达梦数据库索引建立使用小技巧

达梦数据库支持聚集索引，复合索引，函数索引，唯一索引，位图索引等等。下面我们就来一起创建达梦数据库在各种场景中的索引。我们的测试环境是：

操作系统：中标麒麟 6 64 位。

达梦数据库：DM8.1。

```
[root@NeoKylin6-dm8 ~]# uname -ra
Linux NeoKylin6-dm8 2.6.32-220.el6.x86_64 #1 SMP Tue Apr 10 05:55:10 EDT 2012 x86_64 x86_64 x86_64 GNU/Linux
[root@NeoKylin6-dm8 ~]# uname -m
x86_64
[root@NeoKylin6-dm8 ~]#
```

```
[dmdba@NeoKylin6-dm8 bin]$ ./disql sysdba
密码:

服务器[LOCALHOST:5236]:处于普通打开状态
登录使用时间: 6.293(毫秒)
disql V8.1.0.147-Build(2019.03.27-104581)ENT
SQL>
```

1 建立索引的准则

1.1 在表中插入数据后创建索引

一般情况下，在插入或装载了数据后，为表创建索引会更加有效率。如果在装载数据之前创建了一个或多个索引，那么在插入每行时 DM 数据库都必须更改和维护每个索引，使得插入效率降低。

1.2 怎样创建正确的索引

1、如果需要经常地检索大表中的少量的行，就为查询键创建索引；
2. 为了改善多个表的连接的性能，可为连接列创建索引；
3. 主键和唯一键自动具有索引，在外键上很多情况下也创建索引；
4. 小表不需要索引。
5. 列中的值相对比较唯一 ；
6. 取值范围大，适合建立索引；
7. CLOB 和 TEXT 只能建立全文索引、BLOB 不能建立任何索引。

1.3 为性能而安排索引列

在 CREATE INDEX 语句中列的排序会影响查询的性能。通常，将**最常用的列**放在最前面。如果查询中有多个字段组合定位，则不应为每个字段单独创建索引，而应该创建一个组合索引。当两个或多个字段都是等值查询时，组合索引中各个列的前后关系是无关紧要的。 但是如果是非等值查询时，要想有效利用组合索引，则应该按等值字段在前，非等值字段在 后的原则创建组合索引，查询时只能利用一个非等值的字段。

1.4 限制每个表的索引的数量

一个表可以有任意数量的索引。但是，索引越多，修改表数据的开销就越大。当插入或删除行时，表上的所有索引也要被更改；更改一个列时，包含该列的所有索引也要被更改。因此，在从表中检索数据的速度和更新表的速度之间有一个折衷。例如，如果一个表主要仅用于读，则索引多就有好处；如果一个表经常被更新，则索引不宜多建。

2 创建索引

2.1 创建聚集索引

DM 数据库中表（列存储表和堆表除外）都是使用 B+树索引结构管理的，每一个普通表都有且仅有一个聚集索引，数据都通过聚集索引键排序，根据聚集索引键可以快速查询任何记录。

当建表语句未指定聚集索引键时，DM 数据库的默认聚集索引键是 ROWID。若指定索引键，表中数据都会根据指定索引键排序。 建表后，DM 数据库也可以用创建新聚集索引的方式来重建表数据，并按新的聚集索引排序。 例如，可以对 employee 表以 employee_name 列新建聚集索引。

先建立索引表空间 ind_tbs:

```
Sql>create tablespace ind_tbs datafile
    '/dm8/data/DAMENG/ind_tbs.dbf' size 32 autoextend on next 10
    maxsize 2000;
```

```
SQL> create tablespace ind_tbs datafile '/dm8/data/DAMENG/ind_tbs.dbf' size 32 autoextend on next 10 maxsize 2000
;
操作已执行
已用时间: 95.026(毫秒). 执行号:4.
SQL>
```

```
Sql>create index ind_emp on dmhr.employee(employee_name)
    tablespace ind_tbs;
```

```
SQL> create index ind_emp on dmhr.employee(employee_name) tablespace ind_tbs;
操作已执行
已用时间: 148.062(毫秒). 执行号:6.
SQL>
```

2.2 复合索引

```
Sql>create index ind_emp_dep on
   dmhr.employee(employee_id,department_id) tablespace ind_tbs;
```

```
SQL> create index ind_emp_dep on dmhr.employee(employee_id,department_id) tablespace ind_tbs;
操作已执行
已用时间: 4.792(毫秒). 执行号:7.
SQL>
```

2.3 函数索引

基于函数的索引促进了限定函数或表达式的返回值的查询，该函数或表达式的值被预先计算出来并存储在索引中。

Sql> create index ind_emp1 on dmhr.employee(abs(salary)) tablespace ind_tbs;

```
SQL> create index ind_emp1 on dmhr.employee(abs(salary)) tablespace ind_tbs;
操作已执行
已用时间: 6.869(毫秒). 执行号:9.
SQL>
```

2.4 位图索引

位图索引主要针对含有大量相同值的列而创建。位图索引被广泛引用到数据仓库中。

Sql>create bitmap index ind_t on t1(id);

```
SQL> select * from t1;

行号      ID          NAME
---------- ----------- -------
1           1           sysdba

已用时间: 41.838(毫秒). 执行号:10.
SQL> create bitmap index ind_t on t1(id);
操作已执行
已用时间: 70.205(毫秒). 执行号:11.
SQL>
```

2.5 唯一索引

索引可以是唯一的或非唯一的。唯一索引可以保证表上不会有两行数据在键列上具有相同的值。

SQL> create unique index ind_t2 on t2(id);

```
SQL> create table t2(id int);
操作已执行
已用时间: 2.627(毫秒). 执行号:12.
SQL> create unique index ind_t2 on t2(id);
操作已执行
已用时间: 4.979(毫秒). 执行号:13.
SQL>
```

3 索引重建

当一个表经过大量的增删改操作后，表的数据在物理文件中可能存在大量碎片，从而影响访问速度。另外，当删除表的大量数据后，若不再对表执行插入操作，索引所处的段可能占用了大量并不使用的簇，从而浪费了存储空间。可以使用重建索引来对索引的数据进行重组，使数据更加紧凑，并释放不需要的空间，从而提高访问效率和空间效率。

DM 数据库提供的重建索引的系统函数为:

SP_REBUILD_INDEX(SCHEMA_NAME varchar(256), INDEX_ID int);

SCHEAM_NAME 为索引所在的模式名。

INDEX_ID 为索引 ID。

使用说明:

1. 水平分区子表，临时表和系统表上建的索引不支持重建

2. 虚索引和聚集索引不支持重建

Sql>select name,id,subtype$ from sysobjects where subtype$='INDEX' and name='IND_EMP_DEP';

```
SQL> select name,id,subtype$ from sysobjects where subtype$='INDEX' and name='IND_EMP_DEP';

行号     NAME        ID          SUBTYPE$
---------- ----------- ----------- --------
1          IND_EMP_DEP 33555517    INDEX

已用时间: 4.202(毫秒). 执行号:8.
SQL>
```

Sql>sp_rebuild_index('DMHR', '335555517');

```
SQL> sp_rebuild_index('DMHR','33555517');
DMSQL 过程已成功完成
已用时间: 188.854(毫秒). 执行号:9.
SQL>
```

SQL> alter index DMHR.IND_EMP_DEP rebuild;

```
SQL> alter index DMHR.IND_EMP_DEP rebuild;
操作已执行
已用时间: 30.889(毫秒). 执行号:10.
SQL>
```

在线重建索引

SQL> alter index DMHR.IND_EMP_DEP rebuild online;

```
SQL> alter index DMHR.IND_EMP_DEP rebuild online;
操作已执行
已用时间: 126.654(毫秒). 执行号:11.
SQL>
```

4 删除索引

用户可能出于以下某项原因需要删除一个索引:

1. 不再需要该索引;

2. 该索引没有为针对其相关的表所发布的查询提供所期望的性能改善。例如，表可能很小，或者尽管表中有许多行但只有很少的索引项;

3. 应用没有用该索引来查询数据。要想删除索引，则该索引必须包含在用户的模式中或用户必须具有 DROP ANY INDEX 数据库权限。索引删除之后，该索引的段的所有簇都返回给包含它的表空间，并可用于表空间中的其他对象。

Sql>DROP INDEX dmhr.IND_EMP_DEP;

```
SQL> DROP INDEX dmhr.IND_EMP_DEP;
操作已执行
已用时间: 104.536(毫秒). 执行号:12.
SQL> select name,id,subtype$ from sysobjects where subtype$='INDEX' and name='IND_EMP_DEP';
未选定行

已用时间: 1.676(毫秒). 执行号:13.
SQL>
```

5 查看索引信息

创建索引后，可以通过 INDEXDEF 系统函数查看索引的定义。

INDEXDEF(INDEX_ID int, PREFLAG int);

INDEX_ID 为索引 ID。

PREFLAG 表示返回信息中是否增加模式名前缀。

例如，需要查看索引 IND_EMP 的定义，那么使用以下语句查看索引定义。

Sql>select name,id,subtype$ from sysobjects where subtype$='INDEX' and name='IND_EMP';

```
SQL> select name,id,subtype$ from sysobjects where subtype$='INDEX' and name='IND_EMP';

行号     NAME    ID          SUBTYPE$
---------- ------- ----------- --------
1          IND_EMP 33555516    INDEX

已用时间: 2.641(毫秒). 执行号:14.
SQL>
```

Sql>SELECT INDEXDEF(33555516, 0)；或 SELECT INDEXDEF(33555516, 1)；

```
SQL> SELECT INDEXDEF(33555516, 0);

行号     INDEXDEF(33555516,0)
---------- --------------------------------------------------------------------------------
1          CREATE  INDEX "IND_EMP" ON "EMPLOYEE"("EMPLOYEE_NAME" ASC) STORAGE(ON "IND_TBS",
 CLUSTERBTR) ;

已用时间: 2.379(毫秒). 执行号:15.
SQL> SELECT INDEXDEF(33555516, 1);

行号     INDEXDEF(33555516,1)
---------- --------------------------------------------------------------------------------
1          CREATE  INDEX "IND_EMP" ON "DMHR"."EMPLOYEE"("EMPLOYEE_NAME" ASC) STORAGE(ON "IN
D_TBS", CLUSTERBTR) ;

已用时间: 1.469(毫秒). 执行号:16.
SQL>
```

好了，今天的索引使用小技巧就分享到这里了。大家在长假期间可以去尝试一下！