Improving PowerCenter Performance with IBM DB2 Range Partitioned Tables
Abstract
PowerCenter can leverage the range partitioning capability in IBM DB2 to reduce the time taken to extract and load a set of DB2 range partitioned tables. This article describes how to improve the PowerCenter Integration Service performance when reading data from DB2 range partitioned tables or when loading data to DB2 range partitioned tables.

Supported Versions
- PowerCenter 8.6.1 HotFix 11 - 9.1.0
- IBM DB2 9.x

Table of Contents
Overview .......................................................................................................................................... 2
DB2 Range Partitioning ............................................................................................................... 2
DB2 Range Partitioning with Other Partitioning Schemes ........................................................... 3
Example.................................................................................................................................... 3
PowerCenter Partitioning Schemes................................................................................................. 4
Pipeline Partitioning ..................................................................................................................... 4
Example.................................................................................................................................... 5
Dynamic Partitioning .................................................................................................................... 5
Configuring a Session for Pipeline Database Partitioning ............................................................... 5
Configuring a Session for Dynamic Database Partitioning ............................................................. 5
Validation Queries ........................................................................................................................... 5
Rules and Guidelines for Partitioning ................................................................................................ 6

Overview
In PowerCenter, you can add partitions to increase the speed of the query. Each partition represents a subset of the data that the PowerCenter Integration Service processes. You can enable session partitioning to run a session with multiple processes to improve the overall session performance. A method of session partitioning in PowerCenter is database partitioning, which you can use for IBM DB2 sources and targets.

DB2 Range Partitioning
From IBM DB2 9.x, you can select range partition or table partition when you have data distributed across data partitions based on the values in one or more columns. The data partitions can be in different tablespaces, in the same tablespace, or both. When you use DB2 range partitioning, it increases the query performance since the database query optimizer searches only the tablespaces where the particular data range is present.
If the database tablespace consists of N logical nodes on a multi-node DB2 database, the database creates N logical DB2 tables that are identical. DB2 assigns a node value to each of the logical tables and routes the data to the appropriate target using a partition key hashing algorithm.
The following syntax shows how to create a sample DB2 table with range partitioning:

```
CREATE TABLE orders (id INT, shipdate DATE)
```
PARTITION BY RANGE (shipdate)
{
    STARTING '1/1/2011' ENDING '12/31/2011'
    EVERY 3 MONTHS
}

The table contains four data partitions with each partition containing three months data. If you need to get data for the month of August, you need to query only the third partition of the orders table.

When reading data from a range partitioned DB2 table, the PowerCenter Integration Service uses the table partitioning scheme in the database to increase performance. When loading data to a range partitioned DB2 table, DB2 loads data to the appropriate table partitions. By using the partitioning information of the target, the data commits faster and results in increased load performance.

**DB2 Range Partitioning with Other Partitioning Schemes**

In DB2 9.x, you can use table partitioning in isolation or in combination with other data organization schemes. Each clause of the CREATE TABLE statement includes an algorithm to indicate how you should organize data.

The following constructs demonstrate the levels of data organization in DB2 9.x:

1. **DISTRIBUTE BY.** Spreads data evenly across database partitions. You need to enable the Database Partitioning Feature (DPF) to use the construct. DPF enables intraquery parallelism and distributes the load across each database partition.

2. **PARTITION BY.** Groups rows with similar values of a single dimension in the same data partition. This concept is known as table partitioning or range partitioning.

3. **ORGANIZE BY.** Groups rows with similar values on multiple dimensions in the same table extent. This concept is known as multidimensional clustering (MDC).

**Example**

The following example uses all the three different data organization schemes in the DB2 9.x database:

```sql
CREATE TABLE demo_custinfo(c_custid INTEGER NOT NULL, c_firstname VARCHAR(30), c_age INTEGER, c_dob DATE, c_quarter integer, PRIMARY KEY(c_custid)) DISTRIBUTE BY HASH(c_custid) PARTITION BY RANGE(c_dob)(PARTITION "P1" STARTING FROM '1/1/1754' INCLUSIVE ENDING AT '12/31/1949' INCLUSIVE IN TS1, PARTITION "P2" STARTING FROM '1/1/1950' INCLUSIVE ENDING AT '12/31/1999' INCLUSIVE IN TS2, PARTITION "P3" STARTING FROM '1/1/2000' INCLUSIVE ENDING AT '12/31/9999' INCLUSIVE IN TS3) ORGANIZE BY DIMENSIONS(c_quarter)
```
The following figure shows how the above table demo_custinfo is distributed in the database:

```
Node 1
  └─── DISTRIBUTED
      └─── PARTITION
        └─── ORGANIZE
```

DB2 9.x allows data partitions to be easily added or removed from the table without having to take the database offline. This ability can be particularly useful in a data warehouse environment where you often need to load or delete data to run decision-support queries. For example, a typical insurance data warehouse may have three years of claims history. As each month is loaded and rolled-in into the warehouse, the oldest month can be archived and removed from the active table. This method of rolling out data partitions is also more efficient as it does not need to log delete operations, which would be the case when deleting specific data ranges.

**PowerCenter Partitioning Schemes**

PowerCenter leverages range partitioning feature of DB2 tables based on one of the following partitioning schemes:

- **Pipeline partitioning**
- **Dynamic partitioning**

**Pipeline Partitioning**

In the pipeline partitioning scheme, you can provide additional partition points for a session so that PowerCenter generates processes for additional database connections to load or read data. You can select database partitioning as the partition type for both DB2 sources and targets. Since you predefine the number of partitions before the session run starts, this type of partitioning is also termed as static partitioning.

When you run a mapping with a DB2 source with the database partitioning scheme, the PowerCenter Integration Service first queries the database for the table partitioning information. If you provide more number of partition points than the actual number of nodes on which the table resides, the PowerCenter Integration Service creates database connections equivalent to the number of session partition points. This results in some of the database connections being idle. If you provide fewer partition points compared to the actual table partitions, then some session
partitions would have one database partition associated with it while others would have more database partitions.

For optimal performance, provide as many partition points as the number of database nodes on which the partitioned table resides. In the Workflow Monitor, the target load statistics is based on the total load on individual partitions or database nodes and not on the specific tablespaces.

**Example**

There are four database nodes and three session partitions. After querying the DB2 nodes, there are four instances of the same target and a partitioning key for each. Using an internal hash algorithm, each session partition associates itself with one database partition. So three partitions associates with three nodes, and then the hash algorithm assigns the remaining database partition to a session partition with an associated database partition.

**Dynamic Partitioning**

When you use the dynamic partitioning scheme, the PowerCenter Integration Service identifies the number of partitions in the DB2 table to establish the number of connections at session run-time. The PowerCenter Integration Service loads or reads from a DB2 table based on the partition information available for the table at the database.

**Note:** You do not need to provide the number of partition points before the session run starts.

### Configuring a Session for Pipeline Database Partitioning

1. Create a session for a mapping that contains a DB2 source and a DB2 target.
2. Edit the session properties and configure partitioning information on the Partitions view of the Mapping tab.
3. Click the Add a Partition Point button. You can add as many partition points as there are nodes in the DB2 database for optimal performance.
4. You can determine the number of logical nodes in a DB2 system from the following database query:
   ```sql
   SELECT DISTINCT(NODENUM) FROM SYSCAT.NODEGROUPDEF
   ```
5. Choose Database Partitioning as the partitioning type.
6. Run the session.

### Configuring a Session for Dynamic Database Partitioning

1. Create a session for a mapping that contains a DB2 source and a DB2 target.
2. Edit the session properties and on the Partitions view of the Mapping tab.
3. Provide one partition point for the source and target.
4. Set Database Partitioning as the partitioning type.
5. On the Configuration Object tab, select dynamic partitioning based on source partitioning.
6. Run the session.

### Validation Queries

The following database queries help users to validate the data distribution in the database after a session runs:

- Use the following query to get details about the data partitions for the particular table:
  ```sql
  DESCRIBE DATA PARTITIONS FOR TABLE demo_custinfo SHOW DETAIL
  ```
• Use the following query to get details about the tablespaces in the database and the nodes on which the tablespaces reside:

```sql
SELECT NODEGROUP.NODENUM, TABLESPACES.TBSPACE, TABLESPACES.TBSPACEID FROM SYSCAT.NODEGROUPDEF AS NODEGROUP JOIN SYSCAT.TABLESPACES AS TABLESPACES ON NODEGROUP.NGNAME = TABLESPACES.NGNAME WHERE NODEGROUP.IN_USE = 'Y' ORDER BY TABLESPACES.TBSPACEID
```

• Use the following query to fetch the records of the table that are stored in Node 0 of the database:

```sql
SELECT * FROM demo_custinfo WHERE (NODENUMBER(ods.demo_custinfo.c_custid)=0)
```

Where, demo_custinfo is the table name, ods is the table owner, and c_cust_id is the column on which has the hash partition defined.

• Use the following query to check if a DB2 table was created using DISTRIBUTE BY HASH and ORGANIZE BY DIMENSIONS constructs:

```sql
SELECT PARTITION_MODE, CLUSTERED FROM SYSCAT.TABLES WHERE TABNAME = 'DEMO_CUSTINFO'
```

• Use the following query to get a list of columns used with DISTRIBUTE BY HASH:

```sql
SELECT COLNAME FROM SYSCAT.COLUMNS WHERE TABNAME = 'DEMO_CUSTINFO' AND PARTKEYSEQ != 0
```

• Use the following query to get a list of columns used with ORGANIZE BY DIMENSIONS:

```sql
SELECT COLNAME FROM SYSCAT.COLUSE WHERE TABNAME = 'DEMO_CUSTINFO'
```

• Use the following query to get a list of table partitions and tablespaces for a particular table:

```sql
SELECT SUBSTR(DATAPARTITIONNAME,1,15) AS TABLEPART, TBSPACEID FROM SYSCAT.DATAPARTITIONS WHERE TABNAME = 'DEMO_CUSTINFO'
```

• Use the following query to get all the partition numbers of the table which has data and the total number of records stored in each of those partitions for a particular table. You can use the query to validate the target load statistics in the Workflow Monitor after loading a DB2 range-partitioned table:

```sql
SELECT DBPARTITIONNUM(c_custid) AS PARTN_NO, COUNT(*) AS TOT_REC FROM demo_custinfo GROUP BY DBPARTITIONNUM(c_custid)
```

Rules and Guidelines for Partitioning

• Depending on the number of partitions, the number of TCP ports that DB2 uses may need to be increased. By default, DB2 uses 64 TCP port numbers from 6000 to 6063. You can increase the port range by setting the DB2 registry variable, DB2ATLD_PORTS to 6000:9999.

• A partitioning column can be any of the DB2 base datatypes except for LOBs and LONG VARCHAR columns.

Author

Anju Andrews
Senior Software Engineer, QA