Performance Improvement of Post-ETL in OMOP CDM

Presenter: Wai Yi Man

**Background:** Transformation of real-world data into the Observational Medical Outcomes Partnership (OMOP) Common Data Model (CDM) is not simply an Extract-Transform-Load (ETL) process. It also requires the building of primary keys, indexes and constraints using OHDSI provided sequential SQL code before the standardized data can be used. This implementation is a post-ETL operation, whose execution time depends closely on the data dimension, and can be very time consuming. The simpler approach for building primary keys, indexes, and constraints on OMOP CDM tables is running one after the other the OMOP CDM GitHub provided SQL scripts, one for PK, one for indexes and one for constraints. However, this operation could become significantly more efficient by splitting and merging these files in a way that allows concurrency to be used.

**Result 1:** The time complexity of execution of PK and indexes

(Where \( R \) is the number of rows in the \( T_i \) table, and \( n \) is the number of tables)

\[
O\left(R(T_1)\right) + O\left(R(T_2)\right) + \ldots + O\left(R(T_n)\right) = \sum_{i=1}^{n} O\left(R(T_i)\right) \quad \rightarrow \quad \max_{1 \leq i \leq n} O\left(R(T_i)\right)
\]

**Result 2:** The time complexity execution of foreign keys when processed on target tables \( T_i \) with parent tables \( P_j \)

(Where \( z \) is the number of shift)

\[
\sum_{i=1}^{n} \left( \sum_{j=1}^{m} O\left(R(T_i, P_j)\right) \right) \rightarrow \sum_{i=1}^{n-1} \max_{1 \leq j \leq n} O\left(R(T_i, P_j)\right)
\]

\[ z = 0 \Rightarrow i = \text{mod}(j + z - 1, n) + 1 \]

**Methods**

1. Running PK and indexes .sql files in parallel

<table>
<thead>
<tr>
<th>ID</th>
<th>File name</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>pk_ids_location.sql</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>pk_ids_care_site.sql</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>pk_ids_provider.sql</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>pk_ids_person.sql</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>pk_ids_observation_period.sql</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>pk_ids_visit_occurrence.sql</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>pk_ids_visit_detail.sql</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>pk_ids_measurement.sql</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>pk_ids_drug_exposure.sql</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>pk_ids_procedure_occurrence.sql</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>pk_ids_drug_era.sql</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>pk_ids_condition_era.sql</td>
<td></td>
</tr>
</tbody>
</table>

2. Shift algorithm that identifies .sql files suitable to run in parallel

- No shift
  - condition_occurrence
  - drug_exposure
  - measurement
  - observation
  - procedure_occurrence
  - visit_occurrence

- 1st shift
  - device_exposure
  - drug_exposure
  - measurement
  - observation
  - procedure_occurrence
  - visit_occurrence
  - device_exposure
  - drug_exposure
  - measurement
  - observation
  - procedure_occurrence
  - visit_occurrence

- 2nd shift
  - drug_exposure
  - measurement
  - observation
  - procedure_occurrence
  - visit_occurrence

- 3rd shift
  - device_exposure
  - drug_exposure
  - measurement
  - observation
  - procedure_occurrence
  - visit_occurrence

\[ i \text{th position skips the same table name, and so on...} \]