Partitioning Oracle Sources in PowerCenter
Abstract
You can partition Oracle database table reads to increase performance. This article explains some techniques for partitioning Oracle source data.

Supported Versions
- Informatica PowerCenter 9.x

Table of Contents

Overview ..................................................................... 2
Database Partitioning ............................................................. 3
Key Range Partitioning ............................................................ 3
  Configuring Key Range Partitioning .................................................. 3
Pass-Through Partitioning .......................................................... 5
MOD Function .................................................................. 5
  Configuring the Filter Condition with the MOD Function. .................. 6
Function-Based Index. ...................................................... 6
Filter on ROWID ................................................................ 7
Filter on ROWID Example .......................................................... 7
CreatePartitionInfo Mapping ....................................................... 8
DimReadTest Mapping. ......................................................... 10

Overview
When an Oracle source is the bottleneck for PowerCenter session performance, you can increase performance by configuring source partitioning in the session properties. With source partitioning, the PowerCenter Integration Service reads multiple Oracle rows in parallel.

The PowerCenter Integration Service creates a reader thread for each pipeline partition. For a relational database source, each partition issues an SQL statement to access the source data. To optimize performance, the SQL statements should create efficient and fairly equal-sized data sets.

You can use the following approaches for partitioning Oracle source data:

Database partitioning
  Creates a pipeline for each physical table partition in the database.

Key range partitioning
  Distributes source rows into partitions based on the values of a port or set of ports.

Pass-through partitioning
  Passes rows into static partitions based on a filter condition for each partition.
Database Partitioning

You can optimize session performance by using the database partitioning partition type for Oracle sources. Use database partitioning for Oracle sources whenever possible.

Database partitioning creates a pipeline for each physical table partition in the Oracle database. When you use database partitioning, the PowerCenter Integration Service queries the database system for table partition information and fetches data into the session partitions. You can use any number of session partitions and any number of database partitions. When the pipeline partitions do not equal the database partitions, the PowerCenter Integration Service generates SQL queries for each database partition and distributes the data among the session partitions equally. However, you can improve performance when the number of pipeline partitions equals the number of database partitions.

For Oracle sources that use composite partitioning, you can increase performance when the number of pipeline partitions equals the number of database subpartitions. For example, if an Oracle source contains three partitions and two subpartitions for each partition, set the number of pipeline partitions at the source to six.

You can use other partitioning methods when the Oracle table is not partitioned or when the table is partitioned in a way that is not useful for extracting data sets of equal size.

Key Range Partitioning

Configure key range partitioning to partition Oracle data based on the value of a port or set of ports. With key range partitioning, the PowerCenter Integration Service distributes rows of source data based the ports that you define as partition keys. The PowerCenter Integration Service compares the port value to the range values for each partition and sends rows to the appropriate partition.

An advantage of key range partitioning is that you can use key range partitioning with PowerCenter dynamic partitioning. With dynamic partitioning, the PowerCenter Integration Service scales the number of session partitions at run time based on the source database partitions or the number of nodes in a grid.

Use key range partitioning for columns that have an even distribution of data values. Otherwise, the partitions might have unequal size. For example, a column might have 10 rows between key values 1 and 1000 and the column might have 999 rows between key values 1001 and 2000.

A disadvantage of key range partitioning is that Oracle might perform a table scan for each partition. Multiple queries might run slower than one query because the Oracle database runs multiple concurrent table scans.

Note: If you create an database index on the partition key, the Oracle database might perform a table scan anyway. The index might not increase performance.

With key range partitioning, a query for one partition might return rows sooner than another partition. Or, one partition can return rows while the other partitions are not returning rows. This situation occurs when the rows in the table are in a similar order as the key range. One query might be reading and returning rows while the other queries are reading and filtering the same rows.

Configuring Key Range Partitioning

Configure key range partitioning to partition Oracle data based on port values.

1. In the Workflow Manager, double-click the session you want to partition.
2. Select the Source Qualifier partition point on the Partitions view of the Mapping tab.
3. Click Add to add partitions.
4. Choose **Key Range Partitioning** and click **OK**.
The Edit **Partition Key** dialog box appears.
5. Choose at least one port to use as the key.
6. Click **Edit Keys** to enter a range of values for each partition.

![PowerCenter Integration Service interface](image)

**Pass-Through Partitioning**

With pass-through partitioning, the PowerCenter Integration Service passes all rows from one partition point to the next partition point without redistributing data across partitions. All rows in a partition stay in that partition after crossing a partition point. Pass-through partitioning is the default partitioning method.

When the session has pass-through partitioning, you can configure a filter condition for each static partition. You can use the following filters to partition data:

- **MOD function**
  - Use a MOD function to filter data into different partitions based on a value of a numeric column.

- **ROWID**
  - Partition data by ROWID. Oracle can perform direct reads on rows of data by ROWID.

**MOD Function**

You can create a filter condition to partition data with the Oracle MOD function. The Oracle MOD function receives two numeric input values and returns the remainder. For example, \( \text{MOD}(4, 2) = 0 \) and \( \text{MOD}(4, 3) = 1 \).

To use the MOD function to filter rows, define the Source Qualifier partition type as pass-through. The PowerCenter Integration Service generates a WHERE clause that includes any filter condition you enter in the session properties.

Enter the filter condition for each partition on the **Transformations** view of the **Mapping** tab. The filter overrides any filter condition that you set in the Designer when you configure the Source Qualifier transformation.

For example, if the session has two partitions you might configure the following function for the first partition:

\[ \text{MOD}(\text{columnName}, 2) = 0 \]
Configure the following function for the second partition:

\[
\text{MOD}(\text{columnName}, 2) = 1
\]

When the value of the column is an even number, the first partition receives the row. When the column value is odd, the second partition receives the row.

When you configure the MOD function, choose a numeric column that has an even distribution of values. You can use a key column. Do not use a column that has few values because the partitions will be unequal sizes. For example, if a column can contain zero or one, you cannot partition the row into more than two partitions.

**Configuring the Filter Condition with the MOD Function**

Create a MOD statement for each partition.

The following example shows how to use the MOD function to partition Oracle source data in a session with four partitions.

1. In the session properties, configure pass-through partitioning.
2. Configure four partitions.
3. On the Source Filter attribute, enter the following filter conditions for each partition:

   - Partition#1: \(\text{MOD}(\text{InvoiceID}, 4) = 0\)
   - Partition#2: \(\text{MOD}(\text{InvoiceID}, 4) = 1\)
   - Partition#3: \(\text{MOD}(\text{InvoiceID}, 4) = 2\)
   - Partition#4: \(\text{MOD}(\text{InvoiceID}, 4) = 3\)

The following figure shows where to configure the MOD functions in session properties:

![Function-Based Index](image)

**Function-Based Index**

You can define a function-based index on a MOD function to increase performance and eliminate full table scans. With a function-based index, Oracle performs fast index range scans. Using a function-based index can affect all SQL statements that have the matching predicate.

To create a function-based index use the following SQL syntax:

```
create index invoiceID-mod4-idx on MOD(InvoiceID, 4)
```
If you change the number of partitions in the session, you need to build the function-based index to match the SELECT query.

**Note:** Without a function-based index, MOD partitioning performance is similar to range-based partitioning. Each query typically requires a table scan.

### Filter on ROWID

When a session reads all the rows in a table, you can configure the session partitions to read a table by the Oracle ROWID. The ROWID is the physical address of a row in the table. Oracle performs direct reads on a row using ROWID.

To filter with ROWID, you need to determine the ROWID values in the database table. You can configure a SQL statement that returns the ROWID for specific rows in a table. For example, the following SELECT statement returns the ROWID and last name of each customer in department 20:

```sql
SELECT ROWID, last_name FROM Invoices WHERE Dept = 20
```

To partition the table read using ROWID, run a SQL query that returns a minimum and maximum ROWID for each partition you plan to have in the session. After you determine the minimum and maximum ROWID, configure the Source Read partition filters using the minimum and maximum ROWID for each partition.

For example, a session that reads the Invoices table has four partitions. Configure the following SQL statement to return the minimum and maximum ROWID for each partition:

```sql
SELECT min(ROWID), max(ROWID), tile from (select ROWID, ntile(4) over (order by ROWID) as tile from Invoices) group by tile order by 1
```

The query returns the following values:

<table>
<thead>
<tr>
<th>MIN ROWID</th>
<th>MAX ROWID</th>
<th>TILE</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAATtYAA9AAAAALAAA</td>
<td>AAATtYAA9AAAAK5AAAJ</td>
<td>1</td>
</tr>
<tr>
<td>AAATtYAA9AAAAK5AAAK</td>
<td>AAATtYAA+AAAA2zAAAA</td>
<td>2</td>
</tr>
<tr>
<td>AAATtYAA+AAAA2zAABB</td>
<td>AAATtYAA+AAA9DAAI</td>
<td>3</td>
</tr>
<tr>
<td>AAATtYAA+AAA9DAAJ</td>
<td>AAATtYAA+AAA1M1AAJ</td>
<td>4</td>
</tr>
</tbody>
</table>

In the session properties, configure the filter condition for each of the partitions. Configure the filter conditions with the minimum and maximum ROWID values from the SQL query.

- `rowid between chartorowid('AAATtYAA9AAAAALAAA') and chartorowid('AAATtYAA9AAAAK5AAAJ')`
- `rowid between chartorowid('AAATtYAA9AAAAK5AAAK') and chartorowid('AAATtYAA+AAAA2zAAAA')`
- `rowid between chartorowid('AAATtYAA+AAAA2zAABB') and chartorowid('AAATtYAA+AAA9DAAI')`
- `rowid between chartorowid('AAATtYAA+AAA9DAAJ') and chartorowid('AAATtYAA+AAA1M1AAJ')`

A disadvantage to partitioning with ROWID is that you must maintain the minimum and maximum values for the filter conditions. When the table contains new rows, and the filter conditions do not contain the new ROWID values, the session does not select the rows. You can automate the process to maintain the ROWID values in the filter conditions.

### Filter on ROWID Example

The following example shows how to partition a session with the Oracle pseudocolumn ROWID. You can download the workflows for this example from the following location: [https://communities.informatica.com/docs/DOC-8126](https://communities.informatica.com/docs/DOC-8126).

The session contains 16 partitions. The example has the following workflows:

**CreatePartitionInfo**

- Runs an SQL query to determine the ROWIDs in a table. Generates a parameter file that contains a set of SQL WHERE clauses based on the current ROWID values.

**DimReadTest**

- Reads the parameter file to determine how to partition the source rows in the session. The DimReadTest workflow performs a direct read by ROWID for each partition.
CreatePartitionInfo Mapping

Create a mapping that returns a parameter file containing the ROWID based WHERE clauses.

The mapping contains the following objects:

**Partition_RowIDs**
Oracle source that contains the data you want to read in a session.

**SQ_Partition_RowIDs**
Source Qualifier that contains the SQL query that retrieves the minimum and maximum ROWID values for each partition you configure in DimReadTest.

**Exprtrans**
Expression transformation that returns a WHERE clause in the output port for each partition.

**Partition_Where_Clauses**
Flat file target that is the parameter file for DimReadTest.

**SQ_Partition_RowIDs Source Qualifier**

The mapping has two input parameters:

**$$number_of_partitions**
The number of partitions to include in the DimReadTest session. For this example, the value is 16.

**$$source_table_name**
The name of the source table to query. The table name is DIM_COM_ACCOUNT_TERM1.

**Note:** The session does not use an input parameter file. For this example, you can manually modify the values in the Mapping Designer.

The SQL query in the Source Qualifier Properties tab contains the following text:

```
SELECT MIN(rowid) AS MIN_ROWID, MAX(rowid) AS MAX_ROWID, PARTITION_NUM
FROM (SELECT rowid, ntile($$number_of_partitions) OVER (ORDER BY rowid) AS PARTITION_NUM FROM $source_table_name$) GROUP BY PARTITION_NUM ORDER BY 1
```
Exptrans Expression Transformation

The Expression transformation receives the MIN and MAX ROWIDs for 16 partitions.

The Expression transformation has the following expression in the Where_Clause_ output port:

'$$where_p' || PARTITION_NUM || '=rowid between chartorowid(' || CHR(39) || MIN_ROWID || CHR(39) || ') and chartorowid(' || CHR(39) || MAX_ROWID || CHR(39) || ')'"

Target

Configure the target session properties and select the Use Header Command Output Header option. The PowerCenter Integration Service adds a header to the target. It appends the contents of the partition_where_header.txt file.

Configure the Header Command field to generate a header row. The Header Command contains the following text:

`cat /u01/app/infa_shared/presales/sdorcey/TgtFiles/partition_where_header.txt`

The `partition_where_header.txt` file contains the following text:

```
[Global]```
Any session include use the parameter file.

The following figure shows the Header Options and Header Command attributes for the target session properties:

**DimReadTest Mapping**

The DimTest mapping reads the parameter file from the CreatePartitionInfo mapping in order to determine how to partition the source data and perform fast parallel reads. The DimTest mapping reads source rows in 16 partitions. The mapping passes each row through a Filter transformation.

**Note:** The Filter transformation returns no rows in the target. You can change the example to include different transformations.

The DimReadTest mapping contains the following objects:

**DIM_COM_ACCOUNT_TERM1**

Oracle source that contains the data you want to read in the session.

**SQ_DIM_COM_ACCOUNT_TERM1**

Source Qualifier that contains the SQL query that retrieves the minimum and maximum ROWID values for each partition you configure in DimReadTest.

**FILTRANS**

Filter transformation that returns no rows for this example. The expression is set to False.
**DIM_COM_ACCOUNT_TERM**

Target that receives no rows.

The following figure shows the parameter file path in the General Options section of the Properties tab:

![Parameter File Path](image)

**DimReadTest Source Qualifier**

Configure the **Source Filter** attribute for each partition on the Mapping tab of the session properties. To navigate to the Source Filter attribute, select the Source Qualifier SQ_DIM_COM_ACCOUNT_TERM in the **Navigation** panel. Scroll to the **Source Filter** attribute.

Enter an attribute for each of the 16 partitions. The naming convention for the attribute value is "$$where_pN" where "N" is the partition number.

For example:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition#1</td>
<td>$$where_P1</td>
</tr>
<tr>
<td>Partition#2</td>
<td>$$where_P2</td>
</tr>
<tr>
<td>Partition#3</td>
<td>$$where_p3</td>
</tr>
</tbody>
</table>

The following figure shows the Mapping tab:
If you change the number of partitions in the DimReadTest session, change the `$number_of_partitions` parameter in the CreatePartitionInfo mapping to match the number of partitions in the session.

**Authors**

Ellen Chandler  
Principal Technical Writer  

Stan Dorcey  
Sr. Product Specialist