CSCI317 Database Performance Tuning

Performance Tuning with Advanced SQL

Dr Janusz R. Getta

School of Computing and Information Technology -
University of Wollongong

Performance Tuning with Advanced SQL

Outline

Too simplistic SQL

Consider the following data manipulation problem

Split a table `LINEITEM` into two tables such both tables include primary key and one table includes all dates and the other table includes all other attributes (vertical partitioning of a relational table)

CTAS statement

```
CREATE TABLE DATES AS
 (SELECT L_ORDERKEY, L_LINENUMBER, L_SHIPDATE,
         L_COMMITDATE, L_RECEIPTDATE
  FROM LINEITEM );
```

CTAS statement

```
CREATE TABLE OTHERS AS
 (SELECT L_ORDERKEY, L_LINENUMBER, L_QUANTITY,
         L_EXTENDEDPRICE, L_DISCOUNT, L_TAX, L_RETURNFLAG,
         L_LINESTATUS, L_SHIPDATE, L_COMMITDATE,
         L_RECEIPTDATE, L_SHIPINSTRUCT, L_SHIPMODE,
         L_COMMENT
  FROM LINEITEM );
```

Too simplistic SQL

Consider the following data manipulation problem

Create a table `ALL_SUPPLIERS` with the same contents as `SUPPLIER` table,

CTAS statement

```
CREATE TABLE ALL_SUPPLIERS AS
 (SELECT * FROM SUPPLIER);
```

Assume that some rows have been added or updated (an attribute `S_ADDRESS` has been changed) in `SUPPLIER` table

Replicate insertions and updates in `ALL_SUPPLIERS` table

INSERT statement

```
INSERT INTO ALL_SUPPLIERS
 (SELECT * FROM SUPPLIER MINUS
  SELECT * FROM ALL_SUPPLIERS);
```

UPDATE statement

```
UPDATE ALL_SUPPLIERS
SET S_ADDRESS = (SELECT S_ADDRESS FROM SUPPLIER
                 WHERE ALL_SUPPLIERS.S_SUPPKEY = SUPPLIER.S_SUPPKEY);
```

Performance Tuning with Advanced SQL

Outline

Conditional multitable `INSERT` statement

Consider the following data manipulation problem

Split a table `LINEITEM` into two tables such both tables include primary key and one table includes all dates and the other table includes all other attributes

Conditional multitable INSERT statement

```
INSERT
ALL
INTO DATES
 VALUES(L_ORDERKEY, L_LINENUMBER, L_SHIPDATE,
        L_COMMITDATE, L_RECEIPTDATE )
INTO OTHERS
 VALUES(L_ORDERKEY, L_LINENUMBER, L_QUANTITY,
        L_EXTENDEDPRICE, L_DISCOUNT, L_TAX, L_RETURNFLAG,
        L_LINESTATUS, L_SHIPDATE, L_COMMITDATE,
        L_RECEIPTDATE, L_SHIPINSTRUCT, L_SHIPMODE,
        L_COMMENT)
SELECT *
FROM LINEITEM;
```

Conditional multitable `INSERT` statement

Split a table `LINEITEM` into two tables such both tables include primary key and one table includes all dates from the rows where a value of `L_QUANTITY > 40` and the other table includes all other attributes from the rows where `L_QUANTITY < 50`

Conditional Multitable INSERT statement

```
INSERT
ALL
WHEN L_QUANTITY > 40 THEN
INTO DATES
 VALUES(L_ORDERKEY, L_LINENUMBER, L_SHIPDATE,
        L_COMMITDATE, L_RECEIPTDATE )
WHEN L_QUANTITY < 50 THEN
INTO OTHERS
 VALUES(L_ORDERKEY, L_LINENUMBER, L_QUANTITY,
        L_EXTENDEDPRICE, L_DISCOUNT, L_TAX, L_RETURNFLAG,
        L_LINESTATUS, L_SHIPDATE, L_COMMITDATE,
        L_RECEIPTDATE, L_SHIPINSTRUCT, L_SHIPMODE,
        L_COMMENT)
SELECT *
FROM LINEITEM;
```

Conditional multitable `INSERT` statement

Split a table `LINEITEM` into two tables such both tables include primary key and one table includes all dates from the rows where a value of `L_QUANTITY > 40` and the other table includes all other attributes from the rows where `L_QUANTITY < 50`, If a row can be inserted into both tables then insert it only to a table with dates

Conditional multitable INSERT statement

```
INSERT
FIRST
WHEN L_QUANTITY > 40 THEN
INTO DATES
VALUES(L_ORDERKEY, L_LINENUMBER, L_SHIPDATE,
       L_COMMITDATE, L_RECEIPTDATE )
WHEN L_QUANTITY < 50 THEN
INTO OTHERS
 VALUES(L_ORDERKEY, L_LINENUMBER, L_QUANTITY,
        L_EXTENDEDPRICE, L_DISCOUNT, L_TAX, L_RETURNFLAG,
        L_LINESTATUS, L_SHIPDATE, L_COMMITDATE,
        L_RECEIPTDATE, L_SHIPINSTRUCT, L_SHIPMODE,
        L_COMMENT)
SELECT *
FROM LINEITEM;
```

Performance Tuning with Advanced SQL

Outline

`MERGE` statement

`MERGE` statement can be used to select the rows from one or more sources for update or insertion into a relational table or into a relational view

`INTO` clause determines the target table or view to be used for updating or inserting into

`USING` clause determines the source of the data to be updated or inserted

`ON` clause determines the condition upon which the `MERGE` operation either updates or inserts

For each row in the target table for which the search condition is true, `MERGE` updates the row with corresponding data from the source table

If the condition is not true for any rows, then `MERGE` inserts into the target table based on the corresponding source table row

MERGE statement

For example, merge into `ALL_SUPPLIERS` relational table all rows from `SUPPLIER` relational table with the same values of an attribute `S_SUPPKEY`

MERGE statement

```
MERGE
INTO ALL_SUPPLIERS
USING SUPPLIER S
ON (ALL_SUPPLIERS.S_SUPPKEY = S.S_SUPPKEY)
...
```

Performance Tuning with Advanced SQL

Outline

MERGE with UPDATE clause

merge_update_clause determines the new column values of the target table

MERGE performs an update if a condition in ON clause is true

MERGE statement with UPDATE clause

```
MERGE
INTO ALL_SUPPLIERS
USING SUPPLIER S
ON (ALL_SUPPLIERS.S_SUPPKEY = S.S_SUPPKEY)
WHEN MATCHED THEN
UPDATE SET ALL_SUPPLIERS.S_ADDRESS = S.S_ADDRESS;
```

Performance Tuning with Advanced SQL

Outline

MERGE with INSERT clause

merge_insert_clause determines the values to be inserted into a column of target table if a condition in ON clause is false

MERGE statement with INSERT clause

```
MERGE
INTO ALL_SUPPLIERS
USING SUPPLIER S
ON (ALL_SUPPLIERS.S_SUPPKEY = S.S_SUPPKEY)
WHEN NOT MATCHED THEN
INSERT VALUES(S.S_SUPPKEY, S.S_NAME, S.S_ADDRESS,
              S.S_NATIONKEY,S.S_PHONE, S.S_ACCTBAL,
              S.S_COMMENT);
```

Performance Tuning with Advanced SQL

Outline

MERGE with WHERE clause

`where_clause` determines a condition that must be true to execute `UPDATE`/`INSERT` operation

The condition can refer to either the data source or the target table

If the condition is not true, then the database skips `UPDATE`/`INSERT`

MERGE statement with WHERE clause

```
MERGE
INTO ALL_SUPPLIERS
USING SUPPLIER S
ON (ALL_SUPPLIERS.S_SUPPKEY = S.S_SUPPKEY)
WHEN NOT MATCHED THEN
INSERT VALUES(S.S_SUPPKEY, S.S_NAME, S.S_ADDRESS,
              S.S_NATIONKEY,S.S_PHONE, S.S_ACCTBAL,
              S.S_COMMENT)
WHERE S.S_NAME = 'James Bond';
```

Performance Tuning with Advanced SQL

Outline

MERGE with DELETE clause

`DELETE where_clause` allows to clean up data in a table while populating or updating it

The only rows affected by this clause are those rows in the destination table that are updated by the merge operation

`DELETE where_clause` evaluates the updated value, not the original value that was evaluated by the `UPDATE SET ... WHERE` condition

If a row of the destination table meets the `DELETE` condition but is not included in a join defined by `ON` clause, then it is not deleted

MERGE statement with DELETE clause

```
MERGE
INTO ALL_SUPPLIERS
USING SUPPLIER S
ON (ALL_SUPPLIERS.S_SUPPKEY = S.S_SUPPKEY)
WHEN MATCHED THEN
UPDATE SET ALL_SUPPLIERS.S_ADDRESS = S.S_ADDRESS
DELETE WHERE S.S_NAME = 'James Bond';
```

Performance Tuning with Advanced SQL

Outline

MERGE with INSERT and UPDATE clause

Create a table ALL_SUPPLIERS with the same contents as SUPPLIER table,

CTAS statement

```
CREATE TABLE ALL_SUPPLIERS AS
(SELECT * FROM SUPPLIER);
```

Assume that some rows have been added or updated (the values of an attribute S_ADDRESS) in SUPPLIER table, replicate insertions and updates in ALL_SUPPLIERS table

MERGE statement with INSERT and UPDATE clauses

```
MERGE
INTO ALL_SUPPLIERS
USING SUPPLIER S
ON (ALL_SUPPLIERS.S_SUPPKEY = S.S_SUPPKEY)
WHEN NOT MATCHED THEN
INSERT VALUES(S.S_SUPPKEY, S.S_NAME, S.S_ADDRESS, S.S_NATIONKEY,
              S.S_PHONE, S.S_ACCTBAL, S.S_COMMENT)
WHEN MATCHED THEN
UPDATE SET ALL_SUPPLIERS.S_ADDRESS = S.S_ADDRESS;
```

References

Oracle® Database SQL Language Reference 19c Release 2 (12.2), `SELECT` statement, `INSERT` statement, `MERGE` statement