Improve ETL performance in BLU Acceleration

Karan Kumar (karan_gupta@in.ibm.com) 23 April 2015
Senior Staff Software Engineer
IBM

IBM DB2® with BLU Acceleration is a leap forward in database technology to improve query performance, particularly in the data warehouse (DW) and decision-support system (DSS) environments. It leverages dynamic in-memory, actionable compression, parallel vector processing, and data-skipping innovations from IBM Research to deliver a unique combination of performance, ease of use, and cost efficiency. In real-time reporting systems where the ETL process needs to update/delete data frequently from column-organized tables, it usually takes more time, resulting in performance issues for reports and dashboards. Learn how to improve these functions.

Introduction

The ETL (extract, transform, and load) process is the data integration part of the business intelligence reporting system that loads and keeps the data up to date in DW tables. Current data is needed so customers can create/access reports or dashboards that will give them a detailed view of underlying business data. Typical ETL processes load incremental data at specific intervals of time (every 30 minutes), requiring infrequent update/delete of data in DW tables. When using the DB2 BLU Acceleration feature with column organized tables (BLU tables), we notice that inserting records is more effective than updating records.

When performing ETL with column-organized tables, it is difficult to identify a list of columns for each record to update only the changed columns. Many times, we end up updating all the columns in the table, which results in slower performance because we are reading more pages as each column is stored in a different page. Our goal is to change the update/delete operation so it processes only the necessary columns to leverage maximum advantage of BLU Acceleration.

The example below will explain how we can convert update/delete to Insert single-column update in a column organized table. FDocumento is a column organized table without a primary column. It totally has 20 columns of which 5 are key columns, which means that the remaining 15 columns can be changed during the ETL run.

```sql
CREATE TABLE TEST.FDocumento ( 
  DOCUMENT_ID BIGINT NOT NULL, 
  DIRECTION_ID BIGINT NOT NULL, 
  COMPANY_ID BIGINT NOT NULL, 
  DOCUMENT_TYPE_ID BIGINT NOT NULL, 
```
INTERCHANGE_STATUS_ID BIGINT NOT NULL,
TP_IDENTIFIER_ID BIGINT NOT NULL,
RECEIVER_IDENTIFIER_ID BIGINT NOT NULL,
APPLICATION_ID BIGINT NOT NULL,
APPLICATION_ACCOUNT_ID BIGINT NOT NULL,
CONTROL_NUMBER_ID BIGINT NOT NULL,
RETENTION_PERIOD INTEGER NOT NULL,
PROCESS_DATE_ID BIGINT NOT NULL,
PROCESS_TIME_ID BIGINT NOT NULL,
AVAILABLE_DATE_ID BIGINT NOT NULL,
AVAILABLE_TIME_ID BIGINT NOT NULL,
ACCEPTED_DATE_ID BIGINT NOT NULL,
ACCEPTED_TIME_ID BIGINT NOT NULL,
DOCUMENT_CHARS BIGINT,
DOCUMENT_BYTES BIGINT,
DOCUMENT_SEGMENTS BIGINT
)
ORGANIZE BY COLUMN IN TEST_DATA;

When the ETL process has to update any record, it will execute the below statement in the regular run, which will try to update all the columns, including the unchanged columns. This results in reading more pages and ends up taking more time to complete the execution.

UPDATE TEST.F_DOCUMENT
SET DOCUMENT_CHARS=<SOURCE>.DOCUMENT_CHARS,
DOCUMENT_BYTES=<SOURCE>.DOCUMENT_BYTES,
DOCUMENT_SEGMENTS=<SOURCE>.DOCUMENT_SEGMENTS,
ACCEPTED_TIME_ID=<SOURCE>.ACCEPTED_TIME_ID,
AVAILABLE_TIME_ID=<SOURCE>.AVAILABLE_TIME_ID,
PROCESS_TIME_ID=<SOURCE>.PROCESS_TIME_ID,
ACCEPTED_DATE_ID=<SOURCE>.ACCEPTED_DATE_ID,
AVAILABLE_DATE_ID=<SOURCE>.AVAILABLE_DATE_ID,
APPLICATION_ID=<SOURCE>.APPLICATION_ID,
RECEIVER_PROTOCOL_ID=<SOURCE>.RECEIVER_PROTOCOL_ID,
PROCESS_DATE_ID=<SOURCE>.PROCESS_DATE_ID,
RETENTION_PERIOD=<SOURCE>.RETENTION_PERIOD,
APPLICATION_ACCOUNT_ID=<SOURCE>.APPLICATION_ACCOUNT_ID,
CONTROL_NUMBER_ID=<SOURCE>.CONTROL_NUMBER_ID,
RECEIVER_IDENTIFIER_ID=<SOURCE>.RECEIVER_IDENTIFIER_ID,
TP_IDENTIFIER_ID=<SOURCE>.TP_IDENTIFIER_ID
WHERE
DOCUMENT_ID=<SOURCE>.DOCUMENT_ID
AND DIRECTION_ID=<SOURCE>.DIRECTION_ID
AND COMPANY_ID=<SOURCE>.COMPANY_ID
AND DOCUMENT_TYPE_ID=<SOURCE>.DOCUMENT_TYPE_ID
AND INTERCHANGE_STATUS_ID=<SOURCE>.INTERCHANGE_STATUS_ID;

Steps to convert a multi-column update to a single-column update and insert data

1. Add an additional column ACTIVE SMALLINT to the table F_DOCUMENT.
2. For every insert operation to the table, make sure the ACTIVE column is set to 1.
3. For every update/delete operation in the table, make sure the ACTIVE column is set to 0.

CREATE TABLE TEST.F_DOCUMENT ( DOCUMENT_ID BIGINT NOT NULL,
DIRECTION_ID BIGINT NOT NULL,
COMPANY_ID BIGINT NOT NULL,
DOCUMENT_TYPE_ID BIGINT NOT NULL,
INTERCHANGE_STATUS_ID BIGINT NOT NULL,
TP_IDENTIFIER_ID BIGINT NOT NULL,
RECEIVER_IDENTIFIER_ID BIGINT NOT NULL,
APPLICATION_ID BIGINT NOT NULL,
APPLICATION_ACCOUNT_ID BIGINT NOT NULL,
CONTROL_NUMBER_ID BIGINT NOT NULL,
RETENTION_PERIOD INTEGER NOT NULL,
PROCESS_DATE_ID BIGINT NOT NULL,
PROCESS_TIME_ID BIGINT NOT NULL,
AVAILABLE_DATE_ID BIGINT NOT NULL,
AVAILABLE_TIME_ID BIGINT NOT NULL,
ACCEPTED_DATE_ID BIGINT NOT NULL,
ACCEPTED_TIME_ID BIGINT NOT NULL,
DOCUMENT_CHARS BIGINT,
DOCUMENT_BYTES BIGINT,
DOCUMENT_SEGMENTS BIGINT,
ACTIVE SMALLINT
)
ORGANIZE BY COLUMN IN TEST_DATA;

Once the table has the ACTIVE column added, the ETL process needs to execute two statements for each update. The first one will be an update statement to set the ACTIVE column to 0, which results in improved performance because the process has to read fewer pages. And the second statement is to insert a new record with the ACTIVE column as 1.

UPDATE TEST.F_DOCUMENT
    SET ACTIVE = 0
WHERE
    DOCUMENT_ID=<SOURCE>.DOCUMENT_ID
    AND DIRECTION_ID=<SOURCE>.DIRECTION_ID
    AND COMPANY_ID=<SOURCE>.COMPANY_ID
    AND DOCUMENT_TYPE_ID=<SOURCE>.DOCUMENT_TYPE_ID
    AND INTERCHANGE_STATUS_ID=<SOURCE>.INTERCHANGE_STATUS_ID;

INSERT INTO TEST.F_DOCUMENT
    (DOCUMENT_ID,
     DIRECTION_ID,
     COMPANY_ID,
     DOCUMENT_TYPE_ID,
     INTERCHANGE_STATUS_ID,
     TP_IDENTIFIER_ID,
     RECEIVER_IDENTIFIER_ID,
     APPLICATION_ID,
     APPLICATION_ACCOUNT_ID,
     CONTROL_NUMBER_ID,
     RETENTION_PERIOD,
     PROCESS_DATE_ID,
     PROCESS_TIME_ID,
     AVAILABLE_DATE_ID,
     AVAILABLE_TIME_ID,
     ACCEPTED_DATE_ID,
     ACCEPTED_TIME_ID,
     DOCUMENT_CHARS,
     DOCUMENT_BYTES,
     DOCUMENT_SEGMENTS,ACTIVE)
VALUES
    (<SOURCE>.DOCUMENT_ID,
     <SOURCE>.DIRECTION_ID,
     <SOURCE>.COMPANY_ID,
     <SOURCE>.DOCUMENT_TYPE_ID,
     <SOURCE>.INTERCHANGE_STATUS_ID,
     <SOURCE>.TP_IDENTIFIER_ID,
     <SOURCE>.RECEIVER_IDENTIFIER_ID,
     <SOURCE>.APPLICATION_ID,
     <SOURCE>.APPLICATION_ACCOUNT_ID,
     <SOURCE>.CONTROL_NUMBER_ID,
Similarly, you can change your ETL process to execute the single-column update for each delete operation.

```
UPDATE TEST.F_DOCUMENT
SET ACTIVE = 0
WHERE
  DOCUMENT_ID=<SOURCE>.DOCUMENT_ID
  AND DIRECTION_ID=<SOURCE>.DIRECTION_ID
  AND COMPANY_ID=<SOURCE>.COMPANY_ID
  AND DOCUMENT_TYPE_ID=<SOURCE>.DOCUMENT_TYPE_ID
  AND INTERCHANGE_STATUS_ID=<SOURCE>.INTERCHANGE_STATUS_ID;
```

Make sure all your select queries for `F_DOCUMENT` has `ACTIVE='1'` in the `WHERE` clause. Later, as part of the maintenance activity you can delete all the records that have `ACTIVE='0'`.

```
SELECT
  DOCUMENT_ID,
  DIRECTION_ID,
  COMPANY_ID,
  DOCUMENT_TYPE_ID,
  DOCUMENT_CHARS,
  DOCUMENT_BYTES,
  DOCUMENT_SEGMENTS,
FROM
  TEST.F_DOCUMENT
WHERE
  DOCUMENT_ID=<SOURCE>.DOCUMENT_ID
  AND DIRECTION_ID=<SOURCE>.DIRECTION_ID
  AND COMPANY_ID=<SOURCE>.COMPANY_ID
  AND DOCUMENT_TYPE_ID=<SOURCE>.DOCUMENT_TYPE_ID
  AND INTERCHANGE_STATUS_ID=<SOURCE>.INTERCHANGE_STATUS_ID
  AND ACTIVE = 1;
```

**Things to keep in mind**

1. Following these techniques will increase the number of records in the table, as we are inserting new records with the ACTIVE column as 1 for each update and changing the existing records to have ACTIVE='0'. We need not worry about the additional records as BLU Acceleration gives us better storage management.
2. You should keep a weekly/monthly maintenance activity to remove records with ACTIVE='0' so we can reclaim storage space and improve performance.
3. IMPORTANT NOTE: All `SELECT` statements should always have "ACTIVE=1" in the `WHERE` clause.
Data representation in the table

The figure below shows how data would look like in your table when the Update or Delete statements are changed to an Insert/single-column update statements.

Figure 1. Select statement with ACTIVE=’1’ in the WHERE clause

<table>
<thead>
<tr>
<th>DOCUMENT ID</th>
<th>DIRECTION ID</th>
<th>COMPANY ID</th>
<th>DOCUMENT TYPE ID</th>
<th>INTERCHANGE STATUS ID</th>
<th>Other columns</th>
<th>ACTIVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>12</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>10</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>11</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>9</td>
<td>...</td>
<td>1</td>
</tr>
</tbody>
</table>

Data in F_DOCUMENT after Update for Document id 1 and 3

<table>
<thead>
<tr>
<th>DOCUMENT ID</th>
<th>DIRECTION ID</th>
<th>COMPANY ID</th>
<th>DOCUMENT TYPE ID</th>
<th>INTERCHANGE STATUS ID</th>
<th>Other columns</th>
<th>ACTIVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>12</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>10</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>11</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>9</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>12</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>11</td>
<td>...</td>
<td>1</td>
</tr>
</tbody>
</table>

Data in F_DOCUMENT after Update for Document id 1 and Delete for Document id 4

<table>
<thead>
<tr>
<th>DOCUMENT ID</th>
<th>DIRECTION ID</th>
<th>COMPANY ID</th>
<th>DOCUMENT TYPE ID</th>
<th>INTERCHANGE STATUS ID</th>
<th>Other columns</th>
<th>ACTIVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>12</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>10</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>11</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>9</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>12</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>11</td>
<td>...</td>
<td>0</td>
</tr>
</tbody>
</table>

Data in F_DOCUMENT after maintenance activity to remove ACTIVE=0 records

<table>
<thead>
<tr>
<th>DOCUMENT ID</th>
<th>DIRECTION ID</th>
<th>COMPANY ID</th>
<th>DOCUMENT TYPE ID</th>
<th>INTERCHANGE STATUS ID</th>
<th>Other columns</th>
<th>ACTIVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>10</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>6</td>
<td>11</td>
<td>...</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>12</td>
<td>...</td>
<td>1</td>
</tr>
</tbody>
</table>

Conclusion

In DB2 BLU Acceleration, each update is internally converted into a Delete and an Insert operation. The old row is deleted and a new row with the updated values are inserted. This method
delays the delete operation to a later point in time; during your maintenance window, and therefore you gain performance by decreasing the time taken.

We saw a significant improvement during the update operation in the ETL process. It took just a few seconds to update 100000 records now, compared to the earlier time taken, which was in minutes.

Another important point to remember to increase performance is that if the primary key exists within the table, the ETL operation will be faster in BLU Acceleration.
Resources

• Learn more about DB2 BLU Acceleration.
• The Information Management area on developerWorks provides resources for architects, developers, and engineers.
• Stay current with developer technical events and webcasts focused on a variety of IBM products and IT industry topics.
• Follow developerWorks on Twitter.
• Watch developerWorks demos ranging from product installation and setup demos for beginners, to advanced functionality for experienced developers.
• Get involved in the developerWorks Community. Connect with other developerWorks users while you explore developer-driven blogs, forums, groups, and wikis.
About the author

Karan Kumar

Karan Kumar is a senior software developer with the IBM Commerce business unit. He is a DataStage ETL developer for the IBM Sterling product, B2B Analytics, since May 2013. To add to that, Karan has around seven years of QA experience in OLTP and DW application. He has specialized in testing, and has implemented test processes and handled many releases.

© Copyright IBM Corporation 2015
Trademark
(www.ibm.com/developerworks/ibm/trademarks/)