TPC-H 数据生成与导入小结(续)

黄一夫

qgen 使用

qgen 是产生 22 个查询语句的生成器，运行的同时需要 dbgen\ queries 下的 22 个 sql 模板和 dbgen\dists.dss 字典文件。

首先按与 dbgen 相同的方法组建工程 qgen，将得到的 dbgen\Debug\qgen.exe，dbgen\queries 下的 22 个 sql 查询模板和 dbgen\dists.dss 字典文件移动到新建的文件夹 to_query 中。

然后打开 cmd，cd 到 to_query 目录，执行指令如下

qgen –d 1 > d1.sql

指令解释：-d 代表生成 default 格式的 sql 语句，1 代表使用 dbgen\queries 下的第 1 个模板，> d1.sql 代表利用命名管道将控制台的输出重定向到当前目录的 d1.sql 中。

最后，生成的 sql 语句还有一定的问题，需要做一些修改。结尾有 set rowcount -1 go 语句的将之去掉，将 substring 函数修改为 substr 函数，将表别名前面的 as 关键字去掉,将子查询构成的别名后的列名移动到子查询的 select 子句。

按照该格式一共需要生成 22 个 default 的 sql 文件，然后在 sqlplus 中 set timing on 开启计时器，@刚才生成的 sql 文件，进行测试，以下是我的测试结果

```
PC: Lenovo Y460
CPU:  Intel(R) Core(TM) i3 CPU M 350 @ 2.27GHz
RAM:  3.00GB(2.43G)
OS: Windows 7 32bit
DB: Oracle 11g R2
==========================================================
d1  12.02 12.17 12.49
d2  10.47 10.22 15.29
d3  26.30 26.08 27.52
d4  14.97 14.82 14.70
d5  05:30.06  02:46.95  02:43.48
d6  11.98 12.05 12.04
d7  15.87 15.35 15.08
d8  15.42 15.44 15.56
d9  17.86 17.19 17.26
d10 03:28.29  03:30.08  03:29.95
d11 02.93 02.66 02.87
d12 14.88 15.06 14.71
d13 03.29 03.21 03.21
d14 12.55 12.52 12.40
d15 (00.40,13.17,02.24) (00.05,12.15,00.03) (00.01,12.14,00.00)
d16 14.70 13.47 13.56
d17 12.54 12.74 12.51
d18 17.32 14.42 13.65
d19 12.47 12.43 12.91
d20 15.70 15.26 14.99
d21 40.72 33.58 31.76
d22 03.57 03.47 03.43
```

如果是需要强制并行查询，则在 sqlplus 中设置

alter session force parallel query

无论原始表是否开启了并行，设定了什么并行度，查询优化器都采用并行查询。

---

sqlldr 补充

首先，sqlldr 的加载有 2 种模式，常规路径和直接路径，前者要将数据转化为 INSERT 语句，通过 SGA 区加载，后者将数据在内存中组成数据库的数据块格式，直接写入数据文件，避免了语句解释和记录日志的开销，因此在类似数据仓库的大量数据导入时，一般采用直接路径加载。

然后，加载方式除 INSERT 外,还可以取值 APPEND、REPLACE 和 TRUNCATE。要执行

INSERT， 必须保证表为空，否则 sqlldr 报错，不能继续执行。如果想向表中增加记录，可以指定加载选项为 APPEND;为了替换表中已有的数据，可以使用 REPLACE 或 TRUNCATE。REPLACE 使用 DELETE 语句删除全部记录;因此，如果要加载的表中已经包含许多记录，这个操作执行得很慢。TRUNCATE 使用 TRUNCATE SQL 命令，执行更快，因为它不必物理地删除每一行。但是 TRUNCATE 不能回退。要小心地设置这个选项，有时候其他参数也会影响这个选项。

最后，如果同一个表有多个外部数据文件，那么通过设置 Parallel 参数=TRUE，采用并行加载，可以提高加载速度，例如：

sqlldr scott/tiger control=lineitem.ctl direct=true parallel=true

注意 Parallel 参数只是表示允许多个 sqlldr 进程同时加载，而不是对当前语句采用并行方式，也就是说，一个 sqlldr 命令只能串行加载。

Windows 操作系统不支持&语法的后台进程，但是可以用打开多个 cmd 窗口，分别执行多个不同的 sqlldr 语句的方式，也能达到相同的效果。需要指出的是，服务器的 I/O 能力对加载有巨大的影响，如果读写的 I/O 带宽已经用满，那么实际上就是 sqlldr 在等待 I/O 完成，那么此刻再启动多个 sqlldr 也不会提高加载性能。

数据查询

该部分实验数据来自于 Internet

为了比较不同条件下的查询结果，我们进行了 4 种组合的查询。分别是：单进程不压缩，并行不压缩，单进程压缩，并行压缩，每种测试做 2 遍，取较快的一遍的结果。

```
--用来压缩表的语句，并行参数可加快速度，但并不改变被move的表的并行度
alter table CUSTOMER move compress parallel 32;
alter table LINEITEM move compress parallel 32;
alter table NATION move compress parallel 32;
alter table ORDERS move compress parallel 32;
alter table PART move compress parallel 32;
alter table PARTSUPP move compress parallel 32;
alter table REGION move compress parallel 32;
alter table SUPPLIER move compress parallel 32;
```

```
--压缩前字节数
SQL> set numw 20
SQL> select segment_name,sum(bytes) from user_segments where segment_name
not like '%EXT%' group by segment_name order by 1;

SEGMENT_NAME              SUM(BYTES)
------------------ --------------------
CUSTOMER                  281804800
LINEITEM                 7730102272
NATION                     65536
ORDERS                   1874067456
PART                     278986752
PARTSUPP                  1367867392
REGION                     65536
SUPPLIER                  16646144

--压缩后
SEGMENT_NAME              SUM(BYTES)
------------------- --------------------
CUSTOMER                  248643584
LINEITEM                 5389484032
NATION                     65536
ORDERS                   1566310400
PART                     207290368
PARTSUPP                  1251344384
REGION                     65536
SUPPLIER                  17301504
```

测试结果

编号	单进程	并行	倍数	压缩单进程	压缩并行	倍数
01	66.24	7.91	8.4	68.68	7.14	9.6
02	3.83	0.92	4.2	3.61	0.57	6.3
03	28.44	5.97	4.8	29.69	3.47	8.6
04	22.18	2.53	8.8	27.08	2.79	9.7
05	30.76	4.43	6.9	33.4	7.87	4.2
06	16.5	1.6	10.3	18.01	1.5	12.0
07	25.1	3.97	6.3	58.98	3.47	17.0
08	20.83	2.91	7.2	23.41	2.71	8.6
09	60.02	8.54	7.0	55.11	9.64	5.7
10	26.06	5.49	4.7	29.21	4.43	6.6
11	3.7	2.55	1.5	3.37	0.45	7.5
12	22.86	3.43	6.7	22.58	2.89	7.8
13	70.98	8	8.9	66.57	7.1	9.4
14	19.08	1.92	9.9	16.13	1.79	9.0
15	17.28	4.29	4.0	16.02	1.65	9.7
16	6.06	2.17	2.8	6.38	1.23	5.2
17	15.4	1.95	7.9	18.2	1.81	10.1
18	58	10.17	5.7	130.43	7.73	16.9
19	18.93	2.11	9.0	37.77	1.83	20.6
20	20.67	3.21	6.4	23.35	2.58	9.1
21	47.88	19	2.5	59.65	9.35	6.4
22	6.31	1.64	3.8	6.04	1.07	5.6
合计	607.11	104.71	5.8	753.67	83.07	9.1

可见无论是否压缩，并行查询比单进程都有几倍或十几倍的提高，具体提高的倍数和查询的类型和机器的 CPU 个数有关。在压缩的情况下，单进程的性能比不压缩更差，所以光看提高的倍数是不够的，还要看查询的实际时间比。

EMP/IMP 使用

导入(IMP)/导出(EXP)是 ORACLE 幸存的最古老的两个操作系统命令行工具，Exp/Imp 是一个好的转储工具，特别是在小型数据库的转储，表空间的迁移，表的抽取，检测逻辑和物理冲突等中有不小的功劳。它作为小型数据库的物理备份后的一个逻辑辅助备份，也是不错的手段。对于越来越大的数据库，特别是 TB 级数据库和越来越多数据仓库的出现，EXP/IMP

越来越力不从心了，这个时候，数据库的备份都转向了 RMAN 和第三方工具。

简单的来说，exp 和 imp 可以将 oracle 中的数据库，表等内容导出成文件，既可以称作备份，又可以做多台计算机间数据库的移植，例如：

exp scott/tiger@orcl file=d:\expfile.dmp tables=(PART)

imp scott/tiger@orcl file=d:\expfile.dmp tables=(PART)
