SQL进阶

一、数据清洗常用方法

1、连接字符串

```
SELECT CONCAT(user_id,item_id) FROM taobao_user;
```

2、大小写转换

```
--小写转大写
SELECT t.*,UPPER(user_geohash)
FROM taobao_user t;

--大写转小写
SELECT t.*,LOWER(user_geohash)
FROM taobao_user t;
```

3、替换

```
--字符串替换
SELECT t.*,REPLACE((user_geohash),'nn','NN')
FROM taobao_user t;

```

4、截取某段字符

```
SELECT t.*,SUBSTR(user_id,1,2)  --表示从user_id字段中的第1位截取到第2位
FROM taobao_user t;
SELECT SUBSTR('Hello SQL!', -4) FROM dual    --从倒数第4个字符开始，截取到末尾。返回'SQL!'
SELECT SUBSTR('Hello SQL!', -4, 3) FROM dual --从倒数第4个字符开始，截取3个字符。返回'SQL'
```

5、重复值（删除）

```
DELETE FROM c_emp
WHERE ROWID NOT IN
(SELECT MIN(ROWID) FROM c_emp GROUP BY deptno);
```

6、空值

```sql
--空值替换
SELECT t.*,NVL(user_geohash,'none')
FROM taobao_user t;

--空值剔除（行）
SELECT *
FROM taobao_user
WHERE user_id IS NOT NULL
AND item_id IS NOT NULL
AND behavior_type IS NOT NULL
AND user_geohash IS NOT NULL
AND item_category IS NOT NULL
AND TIME IS NOT NULL;
```

7、数据类型转换

```sql
SELECT to_char(DATE'2020-02-20','YYYYMMDD') FROM dual;
SELECT to_char(DATE'2020-02-20','YYYY') FROM dual;
SELECT to_char(DATE'2020-02-20','MM') FROM dual;
SELECT to_char(DATE'2020-02-20','Q') FROM dual; --返回季度
SELECT to_char(DATE'2020-08-9','WW') FROM dual; --返回第几周
SELECT to_char(hiredate,'YYYY') FROM emp;
SELECT to_number(to_char(hiredate,'YYYY')) FROM emp;
```

8、查看删除的表

```sql
SELECT * FROM RECYCLEBIN;
```

9、恢复删除的表

```sql
FLASHBACK TABLE emp TO BEFORE DROP;
```

二、数据库操作

1、增量更新--merge into

```sql
--语法格式
/*
merge into [target-table] A
   using [source-table sql] B
   on ([conditional expression] and [...]...)
when matched then      -- 当on中的条件匹配时
    [update sql]       -- 执行操作    更新或删除等
when not matched then  -- 当on中的条件不匹配时
    [insert sql]       -- 执行操作   新增等
*/

--实例
MERGE INTO cc_emp a
```

```
USING(SELECT * FROM c_emp) b
ON (a.ename=b.ename)
WHEN MATCHED THEN UPDATE SET
  a.empno=b.empno,
  a.job=b.job,
  a.mgr=b.mgr,
  a.hiredate=b.hiredate,
  a.sal=b.sal,
  a.comm=b.comm,
  a.deptno=b.deptno
WHEN NOT MATCHED THEN INSERT
  (a.empno,a.ename,a.job,a.mgr,a.hiredate,a.sal,a.comm,a.deptno)
  VALUES
  (b.empno,b.ename,b.job,b.mgr,b.hiredate,b.sal,b.comm,b.deptno);

 --注意两点：1、on中相关匹配的字段不能是重复值，on中出现的字段在when matched then中不能再次出现
```

2、自增序列--sequence

(1)语法结构

```
--创建序列的语法 --
 create sequence [user.]sequence_name
    [increment by n]
    [start with n]
    [maxvalue n | nomaxvalue]
    [minvalue n | nominvalue];

--修改序列的语法--
alter sequence [user.]sequence_name
    [increment by n]
    [maxvalue n | nomaxvalue]
    [minvalue n | nominvalue];

--删除序列
drop sequence seq_emp;

--查看序列
select * from user_sequences;

INCREMENT BY:  指定序列号之间的间隔，该值可为正的或负的整数，但不可为0。序列为升序。忽略该子句时，缺省值为1。
START WITH:指定生成的第一个序列号。在升序时，序列可从比最小值大的值开始，缺省值为序列的最小值。对于降序，序列可由比最大值小的值开始，缺省值为序列的最大值。
MAXVALUE:指定序列可生成的最大值。
NOMAXVALUE:为升序指定最大值为1027，为降序指定最大值为-1。
MINVALUE:指定序列的最小值。
NOMINVALUE:为升序指定最小值为1。为降序指定最小值为-1026。


```

(2)运用举例

```
--01创建示例表 --
```

```
create table Student(
    stuId number(9) not null,
    stuName varchar2(20) not null,
    stuMsg varchar2(50) null
)

-- 02创建序列  Student_stuId_Seq --
create sequence Student_stuId_Seq
increment by 1
start with 1
minvalue 1
maxvalue 999999999;

 --03调用序列，插入Student数据 --
insert into Student(stuId,Stuname) values(Student_stuId_Seq.Nextval,'张三');

insert into Student(stuId,Stuname) values(Student_stuId_Seq.Nextval,'李四');
```

3、临时表

（1）会话级临时表

```
--语法格式：
CREATE GLOBAL TEMPORARY TABLE TABLE_NAME
(COL1 TYPE1,COL2 TYPE2...)
ON COMMIT PRESERVE ROWS;
```

```
--实例
--01创建会话临时表
CREATE GLOBAL TEMPORARY TABLE tmp_STUDENT
    (STU_ID NUMBER(5),
    CLASS_ID  NUMBER(5),
    STU_NAME VARCHAR2(8),
    STU_MEMO VARCHAR2(200)) ON COMMIT PRESERVE ROWS ;
--02插入数据
    INSERT INTO tmp_STUDENT VALUES(1,2,'A','随便');
    INSERT INTO tmp_STUDENT VALUES(2,3,'B','OK');
    COMMIT;
    SELECT * FROM tmp_STUDENT;
--03创建目标表（用于最终保存有用数据）
    CREATE TABLE STUDENT_tmp_cp
    (STU_ID NUMBER(5),
    CLASS_ID  NUMBER(5),
    STU_NAME VARCHAR2(8),
    STU_MEMO VARCHAR2(200));
--04将临时表中有用数据插入目标表
    INSERT INTO STUDENT_tmp_cp
    SELECT *
    FROM tmp_STUDENT
    WHERE STU_MEMO='OK';


    SELECT * FROM tmp_STUDENT;
    SELECT * FROM STUDENT_tmp_cp;
```

（2）事务级临时表

```
--语法格式：
CREATE GLOBAL TEMPORARY TABLE TABLE_NAME
(COL1 TYPE1,COL2 TYPE2...)
ON COMMIT DELETE ROWS;
```

```
--运用举例
-- 支付给商家,商家收到钱以及支付方被扣款
  CREATE GLOBAL TEMPORARY TABLE tmp_pay_customer
  (id NUMBER(5),
  pay_mon VARCHAR2(8)
  ) ON COMMIT DELETE ROWS;

  CREATE GLOBAL TEMPORARY TABLE tmp_get_merchant
  (id NUMBER(5),
  get_mon VARCHAR2(8)
  ) ON COMMIT DELETE ROWS;

  INSERT INTO tmp_pay_customer VALUES(1,-100);
  SELECT * FROM tmp_pay_customer;
  COMMIT; -- 确认支付
  -- 商家收款
  INSERT INTO tmp_get_merchant VALUES(1,100);
```

4、多表插入--insert all

多条插入

```
CREATE TABLE t2 (id NUMBER(2),name VARCHAR2(20));

--插入数据
    INSERT ALL
      INTO t2
    VALUES (1, '张美丽')
      INTO t2
    VALUES (2, '王小二')
    SELECT * FROM dual;  -- 语句结构必须要select ...
    COMMIT;
```

多表插入

```
INSERT ALL
        WHEN product_id = 111 THEN
            INTO apple_orders
        WHEN product_id = 222 THEN
            INTO orange_orders
        ELSE
            INTO banana_orders
        SELECT product_id, product_name, p_month
          FROM t1;
    COMMIT;
```

5、行列转换

（1）行转列--pivot&case when

```
--语法格式
--语法：pivot(任一聚合函数(列) for 需要转的列的值所在列名 in (需转为列名的值))；
```

```
--实例
--方法1
SELECT * FROM chengji
pivot(max(score) for course in(
    '语文' as 语文,
    '数学' as 数学,
    '英语' as 英语,
    '化学' as 化学,
    '历史' as 历史
))
ORDER BY id;

--方法2（优先使用）
SELECT ID,NAME,
SUM(CASE WHEN course='语文' THEN score END) AS 语文,
SUM(CASE WHEN course='数学' THEN score END) AS 数学,
SUM(CASE WHEN course='英语' THEN score END) AS 英语,
SUM(CASE WHEN course='历史' THEN score END) AS 历史,
SUM(CASE WHEN course='化学' THEN score END) AS 化学,
SUM(score) AS 总成绩
FROM chengji
GROUP BY ID,NAME
ORDER BY ID;
```

（2）列转行--unpivot(略)

```
--语法：unpivot(新的列名1[多列中的值转为新增列中值] for 新的列名2[解释:多列的列名转为新增列中值] in (需转为行的列名))；
```

SQL调优

一、常用查看语句

1、查看表结构

```
select * from user_tab_columns where table_name='TAOBAO_USER'
注意：表明要大写
```

2、查看单张表数据量

```
SELECT COUNT(*) FROM taobao_user
```

3、显示所有表的表名和行数

```
SELECT table_name,num_rows FROM user_tables order by num_rows desc;
```

4、显示表名

```
SELECT * FROM USER_TABLES --查看当前用户下的表
SELECT * FROM DBA_TABLES --查看数据库中所有的表
```

5、查看当前执行的sql的日志

```
select * from v$logfile;
```

二、常用调优手段

1、增加索引

```
--创建默认的B-树索引（常用）
create index idx_emp on emp(empno);
--复合索引
create index idx1_emp on emp(deptno,empno);
--查看索引
select * from all_ind_columns where table_name='EMP';
--删除索引
drop index idx_emp;
```

2、HINT调优

```
--Hint调优，使用parallel提高并行数量（一般为2，4，6，8，10）
--/*+parallel(表名1，并行数)[(表名2，并行数).....]*/
select /*+parallel(emp,4)*/ * from emp where deptno=10;
```

3、创建视图

(1)普通视图

```
--创建普通视图
create or replace view v_emp
as
select ename,empno,sal
from emp;
--删除视图
drop view v_emp;
```

(2)物化视图

```
--创建物化视图（仅仅介绍自动刷新的on commit方式的物化视图）
create materialized view mv_emp
refresh force on commit
as
select *
from emp;
--删除物化视图
drop materialized view mv_emp;
```

4、用Exists改写子查询

```
select *
from T1
where exists(select 1 from T2 where T1.a=T2.a)

--当T1数据量小而T2数据量非常大时，T1<<T2 时) 的查询效率高。
--“select 1”这里的 “1”其实是无关紧要的，换成“*”也没问题

SELECT * FROM EMP A
WHERE EXISTS
(SELECT B.SAL
 FROM EMP B
 WHERE A.SAL = B.SAL
 AND B.DEPTNO=30);
```