他事务将要等待该事务结束

#### ▼ 3、死锁

- 死锁定义：两个会话，每个会话持有另一个会话想要的资源，因争夺资源而造成的互相等待的现象，此时就会出现死锁，若无外力，他们将推进下去，这些永远在互相等待的进程叫做死锁进程
- 死锁解决办法：回滚其中一个事务，让另一个事务顺利进行

## ▼ 二、数据库对象简介

- （一）数据库对象定义：是数据库的组成部分，可以使用CREATE创建，ALTER修改和DROP删除，被数据库承认的，有特定名称

## ▼ 二、数据库对象种类

### ▼ 1、表空间 (tablespace)

#### ▼ （1）表空间的定义

- 定义：表空间是一个逻辑概念，若干个操作系统文件（文件可以不是很大）组成一个表空间，表空间统一管理空间中的数据文件，所有的数据对象都存在指定表空间中

#### ▼ 表空间属性

- 一个数据库可以包含多个表空间，一个表空间只属于一个数据库
- 一个表空间包含多个数据文件，一个数据文件只属于一个表空间

#### ▼ （2）ORACLE数据库存储结构

- 从逻辑上看：一个数据库 (database) --分为多个表空间 (tablespace) --一个表空间分为多个段 (segment) (一个表和一个索引要分别占一个段) ---一个段由多个区间 (extent) 组成 --一个区间由一组连续的数据块 (data block) 组成
- 从物理上看：一个表空间由多个数据文件组成，数据文件是实实在在的磁盘上的文件，这些文件由ORACLE数据库操作系统的block组成
- ● 这连续的数据块在逻辑上是连续的，有可能在物理磁盘上是分散的

#### ▼ （3）常见表空间

##### ▼ 1、系统表空间 (SYSTEM)

- 存放系统数据，在数据库创建时创建
- 存在数据字典、视图以及数据库结构等重要系统数据信息，
- 在运行时，system空间不足，对数据库影响很大，可以通过命令扩充，但还是会影响数据库性能，所以有必须在创建时适当把数据文件设置得大一些

##### ▼ 2、TEMP (临时) 表空间

- 安装数据库时创建，可以通过命名扩大临时表空间
- 临时表空间的重要作用数据排序
- 当服务器需要对数据进行排序时，如果数据很大，内存的排序区可能装不下大数据，就需要把一些中间的排序结果写在硬盘的临时表空间中

##### ▼ 3、用户表自定义空间：用户可以通过CREATE TABLESPACE命令创建表空间

## ▼ 2、同义词 (synonymy)




- (1) 定义：数据库对象的一个别名，ORACLE可以为表、视图、序列、过程、函数、程序包等指定一个别名

### ▼ (2) 同义词种类

- 私有同义词：拥有create synonymy权限的用户（包括非管理员用户）即可创建私有同义词，创建的私有同义词只能由当前的用户使用
- 公有同义词：系统管理员可以创建共有同义词，共有同义词可以被所有用户使用

### ▼ (3)同义词的创建和使用

#### ▼ 同义词创建

-  oracle普通用户无创建同义词权限，需要授权：GRANT CREATE SYNONYM TO SCOTT;
-  使用create synonymy语句，前提需要授权，语法：CREATE [PUBLIC] SYNONYM synonym for object
- 语法中关键字的含义如下所示。
  -  PUBLIC：创建一个公共同义词，可以被所有的用户访问。
  - synonym：要被创建的同义词的名字。
  - object：指出要创建同义词的对象。
- 同义词使用：同义词被创建后，就可以像使用本地表一样对scottemp进行操作
- 删除同义词：如果不再需要同义词，可以使用DROP SYNONYM删除同义词。只有数据库管理员可以删除一个公共同义词

代码演示：

```
DROP PUBLIC SYNONYM scottemp;
```

## ▼ 3、序列 (sequence)

- (1) 定义：序列是用来生成连续的整数数据的数据库对象，其主要工作是为表产生唯一值

### ▼ (2) 作用

- 序列的一个典型的用途是创建一个主键的值，它对于每一行必须是唯一的。序列中的可以升序生成，也可以降序生成
- 序列被创建后可以通过数据字典找到序列对象，因此序列可以被多个对象共享

### ▼ (3) 序列的创建和使用

#### ▼ 序列的创建

- 语法

```
CREATE SEQUENCE sequence_name  
[START WITH num]  
[INCREMENT BY increment]  
[MAXVALUE num|NOMAXVALUE]  
[MINVALUE num|NOMINVALUE]  
[CYCLE|NOCYCLE]  
[CACHE num|NOCACHE]
```

- 语法解析：

- ① START WITH：从某一个整数开始，升序默认值是1，降序默认值是-1。
- ② INCREMENT BY：增长数。如果是正数则升序生成，如果是负数则降序生成。升序默认值是1，降序默认值是-1。
- ③ MAXVALUE：指最大值。
- ④ NOMAXVALUE：这是最大值的默认选项，升序的最大值是：1027，降序默认值是-1。
- ⑤ MINVALUE：指最小值。
- ⑥ NOMINVALUE：这是默认值选项，升序默认值是1，降序默认值是-1026。
- ⑦ CYCLE：表示如果升序达到最大值后，从最小值重新开始；如果是降序序列，达到最小值后，从最大值重新开始。
- ⑧ NOCYCLE：表示不重新开始，序列升序达到最大值、降序达到最小值后就报错。默认NOCYCLE。
- ⑨ CACHE：使用CACHE选项时，该序列会根据序列规则预生成一组序列号。保留在内存中，当使用下一个序列号时，可以更快的响应。当内存中的序列号用完时，系统再生成一组新的序列号，并保存在缓存中，这样可以提高生成序列号的效率。Oracle默认会生产20个序列号。
- ⑩ NOCACHE：不预先在内存中生成序列号。

- 序列的使用：序列创建后，可以通过序列对象的CURRVAL 和NEXTVAL两个“伪列”， 分别访问该序列的当前值或下一值

代码演示：

---访问下一个值

```
SELECT MYSEQ.NEXTVAL FROM DUAL;
```

---访问当前值

```
SELECT MYSEQ.CURRVAL FROM DUAL;
```

- ▼ (4) 序列的修改和删除

- ▼ 序列的修改

- 使用ALTER SEQUENCE 来修改

代码演示：

---序列修改

```
ALTER SEQUENCE MYSEQ  
MAXVALUE 10005  
MINVALUE -300
```

- ▼ 修改受限

- a. 不能修改序列的初始值
- b. 最小值不能大于当前值

- c. 最大值不能小于当前值

- 序列的删除：使用drop sequence 来删除序列

代码演示：

--删除序列

```
DROP SEQUENCE MYSEQ;
```

#### ▼ 4、视图 (VIEW)

- (1) 视图的定义：是一张或多张表上的预定义查询，这些表称为基表，可以通过视图查询信息，和从表中查询信息完全一样

##### ▼ (2) 视图的优点

- 可以限制用户只能通过视图检索数据，这样可以对最终用户屏蔽建表时底层的基表，具有安全性
- 将复杂的查询保存为视图，屏蔽复杂性

##### ▼ (3) 视图的创建和使用

###### ▼ 视图的创建

- ORACLE普通用户无创建视图权限，需授权：GRANT CREATE VIEW TO SCOTT;

- 视图创建语法：

```
CREATE [OR REPLACE] [{FORCE|NOFORCE}] VIEW view_name  
AS  
SELECT 查询  
[WITH READ ONLY CONSTRAINT]
```

- 语法解析：

1. OR REPLACE：如果视图已经存在，则替换旧视图。
2. FORCE：即使基表不存在，也可以创建该视图，但是该视图不能正常使用，当基表创建成功后，视图才能正常使用。
3. NOFORCE：如果基表不存在，无法创建视图，该项是默认选项。
4. WITH READ ONLY：默认可以通过视图对基表执行增删改操作，但是有很多在基表上的限制（比如：基表中某列不能为空，但是该列没有出现在视图中，则不能通过视图执行insert操作），WITH READ ONLY说明视图是只读视图，不能通过该视图进行增删改操作。现实开发中，基本上不通过视图对表中的数据进行增删改操作。

- 代码演示：


- 视图的使用

- 注：视图不存储数据

#### ▼ 5、索引

##### ▼ (1) 什么是索引

- 索引扫描逻辑：创建索引对某些特定列的数据排序，生成独立的索引表，查询索引列时，会先从索引表中查询出符合条件记录的ROWID，再通过ROWID快速定位到具体记录，可以提升查询效率。

- ORACLE 选择机制：当某列出现在查询条件中，ORACLE 会比较全表扫描和索引扫描的代价，选择代价小的扫描方式
- 索引可以提高查询的效率，但是在数据增删改时需要更新索引，因此索引对增删改时会有负面影响
- ▼ (2) 什么时候要建索引
  - a. 表中某些字段经常被查询或作为查询条件出现时，考虑对该列创建索引
  - b. 数据量大，单个查询要检索的行数少于或等于整个表行数10%时，考虑索引，如果返回的数据量很大，索引的性能就很差
  - c. 经常需要做连接的列，这些列主要是一些外键，可以加快连接速度
  - d. 经常需要排序的列，因为索引已经排序了，可以加快排序查询时间
  -  e. Oracle数据库会对表的主键和包含唯一约束的列自动创建索引
  - f. 需要根据范围进行搜索的列，因为索引已经排序，其指定的范围的连续的
- ▼ (3) 什么时候不需要索引
  - a. 查询中很少使用或参考的列，增加索引并不能提高效率，反而会降低系统的维护速度和增大空间需求
  - b. 数据值很少的列
  - c. 对于那些定义为text, image和bit数据类型的列不应该增加索引。这是因为，这些列的数据量要么相当大，要么取值很少
  - e. 当修改性能远远大于检索性能时，不应该创建索引，因为修改性能和检索性能是互相矛盾的
- ▼ (4) 索引的优点和缺点
  - ▼ 优点
    - 第一、创建唯一索引，可以保证数据库表中每一行的数据唯一
    - 第二、可以大大加快数据的检索速度
    - 第三、可以加快表和表的连接
    - 第四、在使用分组和排序子查询进行数据检索，可以显著减少查询的时间
  - ▼ 缺点
    - 第一、创建索引和维护索引需要消耗大量时间，这种时间随着索引的数量增加而增加
    - 第二、索引需要占用物理空间（存储空间）
    - 第三、对表进行增删改时，也需要对索引进行动态维护，降低了数据维护速度，
    - 如果一张表20个索引，往里面写入100万条数据，怎么优化？  
先把索引全删了，再把记录写进去，再建立索引。
- ▼ (5) 索引的种类
  - ▼ a. 唯一索引（用的最多）

- 何时建：任意两行的值都不相同并且不为空时
- 注：ORACLE 数据库会为表的主键和唯一约束列自动创建唯一索引
- 创建语法：CREATE UNIQUE INDEX 索引名 ON 表名 (字段)
- ▼ b. 位图索引
  - 何时创建：列中有非常多的重复值时，WHERE 条件中包含了很多OR时
  - 创建语法：CREATE BITMAP INDEX 索引名 ON 表名 (字段)
  - 注：位图索引性能比较差
- ▼ c. 组合索引
  - 何时创建：当两个或多个列经常出现在WHERE条件中时，则对这些列同时创建索引，组合索引列的顺序是任意的，无需相邻。但是建议将最频繁访问的列放在列表的最前面
  - 创建语法：CREATE INDEX 索引名 ON 表名 (字段1, 字段2,...)
- ▼ d. 基于 函数的索引
  - 何时创建：在WHERE 条件语句中包含函数或表达式时，因为索引在遇到函数时将无法识别导致索引失效
  - 函数包括：算数表达式、PL/SQL函数、程序包函数、SQL函数、用户自定义函数。
  - 创建语法：CREATE INDEX 索引名 ON表名 (函数)
- e. 其他索引种类：反向键索引、键压缩索引、索引组织表、分区索引等
- ▼ (6) 创建索引的语法
  - 语法结构：CREATE [UNIQUE] INDEX index\_name ON table\_name(column\_name[,column\_name...])
  - 语法解析：
    - 1.UNIQUE:指定索引列上的值必须是唯一的。称为唯一索引。
    - 2.index\_name：指定索引名。
    - 3.tabl\_name：指定要为哪个表创建索引。
    - 4.column\_name：指定要对哪个列创建索引。我们也可以对多列创建索引；这种索引称为组合索引。
- ▼ (7) 索引失效
  - 第一、隐式转换（比如：系统自动会将数据类型隐式转为为字符串类型）
 

由于表的字段tu\_mdn定义为varchar2(20),但在查询时把该字段作为number类型？以where条件传给Oracle,这样会导致索引失效.

错误的例子：select \* from test where tu\_mdn=13333333333;

正确的例子：select \* from test where tu\_mdn='13333333333';
  - 第二、对索引列进行算术运算、拼接等，算术运算包括（+、-、\*、/、！等）
  - 第三、使用ORACLE内部函数导致失效，这种情况应创建函数索引
  - ▼ 第四、其他应避免使用导致失效

- 使用<>,!=
- 模糊查询 like 的“%\_” 百分号在前的情况
- 字符型字段为数字时, 在where 条件里不添加引号
- 当变量采用的是times变量, 而表的字段采用的是date变量时.或相反情况

## ▼ 6、表分区

- (1) 表分区的定义: 将表中的数据在物理上存放到一个或多个表空间(物理文件上), 表进行分区后, 这样在查询数据时不至于扫描整张表, 逻辑上仍然是一张完整的表。

### ▼ (2) 表分区的作用

- 第一、通过改善可管理性、性能和可用性, 从而为各式应用程序带来了极大的好处
- 第二、分区可以使某些查询以及维护操作的性能大大提高
- 第三、极大简化常见的管理任务, 分区是构建千兆字节数据系统或超高可用性系统的关键工具。
- 第四、分区功能能够将表、索引或索引组织表进一步细分为段, 这些数据库对象的段叫做分区。每个分区有自己的名称, 还可以选择自己的存储特性。从数据库管理员的角度来看, 一个分区后的对象具有多个段, 这些段既可进行集体管理, 也可单独管理, 这就使数据库管理员在管理分区后的对象时有相当大的灵活性。但是, 从应用程序的角度来看, 分区后的表与非分区表完全相同, 使用DML命令访问分区后的表时, 无需任何修改。

### ▼ (3) 什么时候使用表分区


- 1)表的数据量特别大, 超过2GB
- 2)表中包含历史数据, 新的数据被增加到新的分区中。

### ▼ (4)表分区的有点和缺点

#### ▼ 优点

- 1)改善查询性能: 对分区对象的查询可以仅搜索自己关心的分区, 提高检索速度。
- 2)增强可用性: 如果表的某个分区出现故障, 表在其他分区的数据仍然可用;
- 3)维护方便: 如果表的某个分区出现故障, 需要修复数据, 只修复该分区即可;
- 4)均衡I/O: 可以把不同的分区映射到磁盘以平衡I/O, 改善整个系统性能。

#### ▼ 缺点

- 分区表相关, 已经存在的表没有方法可以直接转化为分区表。
- 需要维护。
-  在创建表时, 就要做分区

### ▼ (5) 表分区的种类



▼ 1) 范围分区：RANGE，最常见，而且分区键常采用日期

- 定义：范围分区将数据基于范围映射到每一个分区，这个范围是你在创建分区时指定的分区键决定的。这种分区方式是最为常用的，并且分区键经常采用日期。举个例子：你可能会将销售数据按照月份进行分区。

▼ 分区规则

- 1) 每一个分区都必须有一个VALUES LESS THEN子句，它指定了一个不包括在该分区中的上限值。分区键的任何值等于或者大于这个上限值的记录都会被加入到下一个高一些的分区中。
- 2) 所有分区，除了第一个，都会有一个隐式的下限值，这个值就是此分区的前一个分区上限值。
- 3) 在最高的分区中，MAXVALUE被定义。MAXVALUE代表了一个不确定的值。这个值高于其它分区中的任何分区键的值，也可以理解为高于任何分区中指定的VALUE LESS THEN的值，同时包括空值。

▪ 代码演示

---创建范围分区 (RANGE)

CREATE TABLE EMP\_RANGE ----分区要在表创建时进行分区

```
(EMPNO NUMBER,  
ENAME VARCHAR(20),  
JOB VARCHAR(10),  
MGR NUMBER,  
HIREDATE DATE,  
SAL NUMBER(7,2),  
COMM NUMBER(7,2),  
DEPTNO NUMBER(2)  
)
```

PARTITION BY RANGE(HIREDATE)

```
(  
PARTITION D1981 VALUES LESS THAN (TO_DATE('1981-1-1', 'YYYY/MM/DD')),  
PARTITION D1982 VALUES LESS THAN (TO_DATE('1982-1-1', 'YYYY/MM/DD')),  
PARTITION D1988 VALUES LESS THAN (TO_DATE('1988-1-1', 'YYYY/MM/DD')),  
PARTITION D_MAX VALUES LESS THAN (MAXVALUE)  
);
```

▼ 2) 列表分区：LIST

- 使用范围：列的值只有几个，基于这样的特点我们可以采用列表分区，即某列重复数据多时

- 代码演示

- 按DEPTNO, 创建列表分区的表

- ```
CREATE TABLE EMP_LIST(EMPNO NUMBER,
    ENAME VARCHAR2(10),
    JOB VARCHAR2(10),
    MGR NUMBER,
    HIREDATE DATE,
    SAL NUMBER(7,2),
    COMM NUMBER(7,2),
    DEPTNO NUMBER(2)
)
PARTITION BY LIST(DEPTNO)
(PARTITION D10 VALUES(10),
PARTITION D20 VALUES(20),
PARTITION D30 VALUES(30),
PARTITION D40 VALUES(40)
);
```

- ▼ 3) 散列（哈希）分区：hash

- 使用规则：当列的值没有合适条件时，通过指定分区编号来均匀分布数据，即按数据量分区，对某列的数据做等量分区，分区数量一般为2的n次方

- 代码演示

- 根据EMPNO创建散列分区表

- ```
CREATE TABLE EMP_HASH(EMPNO NUMBER,
    ENAME VARCHAR2(10),
    JOB VARCHAR2(10),
    MGR NUMBER,
    HIREDATE DATE,
    SAL NUMBER(7,2),
    COMM NUMBER(7,2),
    DEPTNO NUMBER(2)
)
PARTITION BY HASH(EMPNO) PARTITIONS 16;
```

- ▼ 4) 组合分区

- 使用范围：这种分区是基于两种分区的组合，大分区包含小分区，分区之中的分区被称为子分区。

- 代码演示

- ▼ (6) 表分区的维护性操作

- ▼ 1) 添加表分区

- 给表添加分区

- 代码演示：

- 以下代码给SALES表添加了一个P3分区

- ```
ALTER TABLE SALES ADD PARTITION P3 VALUES LESS
THAN(TO_DATE('2003-06-01','YYYY-MM-DD'));
```

- 给分区添加子分区

以下代码给SALES表的P3分区添加了一个P3SUB1子分区

```
ALTER TABLE SALES MODIFY PARTITION P3 ADD SUBPARTITION P3SUB1  
VALUES('COMPLETE');
```

- 注意：以上添加的分区界限应该高于最后一个分区界限

- ▼ 2) 删除分区

- 删除分区

以下代码删除了P3表分区：

```
ALTER TABLE SALES DROP PARTITION P3;
```

- 删除子分区

在以下代码删除了P4SUB1子分区：

```
ALTER TABLE SALES DROP SUBPARTITION P4SUB1;
```

- 注意：如果删除的分区是表中唯一的分区，那么此分区将不能被删除，要想删除此分区，必须删除表

- ▼ 3) 截断分区

- 截断某个分区是指删除某个分区中的数据，并不会删除分区，也不会删除其它分区中的数据。当表中即使只有一个分区时，也可以截断该分区

- 截断分区

通过以下代码截断分区：

```
ALTER TABLE SALES TRUNCATE PARTITION P2;
```

- 截断子分区

通过以下代码截断子分区：

```
ALTER TABLE SALES TRUNCATE SUBPARTITION P2SUB2;
```

- ▼ 4) 合并分区

- 合并分区是将相邻的分区合并成一个分区，结果分区将采用较高分区的界限，值得注意的是，不能将分区合并到界限较低的分区。

- 合并分区

以下代码实现了P1 P2分区的合并：

```
ALTER TABLE SALES MERGE PARTITIONS P1,P2 INTO PARTITION P2;
```

- ▼ 5) 拆分区

- 拆分区将一个分区拆分两个新分区，拆分后原来分区不再存在。注意不能对HASH类型的分区进行拆分

- 代码演示

```
ALTER TABLE SALES SPLIT PARTITION P2 AT(TO_DATE('2003-02-01','YYYY-MM-DD')) INTO (PARTITION P21,PARTITION P22);
```

- ▼ 6) 接合分区(coalesce)

- 结合分区是将散列分区中的数据接合到其它分区中，当散列分区中的数据比较大时，可以增加散列分区，然后进行接合，值得注意的是，接合分区只能用于散列分区中。

- 代码演示

通过以下代码进行接合分区：

```
ALTER TABLE SALES COALESCE PARTITION;
```

- 7)重命名表分区

代码演示

以下代码将P21更改为P2

```
ALTER TABLE SALES RENAME PARTITION P21 TO P2;
```

- 7、其他常见数据库对象：表、用户