[查看Oracle中表的索引是否存在](http://blog.csdn.net/u011742227/article/details/24880357)

标签： [oracle](http://www.csdn.net/tag/oracle)[数据库](http://www.csdn.net/tag/%e6%95%b0%e6%8d%ae%e5%ba%93)[sql](http://www.csdn.net/tag/sql)[索引](http://www.csdn.net/tag/%e7%b4%a2%e5%bc%95)

2014-05-02 16:58 291人阅读 [评论](http://blog.csdn.net/u011742227/article/details/24880357#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/u011742227/article/details/24880357#report)

 分类：



Oracle（26）



用user\_indexes和user\_ind\_columns系统表查看已经存在的索引

对于系统中已经存在的索引我们可以通过以下的两个系统视图(user\_indexes和user\_ind\_columns)来查看其具体内容，例如是属于那个表，哪个列和，具体有些什么参数等等。

user\_indexes:     系统视图存放是索引的名称以及该索引是否是唯一索引等信息。

user\_ind\_column:  系统视图存放的是索引名称，对应的表和列等。

查看索引个数和类别:

SQL> select \* from user\_indexes where table='表名' ;

查看索引被索引的字段:

SQL> select \* from user\_ind\_columns where index\_name=upper('&index\_name');

我们可以通过类似下面的语句来查看一个表的索引的基本情况：

select user\_ind\_columns.index\_name,user\_ind\_columns.column\_name,

user\_ind\_columns.column\_position,user\_indexes.uniqueness

from user\_ind\_columns,user\_indexes

where user\_ind\_columns.index\_name = user\_indexes.index\_name

and user\_ind\_columns.table\_name = ‘你想要查询的表名字’;

通过这条SQL语句我们能查看到一个表的具体的索引的情况，如果你想对这表的索引进行进一步的探究你应该到user\_indexes中去具体的看以下这个索引的基本情况。

完整性约束  
　　DBA\_CONSTRAINTS、ALL\_CONSTRAINTS和USER\_CONSTRAINST  显示有关约束的一般信息。  
　　DBA\_CONS\_COLUMNS、ALL\_CONS\_COLUMNS和USER\_CONS\_COLUMNS 显示有关列的相关约束的一般信息。

ALL\_CONS\_COLUMNS 视图和DBA\_CONS\_COLUMNS 视图与USER\_CONS\_COLUMNS有相同的列定义。

ALL\_CONS\_COLUMNS 视图能够显示用户可以访问的所有表上约束的列信息，而不管所有者是谁。  
DBA\_CONS\_COLUMNS 视图列出了整个[**数据库**](http://lib.csdn.net/base/mysql)的列级约束信息。  
USER\_CONS\_COLUMNS

user\_constraints 和 user\_cons\_columns表得作用及其联系

user\_constraints:  是表约束的视图,描述的是约束类型(constraint\_type)是什么,属于哪些表(table\_name),如果约束的类型为R(外键)的话,那么r\_constraint\_name字段存放的就是被引用主表中的主键约束名。

user\_cons\_columns: 是表约束字段的视图,说明表中的和约束相关的列参与了哪些约束。这些约束有主键约束,外键约束,索引约束.

两者可以通过(owner,constraint\_name,table\_name)关联:

select   
a.owner 外键拥有者,   
a.table\_name 外键表,   
substr(c.column\_name,1,127) 外键列,   
b.owner 主键拥有者,   
b.table\_name 主键表,   
substr(d.column\_name,1,127) 主键列   
from   
user\_constraints a,   
user\_constraints b,   
user\_cons\_columns c,   
user\_cons\_columns d   
where   
    a.r\_constraint\_name=b.constraint\_name   
and a.constraint\_type='R'   
and b.constraint\_type='P'   
and a.r\_owner=b.owner   
and a.constraint\_name=c.constraint\_name   
and b.constraint\_name=d.constraint\_name   
and a.owner=c.owner   
and a.table\_name=c.table\_name   
and b.owner=d.owner   
and b.table\_name=d.table\_name

数据字典表列说明：

desc user\_constraints

Name                                                                                   Comments                                                                      
-----------------                ---------------------------------------------------------------------------   
OWNER                                                                   Owner of the table                                                            
CONSTRAINT\_NAME                                             Name associated with constraint definition                                    
CONSTRAINT\_TYPE                                              Type of constraint definition                                                 
TABLE\_NAME                                                          Name associated with table with constraint definition                         
SEARCH\_CONDITION                                             Text of search condition for table check                                     
R\_OWNER                                                                 Owner of table used in referential constraint                                 
R\_CONSTRAINT\_NAME                                          Name of unique constraint definition for referenced table                     
DELETE\_RULE                                                          The delete rule for a referential constraint                                  
STATUS                                                                      Enforcement status of constraint -  ENABLED or DISABLED                       
DEFERRABLE                                                           Is the constraint deferrable - DEFERRABLE or NOT DEFERRABLE                   
DEFERRED                                                                 Is the constraint deferred by default -  DEFERRED or IMMEDIATE                
VALIDATED                                                       Was this constraint system validated? -  VALIDATED or NOT VALIDATED           
GENERATED                                         Was the constraint name system generated? -  GENERATED NAME or USER NAME      
BAD                                                                        Creating this constraint should give ORA-02436.  Rewrite it before 2000 AD.   
RELY                                                                                       If set, this flag will be used in optimizer                                   
LAST\_CHANGE                                                               The date when this column was last enabled or disabled                        
INDEX\_OWNER                                                                The owner of the index used by the constraint                                 
INDEX\_NAME                                                                    The index used by the constraint                                              
INVALID                                                                                            
VIEW\_RELATED        
desc user\_cons\_columns;

Name                                                                                Comments                                                                                           
--------------- -------------- -------- ------- ------------------------------------------------------------------------------------------------   
OWNER                                                                         Owner of the constraint definition                                                                 
CONSTRAINT\_NAME                                               Name associated with the constraint definition                                                     
TABLE\_NAME                                                        Name associated with table with constraint definition                                              
COLUMN\_NAME                    Name associated with column or attribute of object column specified in the constraint definition   
POSITION                                                                      Original position of column or attribute in definition

[**Oracle**](http://lib.csdn.net/base/oracle)的索引和约束详解数据库

Oracle的约束

\* 如果某个约束只作用于单独的字段，即可以在字段级定义约束，也可以在表级定义约束，但如果某个约束作用于多个字段，

必须在表级定义约束

\* 在定义约束时可以通过CONSTRAINT关键字为约束命名，如果没有指定，ORACLE将自动为约束建立默认的名称

定义primary key约束(单个字段）

create table employees (empno number(5) primary key,...)

指定约束名

create table employees (empno number(5) constraint emp\_pk primary key,...)

定义primary key约束(多个字段,在表级定义约束）

create table employees

(empno number(5),

deptno number(3) not null,

constraint emp\_pk primary key(empno,deptno)

using index tablespace indx

storage (initial 64K

next 64K

)

)

ORACLE自动会为具有PRIMARY KEY约束的字段(主码字段)建立一个唯一索引和一个NOT NULL约束,定义PRIMARY KEY约束时可以为它的索引

指定存储位置和存储参数

alter table employees add primary key (empno)

alter table employees add constraint emp\_pk primary key (empno)

alter table employees add constraint emp\_pk primary key (empno,deptno)

not null约束(只能在字段级定义NOT NULL约束,在同一个表中可以定义多个NOT NULL约束)

alter table employees modify deptno not null/null

unique约束

create table employees

( empno number(5),

ename varchar2(15),

phone varchar2(15),

email varchar2(30) unique,

deptno number(3) not null,

constraint emp\_ename\_phone\_uk unique (ename,phone)

)

alter table employees

add constraint emp\_uk unique(ename,phone)

using index tablespace indx

定义了UNIQUE约束的字段中不能包含重复值，可以为一个或多个字段定义UNIQUE约束,因此，UNIQUE即可以在字段级也可以在表级定义，

在UNIQUED约束的字段上可以包含空值.

foreign key约束

\* 定义为FOREIGN KEY约束的字段中只能包含相应的其它表中的引用码字段的值或者NULL值

\* 可以为一个或者多个字段的组合定义FOREIGN KEY约束

\* 定义了FOREIGN KEY约束的外部码字段和相应的引用码字段可以存在于同一个表中，这种情况称为"自引用"

\* 对同一个字段可以同时定义FOREIGN KEY约束和NOT NULL约束

定义了FOREIGN KEY约束的字段称为"外部码字段",被FORGIEN KEY约束引用的字段称为"引用码字段",引用码必须是主码或唯一码,包含外部码的表称为子表，

包含引用码的表称为父表.

A:

create table employees

(.....,

deptno number(3) NOT NULL,

constraint emp\_deptno\_fk foreign key (deptno)

references dept (deptno)

)

如果子表中的外部码与主表中的引用码具有相同的名称，可以写成:

B:

create table employees

(.....,

deptno number(3) NOT NULL

constraint emp\_deptno\_fk references dept

)

注意：   
上面的例子(B)中not null后面没有加逗号,因为这一句的contraint是跟在那一列deptno后面的，属于列定义，所以都无需指明列。而A例中的是表定义，需要指明那一列，所以要加逗号，不能在列后面定义，还可以写成：

create table employees   
(empno char(4),   
deptno char(2) not null constraint emp\_deptno\_fk references dept,   
ename varchar2(10)   
)   
表定义contraint的只能写在最后，再看两个例子：

create table employees   
(empno number(5),   
ename varchar2(10),   
deptno char(2) not null constraint emp\_deptno\_fk references dept,   
constraint emp\_pk primary key(empno,ename)   
)

create table employees   
( empno number(5),   
ename varchar2(15),   
phone varchar2(15),   
email varchar2(30) unique,   
deptno number(3) not null,   
constraint emp\_pk primary key(empno,ename),   
constraint emp\_phone\_uk unique (phone)   
)

添加foreign key约束(多字段／表级）   
alter table employees   
add constraint emp\_jobs\_fk foreign key (job,deptno)   
references jobs (jobid,deptno)   
on delete cascade

更改foreign key约束定义的引用行为(delete cascade/delete set null/delete no action), 默认是delete on action

引用行为(当主表中一条记录被删除时，确定如何处理字表中的外部码字段）：   
delete cascade : 删除子表中所有的相关记录   
delete set null :　将所有相关记录的外部码字段值设置为NULL   
delete no action: 不做任何操作

先删除原来的外键约束,再添加约束   
ALTER TABLE employees DROP CONSTRAINT emp\_deptno\_fk;   
ALTER TABLE employees ADD CONSTRAINT emp\_deptno\_fk FOREIGN KEY(deptno) REFERENCES dept(deptno) ON DELETE CASCADE;

check约束   
\* 在CHECK约束的表达式中必须引用到表中的一个或多个字段，并且表达式的计算结果必须是一个布尔值   
\* 可以在表级或字段级定义   
\* 对同一个字段可以定义多个CHECK约束，同时也可以定义NOT NULL约束   
　   
create table employees   
(sal number(7,2)   
constraint emp\_sal\_ck1 check (sal > 0)   
)

alter table employees   
add constraint emp\_sal\_ck2 check (sal < 20000)

删除约束

alter table dept drop unique (dname,loc) --指定约束的定义内容   
alter table dept drop constraint dept\_dname\_loc\_uk --指定约束名

删除约束时，默认将同时删除约束所对应的索引，如果要保留索引，用KEEP INDEX关键字   
alter table employees drop primary key keep index

如果要删除的约束正在被其它约束引用，通过ALTER TABLE..DROP语句中指定CASCADE关键字能够同时删除引用它的约束

利用下面的语句在删除DEPT表中的PRIMARY KEY约束时，同时将删除其它表中引用这个约束的FOREIGN KEY约束:   
alter table dept drop primary key cascade

禁用/激活约束(禁用/激活约束会引起删除和重建索引的操作)   
alter table employees disable/enable unique email   
alter table employees disable/enable constraint emp\_ename\_pk   
alter tabel employees modify constraint emp\_pk disable/enable   
alter tabel employees modify constraint emp\_ename\_phone\_uk disable/enable

如果有FOREIGN KEY约束正在引用UNIQUE或PRIMARY KEY约束，则无法禁用这些UNIQUE或PRIMARY KEY约束，

这时可以先禁用FOREIGN KEY约束，然后再禁用UNIQUE或PRIMARY KEY约束；或者可以在ALTER TABLE...DISABLE

语句中指定CASCADE关键字，这样将在禁用UNIQUE或PRIMARY KEY约束的同时禁用那些引用它们的FOREIGN KEY约束，如：

alter table employees disable primary key cascade

约束数据字典

all\_constraints/dba\_constraints/user\_constraints 约束的基本信息，包括约束的名称，类型，状态

(约束类型：C(CHECK约束),P(主码约束),R(外部码约束),U(唯一码约束))

all\_cons\_columns/dba/user 约束对应的字段信息

Oracle的索引

    索引和对应的表应该位于不同的表空间中,oracle能够并行读取位于不同硬盘上的数据，可以避免产生I/O冲突

B树索引：在B树的叶节点中存储索引字段的值与ROWID。

唯一索引和不唯一索引都只是针对B树索引而言.

Oracle最多允许包含32个字段的复合索引

索引创建策略

1.导入数据后再创建索引

2.不需要为很小的表创建索引

3.对于取值范围很小的字段（比如性别字段）应当建立位图索引

4.限制表中的索引的数目

5.为索引设置合适的PCTFREE值

6.存储索引的表空间最好单独设定

创建不唯一索引

create index emp\_ename on employees(ename)

tablespace users

storage(......)

pctfree 0;

创建唯一索引

create unique index emp\_email on employees(email)

tablespace users;

创建位图索引

create bitmap index emp\_sex on employees(sex)

tablespace users;

创建反序索引

create unique index order\_reinx on orders(order\_num,order\_date)

tablespace users

reverse;

创建函数索引(函数索引即可以是普通的B树索引，也可以是位图索引)

create index emp\_substr\_empno

on employees(substr(empno,1,2))

tablespace users;

修改索引存储参数(与表类似，INITIAL和MINEXTENTS参数在索引建立以后不能再改变)

alter index emp\_ename storage(pctincrease 50);

由于定义约束时由oracle自动建立的索引通常是不知道名称的，对这类索引的修改经常是利用alter table ..using index语句进行的,而不是alter index语句

利用下面的语句将employees表中primary key约束对应的索引的PCTFREE参数修改为5

alter table employees enable primary key using index pctfree 5;

清理索引碎片

1.合并索引(只是简单的将B树叶结点中的存储碎片合并在一起，并不会改变索引的物理组织结构）

alter index emp\_pk coalesce;

2.重建索引(不仅能够消除存储碎片,还可以改变索引的全部存储参数设置，并且可以将索引移动到其它的表空间中,重建索引

实际上就是再指定的表空间中重新建立一个新的索引,然后删除原来的索引)

alter index emp\_pk rebuild;

删除索引

drop index emp\_ename;

如果索引中包含损坏的数据块，或者包含过多的存储碎片，需要首先删除这个索引，然后再重建它.

如果索引是在创建约束时由oracle自动产生的,可以通过禁用约束或删除约束的方法来删除对应的索引.

在删除一个表时,oracle会自动删除所有与该表相关的索引.

索引数据字典

all\_indexes/dba\_indexes/user\_indexes 索引的基本信息

all\_ind\_columns/dba\_ind\_columns/user\_ind\_columns 索引对应的字段信息

1. 查询一张表里面索引

select \* from user\_indexes where table\_name=upper('tableName');

2. 查询被索引字段

select \* from user\_ind\_columns where index\_name=('indexName');

3. 给某一字段创建索引

create index index\_name on table\_name(col\_name);

1.查看所有用户

     select \* from all\_users; -------查看所有的用户

     select \* from user\_users; --------查看当前用户

2.查看用户或角色系统权限：

     select \* from user\_sys\_privs; --------查看当前用户的权限

3.查看角色所包含的权限

     select \* from role\_sys\_privs;   -------

4.查看用户对象权限

     select \* from all\_tab\_privs;   --------查看所用的用户的可操作表权限  
     select \* from user\_tab\_privs; --------查看当前用户的表可操作权限

5.查看用户或角色所拥有的角色

     select \* from user\_role\_privs;   ------查看当前用户的角色

     select \* from user\_constraints where TABLE\_NAME='？';    -----查看某一个表的约束

6.查看用户下的索引

   1.  select  \* from user\_indexes-          -----查看当前用户下的所有索引

   2.  select  \* from user\_indexes where table\_name='A';      -----查看当前用户下表A的索引  
      （drop index index\_name去掉索引）

   3. select index\_name,index\_type,status,blevel from user\_indexes where table\_name = '？';

           -----查看某一个表的所有索引

   4.  select table\_name, index\_name, column\_name, column\_position from        user\_ind\_columns where  table\_name='？';    ----查看索引的构成

 7. 建索引

       create unique clustered index 索引名on 表名(字段1)  --单索引

       Create index 索引名 on 表名(字段1,字段2)  -------复合索引