第十二章

索引

12.1 理解索引

12.1.1 什么时候使用索引

eg: 索引访问路径

```
drop index sales_fact_c2;
create index sales_fact_c2 on sales_fact (country);
set head off
```

eg:

```
select /*+ index (s sales_fact_c2) */ count(distinct(region)) from sales_fact s where country='Spain';
```

eg:

```
select count(distinct(region)) from sales_fact s where country='Spain';
```

eg: 索引访问路径2

```
alter session set statistics_level=all;
select product, year, week from sales_fact where
product = 'Xtend Memory' and year=1998 and week=1;
select /*+ full(sales_fact) */ product, year, week from sales_fact where product='Xtend Memory' and year=1998 and week=1;
```

12.1.2 列的选择

12.1.3 空值问题

eg:

```
drop table t1;
create table t1 (n1 number, n2 varchar2(100));
insert into t1 select object_id, object_name from dba_objects where rownum < 101;
commit;
create index t1_n1 on (n1);
select * from t1 where n1 is null;
create index t1_n10 on t1(n1, 0);
select * from t1 where n1 is null;
```

12.2 索引结构类型

12.2.1 B-树索引

12.2.2 位图索引

```
drop index sales_fact_part_bm1;
drop index sales_fact_part_bm2;
create bitmap index sales_fact_part_bm1 on sales_fact_part (country) local;
create bitmap index sales_fact_part_bm2 on sales_fact_part (region) local;
set termout off
select * from sales_fact_part where country='Spain' and region='Western Europe';
set termout on
```

12.2.3 索引组织表

eg:

```
drop table sales_iot;
create table sales_iot
(prod_id number not null,
cust_id number not null,
time_id date not null,
channel_id number not null,
promo_id number not null,
quantity_sold number(10,2) not null,
```

```
amount_sold number(10,2) not null,
primary key (prod_id, cust_id, time_id, channel_id, promo_id)
)
organization index;
insert into sales_iot select * from sales;
commit;
eg:
select quantity_sold, amount_sold from sales_iot where
prod_id=13 and cust_id=2 and channel_id=3 and promo_id=999;
eg:
drop index sales_iot_sec;
create index sales_iot_sec on sales_iot (channel_id, time_id, promo_id, cust_id);
eg:
select quantity_sold, amount_sold from sales_iot
where channel_id=3 and promo_id=999 and cust_id=12345 and time_id='30-Jan-00';
```

12.3 分区索引

12.3.1 局部索引

```
eg:
drop table sales_fact_part;
create table sales_fact_part
partition by range (year)
(partition p_1997 values less than (1998),
partition p_1998 values less than (1999),
partition p_1999 values less than (2000),
partition p_2000 vlaues less than (2001),
partition p_max values less than (maxvalue)
)
as select * from sales_fact;
create index sales_fact_part_n1 on sales_fact_part (product, year) local;
set lines 120 pages 100
set serveroutput off
select * from (
select * from sales_fact_part where product = 'Xtend Memory'
) where rownum < 21;
eg:
select * from (
select * from sales_fact_part where product = 'Xtend Memory' and year=1998
) where rownum < 21;
```

12.3.2 全局索引

12.3.2 全局索引

```
eg:
create index sales_fact_part_n1 on sales_fact_part (year)
global partition by range (year)
(partition p_1998 values less  than (1999),
partition p_2000 values less than (2001),
partition p_max values less than (maxvalue)
);
select * from (
select * from sales_fact_part where product = 'Xtend Memory' and year=1998
) where rownum < 21;
```

12.3.3 散列分区与范围分区

```
eg: 散列分区方案
drop sequence sfseq;
```

```
create sequence sfseq cache 200;
drop table sales_fact_part;
create table sales_fact_part
partition by hash (id)
partitions 32
as select sfseq.nextval id, f.* from sales_fact f;
create unique index sales_fact_part_n1 on sales_fact_part (id) local;
set lines 120 pages 100
set serveroutput off
select * from sales_fact_part where id = 1000;
```

eg: 散列分区分布

```
select dbms_rowid.rowid_object(rowid) obj_id, count(*) from sales_fact_part
group by dbms_rowid.rowid_object(rowid);
```

eg: 散列分区算法

```
select dbms_rowid.rowid_object(rowid) obj_id, ora_hash (id, 31, 0) part_id, count(*)
from sales_fact_part
group by dbms_rowid.rowid_object(rowid), ora_hash(id, 31, 0)
order by 1;
```

12.4 与应用特点相匹配的解决方案

12.4.1 压缩索引

eg:

```
select * from (
select product, year, week, sale from sales_fact
order by product, year, week
) where rownum < 21;
create index sales_fact_c1 sales_fact (product, year, week);
select 'Compressed index size(MB) :' || trunc(bytes/1024/1024, 2);
select 'Compressed index size (MB) :' || trunc(bytes/1024/1024, 2)
from user_segments where segment_name = 'SALES_FACT_C1';
```

eg: 最优压缩列数

```
analyze index sales_fact_c1 validate structure;
select opt_cmpr_count, opt_cmpr_pctsave from index_stats
where name = 'SALES_FACT_C1';
```

12.4.2 基于函数的索引

eg:

```
drop index sales_fact_part_fbi1;
select * from sales_fact_part where to_char(id)='1000';
create index sales_fact_part_fbi1 on sales_fact_part(to_char(id));
@analyze_table_sfp
select * from sales_fact_part where to_char(id)='1000';
```

eg: 虚拟列与基于函数的索引

```
select data_default, hidden_column, virtual_column from dba_tab_cols
where table_name='SALES_FACT_PART' and virtual_column='YES'
select index_name, column_name from dba_ind_columns
where index_name='SALES_FACT_PART_FBI1';
```

eg: analyze_table_sfp.sql

```
begin
    dbms_stats.gather_table_stats(
    ownname => user,
    tabname => 'SALES_FACT_PART',
    estimate_parcent => 30,
```

```
    cascade => true);
end;
/

eg: 虚拟列与基于函数的索引
alter table sales_fact_part add
(id_char varchar2(40) generated always as (to_char(id)) virtual)
/
create index sales_fact_part_c1 on sales_fact_part(id_char)
global partition by hash(id_char)
partitions 32;
@analyze_table_sfp
select * from sales_fact_part where to_char(id)='1000';
```

12.4.3 反转键索引

```
eg:
drop index sales_fact_part_n1;
create unique index sales_fact_part_n1 on sales_fact_part(id) global reverse;
select * from sales_fact_part where id=1000;
select * from sales_fact_part where id between 1000 and 1001;
```

12.4.4 降序索引

```
drop index sales_fact_c1;
create index sales_fact_c1 on sales_fact (product desc, year desc, week desc);
set termout off
select year, week from sales_fact s where year in (1998,1999,2000) and week < 5
and product = 'Xtend Memory'
order by product desc, year desc, week desc;
select index_name, index_type from dba_indexes where index_name='SALES_FACT_C1';
```

Note: 11R2开始，降序索引实现为基于函数的索引。

12.5 管理问题的解决方案

12.5.1 不可见索引

一个索引可以加入到数据库中并被标记为不可见，这样优化器就不会选用这个索引。
在数据库中加入索引后， 可以在会话中将 optimizer_use_invisible_indexes参数设置为 true，这样不会影响应用性能。

```
eg: 不可见索引
select * from (
select * from sales_fact where product = 'Xtend Memory' and year=1998 and week=1
) where rownum < 21;
alter index sales_fact_c1 invisible;
select * from (
select * from sales_fact where product = 'Xtend Memory' and year=1998 and week=1
) where rownum < 21;
```

Note: 不可见索引的另一个应用场景。在要删除不使用的索引前先将缩影标记为不可见，等过几周后，如果没有任何进程要用到这个索引，则可以较为安全地将其删掉。如果被标记为不可见后发现索引是需要的，则可以很快地使用一个SQL语句来将索引还原为可见状态。

```
eg: optimizer_use_invisible_indexes
alter session set optimizer_use_invisible_indexes=true;
select * from (
select * from sales_fact where product = 'Xtend Memory' and year=1998 and week=1
) where rownum < 21;
```

12.5.2 虚拟索引

虚拟索引对于查看索引的有效性很有用。虚拟索引不会分配存储空间，因此可以很快建立。
虚拟索引没有与之关联的存储空间，因此也称为无段索引。
会话可以修改隐藏参数 _use_nosegment_indexes 空值优化器是否考虑选择虚拟索引。默认为 false.
Note: 可以用虚拟索引在不影响其他应用的基础上测试虚拟索引：创建索引，在你的会话中将这个参数设置为true, 然后验证SQL语句的执行计划。

eg:

```
create indexes sales_virt on sales (cust_id, promo_id) nosegment;
alter session set "_use_nosegment_indexes"=true;
explain plan for select * from sales where cust_id = :b1 and promo_id = :b2;
select * from table(dbms_xplan.display(null,'','all'));
```

12.5.3 位图联结索引

eg: 一个典型的数据仓库（DW）查询

```
select sum(s.quantity_sold), sum(s.amount_sold)
from sales s, products p, customers c, channels ch
where s.prod_id = p.prod_id
and s.cust_id = c.cust_id
and s.channel_id = ch.channel_id
and p.prod_name='Y box'
and c.cust_first_name='Abigail'
and ch.channel_desc='Direct_sales';
```

eg: 位图联结索引

```
alter table products modify primary key validate;
alter table customers modify primary key validate;
alter table channels modify primary key validate;
create bitmap index sales_bjil on sales (p.prod_name, c.cust_first_name, ch.channel_desc)
from sales s, products p, customers c, channels ch
where s.prod_id = p.prod_id
and s.cust_id = c.cust_id
and s.channel_id = ch.channel_id
local;
select sum(s.quantity_sold), sum(s.amount_sold)
from sales s, products p, customers c, channels ch
where s.prod_id = p.prod_id
and s.cust_id = c.cust_id
and s.channel_id = ch.channel_id
and p.prod_name='Y box'
and c.cust_first_name='Abigail'
and ch.channel_desc='Direct_sales';
```