seanstuber

Tips and Examples for working with Oracle databases

Setting up the Star Schema Benchmark (SSB) in Oracle

JUNE 14, 2017 LEAVE A COMMENT (HTTPS://SEANSTUBER.WORDPRESS.COM/2017/06/14/SETTING-UP-THE-STAR-SCHEMA-BENCHMARK-SSB-IN-ORACLE/#RESPOND)

i
Rate This

In my previous two posts I showed how to setup a schema for the TPC-H tables and test data. A related test system called the Star Schema Benchmark (SSB) from Pat O'Neil, Betty O'Neil, and Xuedong Chen at the University of Massachusetts at Boston alters the TPC-H structures to create a warehousing data model.

The SSB is documented here. (http://www.cs.umb.edu/~poneil/StarSchemaB.PDF) As with the TPC-H setup, a modified version of the dbgen utility creates the data can also be found on the UMass site here. (http://www.cs.umb.edu/~poneil/dbgen.zip)

Unzip the dbgen file somewhere on your linux system. As of the time of this writing June 5, 2009 Revision 3 is the current version of the SSB. In the future the steps may change slightly but should be fairly similar to what is described below.
When you unzip, a directory dbgen will be created. Change to that directory, make a copy of the makefile template and edit it. The SSB version of dbgen doesn't support Oracle but that's not necessary for the data setup, only the query generation requires a database. So you can pick any of the supported databases for the purposes of data creation.

```
$ cd dbgen
$ cp makefile.suite makefile
$ vi makefile
```

Within the makefile, change the following lines

```
CC = gcc
DATABASE= SQLSERVER
MACHINE = LINUX
WORKLOAD = SSBM
```

Then make will create the dbgen utility. You'll get several warnings but since we're not distributing the resulting binary and have controlled usage they are safe to ignore.

Oracle offers a training lab for the 12c In-Memory option utilizing the SSB schema. Their training guide states the test system is based on a 50GB scale; but the query results illustrated in the guide show only a 4GB data set, so I'll demonstrate the same here. We'll generate a small, approximately 4GB, set of test data and place it in a directory we'll use later for the upload into the Oracle tables. Unlike the TPC-H dbgen you must generate each file type individually.

```
$ ./dbgen -s 4 -T c
$ ./dbgen -s 4 -T p
$ ./dbgen -s 4 -T s
$ ./dbgen -s 4 -T d
$ ./dbgen -s 4 -T l
$ mv *.tbl /home/oracle/ssb
```

Within the database, set up the SSB schema. All tables will be created according to the layouts described in section 2 of the SSB specification. For the purposes of data import, a set of external tables will also be created. These are not part of SSB itself and may be left in place or dropped after data load is complete. The SSB specification describes primary keys and foreign keys but no check constraints. The TPC-H schema allows NOT NULL declarations so I've included them here as well. If you prefer, simply remove the "NOT NULL" in the DDL below to allow nulls. The dbgen utility will populate every column though. The descriptions of a couple tables have some errors which I've made note of in the comments below. One is a partially documented; but seemingly unused

column which I have removed. The other is on the date table which defines a day-of-week column of 8 characters but dbgen creates some days of 9 letters in length ("Wednesday".)

One final change, the "DATE" table has been renamed "DATE_DIM" since DATE is an Oracle keyword. This change also makes the schema compatible with the Oracle In-Memory lab.

```
CREATE USER ssb IDENTIFIED BY ssb;

GRANT CREATE SESSION,
      CREATE TABLE,
      CREATE ANY DIRECTORY,
      UNLIMITED TABLESPACE
    TO ssb;

CREATE OR REPLACE DIRECTORY ssb_dir AS '/home/oracle/ssb';

GRANT READ, WRITE ON DIRECTORY ssb_dir TO ssb;

CREATE TABLE ssb.ext_lineorder
(
    lo_orderkey        INTEGER,
    lo_linenumber      NUMBER(1, 0),
    lo_custkey         INTEGER,
    lo_partkey         INTEGER,
    lo_suppkey         INTEGER,
    lo_orderdate       INTEGER,
    lo_orderpriority   CHAR(15),
    lo_shippriority    CHAR(1),
    lo_quantity        NUMBER(2, 0),
    lo_extendedprice   NUMBER,
    lo_ordtotalprice   NUMBER,
    lo_discount        NUMBER(2, 0),
    lo_revenue         NUMBER,
    lo_supplycost      NUMBER,
    --lo_ordsupplycost   NUMBER, -- this is mentioned in 2.2 Notes(c)
    lo_tax             NUMBER(1, 0),
    lo_commitdate      INTEGER,
    lo_shipmode        CHAR(10)
)
ORGANIZATION EXTERNAL
    (TYPE oracle_loader
          DEFAULT DIRECTORY ssb_dir
              ACCESS PARAMETERS (
                  FIELDS
                      TERMINATED BY '|'
                  MISSING FIELD VALUES ARE NULL
              )
          LOCATION('lineorder.tbl*'))
          PARALLEL 4;

CREATE TABLE ssb.lineorder
(
    lo_orderkey        INTEGER NOT NULL,
    lo_linenumber      NUMBER(1, 0) NOT NULL,
    lo_custkey         INTEGER NOT NULL,
    lo_partkey         INTEGER NOT NULL,
    lo_suppkey         INTEGER NOT NULL,
    lo_orderdate       NUMBER(8,0) NOT NULL,
    lo_orderpriority   CHAR(15) NOT NULL,
    lo_shippriority    CHAR(1) NOT NULL,
    lo_quantity        NUMBER(2, 0) NOT NULL,
    lo_extendedprice   NUMBER NOT NULL,
    lo_ordtotalprice   NUMBER NOT NULL,
    lo_discount        NUMBER(2, 0) NOT NULL,
    lo_revenue         NUMBER NOT NULL,
    lo_supplycost      NUMBER NOT NULL,
    --lo_ordsupplycost   NUMBER not null, -- this is mentioned in 2.2
    lo_tax             NUMBER(1, 0) NOT NULL,
    lo_commitdate      NUMBER(8,0) NOT NULL,
    lo_shipmode        CHAR(10) NOT NULL
);

CREATE TABLE ssb.ext_part
(
    p_partkey     INTEGER,
```

```
    p_name        VARCHAR2(22),
    p_mfgr        CHAR(6),
    p_category    CHAR(7),
    p_brand1      CHAR(9),
    p_color       VARCHAR2(11),
    p_type        VARCHAR2(25),
    p_size        NUMBER(2, 0),
    p_container   CHAR(10)
)
ORGANIZATION EXTERNAL
    (TYPE oracle_loader
          DEFAULT DIRECTORY ssb_dir
              ACCESS PARAMETERS (
                  FIELDS
                      TERMINATED BY '|'
                  MISSING FIELD VALUES ARE NULL
              )
          LOCATION('part.tbl'));

CREATE TABLE ssb.part
(
    p_partkey     INTEGER NOT NULL,
    p_name        VARCHAR2(22) NOT NULL,
    p_mfgr        CHAR(6) NOT NULL,
    p_category    CHAR(7) NOT NULL,
    p_brand1      CHAR(9) NOT NULL,
    p_color       VARCHAR2(11) NOT NULL,
    p_type        VARCHAR2(25) NOT NULL,
    p_size        NUMBER(2, 0) NOT NULL,
    p_container   CHAR(10) NOT NULL
);

CREATE TABLE ssb.ext_supplier
(
    s_suppkey   INTEGER,
    s_name      CHAR(25),
    s_address   VARCHAR2(25),
    s_city      CHAR(10),
    s_nation    CHAR(15),
    s_region    CHAR(12),
    s_phone     CHAR(15)
)
ORGANIZATION EXTERNAL
    (TYPE oracle_loader
          DEFAULT DIRECTORY ssb_dir
              ACCESS PARAMETERS (
                  FIELDS
                      TERMINATED BY '|'
                  MISSING FIELD VALUES ARE NULL
              )
          LOCATION('supplier.tbl'));

CREATE TABLE ssb.supplier
(
    s_suppkey   INTEGER NOT NULL,
    s_name      CHAR(25) NOT NULL,
    s_address   VARCHAR2(25) NOT NULL,
    s_city      CHAR(10) NOT NULL,
    s_nation    CHAR(15) NOT NULL,
    s_region    CHAR(12) NOT NULL,
    s_phone     CHAR(15) NOT NULL
);

CREATE TABLE ssb.ext_customer
(
    c_custkey      INTEGER,
    c_name         VARCHAR2(25),
    c_address      VARCHAR2(25),
    c_city         CHAR(10),
    c_nation       CHAR(15),
```

```
    c_region       CHAR(12),
    c_phone        CHAR(15),
    c_mktsegment   CHAR(10)
)
ORGANIZATION EXTERNAL
    (TYPE oracle_loader
          DEFAULT DIRECTORY ssb_dir
              ACCESS PARAMETERS (
                  FIELDS
                      TERMINATED BY '|'
                  MISSING FIELD VALUES ARE NULL
              )
          LOCATION('customer.tbl'));

CREATE TABLE ssb.customer
(
    c_custkey      INTEGER NOT NULL,
    c_name         VARCHAR2(25) NOT NULL,
    c_address      VARCHAR2(25) NOT NULL,
    c_city         CHAR(10) NOT NULL,
    c_nation       CHAR(15) NOT NULL,
    c_region       CHAR(12) NOT NULL,
    c_phone        CHAR(15) NOT NULL,
    c_mktsegment   CHAR(10) NOT NULL
);

CREATE TABLE ssb.ext_date_dim
(
    d_datekey            NUMBER(8,0),
    d_date               CHAR(18),
    d_dayofweek          CHAR(9),    -- defined in Section 2.6 as Size
    d_month              CHAR(9),
    d_year               NUMBER(4, 0),
    d_yearmonthnum       NUMBER(6, 0),
    d_yearmonth          CHAR(7),
    d_daynuminweek       NUMBER(1, 0),
    d_daynuminmonth      NUMBER(2, 0),
    d_daynuminyear       NUMBER(3, 0),
    d_monthnuminyear     NUMBER(2, 0),
    d_weeknuminyear      NUMBER(2, 0),
    d_sellingseason      CHAR(12),
    d_lastdayinweekfl    NUMBER(1, 0),
    d_lastdayinmonthfl   NUMBER(1, 0),
    d_holidayfl          NUMBER(1, 0),
    d_weekdayfl          NUMBER(1, 0)
)
ORGANIZATION EXTERNAL
    (TYPE oracle_loader
          DEFAULT DIRECTORY ssb_dir
              ACCESS PARAMETERS (
                  FIELDS
                      TERMINATED BY '|'
                  MISSING FIELD VALUES ARE NULL
              )
          LOCATION('date.tbl'));

CREATE TABLE ssb.date_dim
(
    d_datekey            NUMBER(8,0) NOT NULL,
    d_date               CHAR(18) NOT NULL,
    d_dayofweek          CHAR(9) NOT NULL,    -- defined in Section 2.
    d_month              CHAR(9) NOT NULL,
    d_year               NUMBER(4, 0) NOT NULL,
    d_yearmonthnum       NUMBER(6, 0) NOT NULL,
    d_yearmonth          CHAR(7) NOT NULL,
    d_daynuminweek       NUMBER(1, 0) NOT NULL,
    d_daynuminmonth      NUMBER(2, 0) NOT NULL,
    d_daynuminyear       NUMBER(3, 0) NOT NULL,
    d_monthnuminyear     NUMBER(2, 0) NOT NULL,
    d_weeknuminyear      NUMBER(2, 0) NOT NULL,
```

```
    d_sellingseason      CHAR(12) NOT NULL,
    d_lastdayinweekfl    NUMBER(1, 0) NOT NULL,
    d_lastdayinmonthfl   NUMBER(1, 0) NOT NULL,
    d_holidayfl          NUMBER(1, 0) NOT NULL,
    d_weekdayfl          NUMBER(1, 0) NOT NULL
);
```

Now load the data. As you scale up into larger volumes, these steps are still valid; but you may want to split the loads into separate steps and alter the LINEORDER external table to read multiple files in parallel and use parallel dml on insert in order to speed up the process. The truncate lines aren't necessary for the first time data load; but are included for future reloads of the dbgen data with other scaling.

```
TRUNCATE TABLE ssb.lineorder;
TRUNCATE TABLE ssb.part;
TRUNCATE TABLE ssb.supplier;
TRUNCATE TABLE ssb.customer;
TRUNCATE TABLE ssb.date_dim;

ALTER TABLE ssb.lineorder PARALLEL 4;
ALTER SESSION ENABLE PARALLEL DML;

INSERT /*+ APPEND */ INTO  ssb.part      SELECT * FROM ssb.ext_part;
commit;
INSERT /*+ APPEND */ INTO  ssb.supplier  SELECT * FROM ssb.ext_supplie
commit;
INSERT /*+ APPEND */ INTO  ssb.customer  SELECT * FROM ssb.ext_custome
commit;
INSERT /*+ APPEND */ INTO  ssb.date_dim  SELECT * FROM ssb.ext_date_di
commit;
INSERT /*+ APPEND */ INTO  ssb.lineorder SELECT * FROM ssb.ext_lineorc
commit;
```

And finally, add the constraints and indexes.

```
ALTER TABLE ssb.lineorder
    ADD CONSTRAINT pk_lineorder PRIMARY KEY(lo_orderkey, lo_linenumber

ALTER TABLE ssb.part
    ADD CONSTRAINT pk_part PRIMARY KEY(p_partkey);

ALTER TABLE ssb.supplier
    ADD CONSTRAINT pk_supplier PRIMARY KEY(s_suppkey);

ALTER TABLE ssb.customer
    ADD CONSTRAINT pk_customer PRIMARY KEY(c_custkey);

ALTER TABLE ssb.date_dim
    ADD CONSTRAINT pk_date_dim PRIMARY KEY(d_datekey);

---

ALTER TABLE ssb.lineorder
    ADD CONSTRAINT fk_lineitem_customer FOREIGN KEY(lo_custkey) REFERE

ALTER TABLE ssb.lineorder
    ADD CONSTRAINT fk_lineitem_part FOREIGN KEY(lo_partkey) REFERENCES

ALTER TABLE ssb.lineorder
    ADD CONSTRAINT fk_lineitem_supplier FOREIGN KEY(lo_suppkey) REFERE

ALTER TABLE ssb.lineorder
    ADD CONSTRAINT fk_lineitem_orderdate FOREIGN KEY(lo_orderdate) REF

ALTER TABLE ssb.lineorder
    ADD CONSTRAINT fk_lineitem_commitdate FOREIGN KEY(lo_commitdate) F
```

And that's it, you should now have a complete SSB scale-4 data set to complete either the SSB suite of test queries, oracle labs, or run your own tests.

If you want to generate larger data sets you can with similar syntax to that seen with TPC-H. The DDL and inserts above are already

defined for either parallel or single-file loads of the lineorder table. The other tables are relatively small in comparison. They can still be split if desired but you probably won't need to.

```
$ ./dbgen -s 10 -T c
$ ./dbgen -s 10 -T p
$ ./dbgen -s 10 -T s
$ ./dbgen -s 10 -T d
$ ./dbgen -s 10 -T l -C 4 -S 1
$ ./dbgen -s 10 -T l -C 4 -S 2
$ ./dbgen -s 10 -T l -C 4 -S 3
$ ./dbgen -s 10 -T l -C 4 -S 4
```

Enjoy!

FILED UNDER ORACLE

About Sean D. Stuber
Developer, DBA, Oracle ACE