Bahmni-Mart

Featured Pages

Content by label

There is no content with the specified labels

Recently Updated

- Frequently(F) Asked(A) Questions(Qs)
  Sep 04, 2020 • contributed by Rakesh Kumar
- Add Custom View Sql to bahmni-mart.json
  Sep 01, 2020 • contributed by Himabindu Thungathurty
- Overview
  Aug 19, 2020 • contributed by Himabindu Thungathurty
- Registration Second Page
  Jun 30, 2020 • contributed by Himabindu Thungathurty
- Conditions
  Jun 30, 2020 • contributed by Sree Kalyan Bapatla (Unlicensed)

Overview

- Addition of Custom Jobs/Views :

Bahmni would like to leverage their patient data to improve efficiency, reduce unnecessary variation and waste, and identify and address gaps in quality of care. One of the main reasons today reporting and data analysis is not as effective as it could be is not being able to extract data from Bahmni and use it with the right tools. Most of the time today is spent managing and creating data (paper based excel reports are collected at various project centres) and constructing data sets in custom softwares like excel rather than generating ad-hoc reports on the fly and analysing the data.

One of the steps in the direction of easing reporting is to design how we store the input data generated from the hospitals and missions. For this we came up with the design of an analytics database where hierarchical database is flattened and pivoted. This database is called the Mart DB or simply the analytics DB. This piece of solution would

- Make it easier for various Data Analytics tools to directly consume Bahmni Analytics database
- Would make it possible for Implementers to extract data / modify existing datasets without totally understanding the openmrs data model.
- Enable Developers to define new reports / extract datasets of interests faster.

In essence the pivoted and flattened structure will remove the hierarchical nature of the data.

A few views have also been created using the flattened tables of the analytical DB based on the needs of the users / statisticians which provides data from multiple tables by running simple sql queries

Jobs Types & Grouped tables:-

To simplify the amount of configuration that we are providing to the bahmni-mart application, we have packaged together similar tables under the respective types. Now the user just need to add a single job of the packaged type and can get the all the tables under that type

Below are the packaged job types and the jobs under that type

Job Type	**Grouped tables**
Programs	Programs_default, program_outcomes_default,patient_program_data_default,program_workflow_default, program_workflow_states_default, program_attributes
Patients	Patient_state_default, patient_allergy_status_default, patient_identifier
appointments	Patient_appointment_default, appointment_service_default, service_availability_default, appointment_speciality_default
bedManagement	Bed_patient_assignment_default, bed_tags_default, current_bed_details_default
location	Location_default, location_tag_map_default, location_attribute_details_default
operationTheater	Surgical_block_default, surgical_appointment_default, surgical_appointment_attribute_type_details_default, surgical_appointment_attributes
person	Person_details_default, person_address_default, person_attribute_info_default, address_hierarchy_level_default, person_attributes
provider	Provider_default, provider_attribute_details_default, provider_attributes
visitsAndEncounters	Patient_visit_details_default,patient_encounter_details_default,visit_attribute_details_default, visit_attributes
medicationAndOrders	Medication_data_default, [All forms under All Orderables as tables]
diagnosesAndConditions	Conditions_default, [All forms under Visit Diagnoses as tables]
bacteriology	[All forms under Bacteriology Concept Set as tables]
metadata	All concept related information class, datatype, with respect to source.
obs	[All forms under All Observation Templates as tables]
Forms 2.0	[All forms which are created with Implementer Interface]
reg	reg_{table Name}
disposition	[All forms under Disposition Set as tables]

Link to the example grouped job configuration

```
{
      "name": "Person",
      "type": "person",
      "chunkSizeToRead": "500",
      "groupedJobConfigs": [
        {
          "tableName": "person_attributes",
          "columnsToIgnore": [
            "primaryContact",
            "secondaryContact",
            "primaryRelative",
            "familyNameLocal",
            "givenNameLocal",
            "middleNameLocal"
          ]
        },
        {
          "tableName": "person_details_default",
          "columnsToIgnore": [
            "prefix",
            "given_name",
            "middle_name",
            "family_name_prefix",
            "family_name",
            "family_name2",
            "family_name_suffix"
          ]
        }
      ]
    }
```

Addition of Custom Jobs/Views :

Bahmni-mart enables the user to add custom jobs & views in addition to the default jobs provided. Below are the respective example configurations.

```
Custom Sql Job Configuration:
{
 "name": "Patient Info",
 "type": "customSql",
 "table_name": “patient_info",
 "readerSql": "SELECT * FROM patient",
 "incrementalUpdateConfig": {
   "updateOn": "patient_id",
   "eventCategory": "Patient",
   "openmrsTableName": "patient"
 }
 }
```

Note : Reader sql should be executable on openmrs database. Incremental configuration is optional and can be given only when you need an incremental update for this job.

Custom View Configuration :

```
{
 "name": "Person Details View",
 "sql": "SELECT * FROM person_details_default pdd JOIN
         person_address_default pad ON pad.person_id = pdd.person_id "
}
```

Note : sql should be executable on analytics database since we are creating view in analytics database

Difference between standard Grouped Jobs, Custom Jobs and Views:

As part of **standard grouped jobs** we flatten the data from openmrs but we don’t get to see the readerSql for each job. The readerSql gets generated dynamically as part of the code. So adding new columns is not straight forward and it requires change at code level.

When it comes to **Custom jobs**, we can directly see the readerSql and it redirects output to the desired table in the analytics. Adding new columns to the analytics table is straightforward as its a change in the sql query.

- The only configuration possible for grouped and custom jobs so far is, `columnsToIgnore`. This will ignore the specified columns mentioned under `columnsToIgnore` json from openmrs database to analytics database.
- For ignoring columns, each column in a table have to be specified. We can’t specify at job level and it has to be under that particular table name for the job.
- We can select particular job to be incremental/full load irrespective of entire bahmni-mart job type.

eg: If the bahmni-mart is running on incremental load, we can still mention few jobs to run on full load.

For **Views**, we can see the sql query but the query runs on the analytics tables but not on the openmrs tables. As part of the views query, we usually combine one or more analytics tables and create a table(view) out of these tables. If there is a data change in the above analytics tables then the view gets updated automatically

- Views will not have the option to ignore columns, incremental/full load options. If we want few columns to be added/ignored that can be directly done in the specific view query.

Setting Up CronJob for bahmni-mart:

To sync up analytics db with openmrs db, we can run bahmni-mart on regular intervals. That can be done once in a day/week based on the requirement. For to run bahmni-mart command automatically we can configure the cron job which will trigger the bahmni-mart command at the specified time in the day.

- Sample cornjob for bahmni-mart

```
00 23 * * *  bahmni-mart
```

The above cronjob runs the Bahmni-mart command daily at 11 pm in the night. The preferable time would be running the mart is after working hours for an implementation. This will run the mart on incremental load so the entire job wouldn’t take much time.

```
yum remove postgresql -y
yum remove postgres\* -y
rm -rf /var/lib/pgsql
yum list installed | grep postgres
```

Remove bahmni-mart if installed before

```
yum remove bahmni-mart -y
```

Download bahmni-mart playbook

```
wget -O /tmp/bahmni-mart-playbook.zip https://github.com[redacted]
/bahmni-mart-playbook/archive/master.zip && unzip -o /tmp/bahmni-mart-
playbook.zip -d /tmp && sudo rm -rf /etc/bahmni-mart-playbook && sudo
mv /tmp/bahmni-mart-playbook-master /etc/bahmni-mart-playbook && rm -rf
/tmp/bahmni-mart-playbook.zip
```

Update bahmni-mart inventory file as below

> change the bahmni-mart url based on the latest artefact

```
cd /etc/bahmni-mart-playbook/inventories/
mv bahmni-mart bahmni-mart_bkp
wget https://s3.ap-south-1.amazonaws.com/[redacted]
artifacts/mart_inventory
mv mart_inventory bahmni-mart
```

Update ip for fresh installation of mart

```
vi /etc/bahmni-mart-playbook/inventories/bahmni-mart
# delete the line with content '<master_ip> ansible_connection=local'
# add below line as first line
<master_ip> ansible_connection=ssh ansible_ssh_user=root
```

For standalone instance use below

```
localhost ansible_connection=local

[bahmni-emr-db]
localhost

[bahmni-mart]
localhost

[bahmni-mart-db]
localhost

[bahmni-mart-db-slave]

[bahmni-mart-scdf]

[metabase]
localhost

[metabase-db]
localhost

[metabase-db-slave]


[local:children]
bahmni-mart
bahmni-mart-db
bahmni-mart-db-slave
bahmni-mart-scdf
bahmni-emr-db
```

```
metabase
metabase-db
metabase-db-slave
```

1. Since the master-slave set-up is not available, remove all the lines with <slave_ip>
2. Update master_ip

Remove entries under [bahmni-mart-db-slave], [bahmni-mart-scdf], [metabase-db-slave] in the bahmni-mart inventory file

```
vi /etc/bahmni-mart-playbook/inventories/bahmni-mart
```

Verify and update the below parameters in setup.yml

`cd /etc/bahmni-mart-playbook`

```
bahmni_mart_version: "2.0.3-1"
openmrs_db_password: P@ssw0rd
metabase_db_password: password
analytics_db_password: password
postgres_password: password
metabase_with_ssl: false
custom_keystore_location: "<ssl certificate in jks format>"
metabase_keystore_password: <password of jks cert>
mail_subject: "Notification regarding failed jobs"
mail_from: "no-reply@bahmni-mart.notifications"
mail_recipients: "<mail recipients separated by comma>"
analytics_db_user: analytics
metabase_db_user: metabase
analytics_db_name: analytics
metabase_db_name: metabase
```

If user wants to install mart with latest bahmni_mart_url then user need to update bahmni_mart_version to the rpm version so that it will install the latest mart.

Modify the file “/etc/bahmni-mart-playbook/roles/postgres/defaults/main.yml” with below entries

```
postgres92_repo_rpm_name: pgdg-centos92-9.2-7.noarch.rpm
postgres96_repo_rpm_name: pgdg-redhat-repo-42.0-11.noarch.rpm
postgres92_repo_download_url: http://yum.postgresql.org/9.2/redhat/rhel-
6-x86_64/{{postgres92_repo_rpm_name}}
postgres96_repo_download_url: https://yum.postgresql.org/9.6/redhat
/rhel-6-x86_64/{{postgres96_repo_rpm_name}}
```

Metabase without ssl

```
ansible-playbook -i /etc/bahmni-mart-playbook/inventories/bahmni-mart
/etc/bahmni-mart-playbook/all.yml --extra-vars '@/etc/bahmni-mart-
playbook/setup.yml' --skip-tags "custom_ssl,lets_encrypt_ssl"
```

Metabase with let's encrypt ssl(optional)

```
ansible-playbook -i /etc/bahmni-mart-playbook/inventories/bahmni-mart
/etc/bahmni-mart-playbook/all.yml --extra-vars '@/etc/bahmni-mart-
playbook/setup.yml' --skip-tags "without_ssl,custom_ssl"
```

Metabase with custom ssl(optional)

```
ansible-playbook -i /etc/bahmni-mart-playbook/inventories/bahmni-mart
/etc/bahmni-mart-playbook/all.yml --extra-vars '@/etc/bahmni-mart-
playbook/setup.yml' --skip-tags "without_ssl,lets_encrypt_ssl"
```

Add necessary jobs or Remove the extra jobs from bahmni-mart.json

```
vi /var/www/bahmni_config/bahmni-mart/bahmni-mart.json
```

Run bahmni mart

```
bahmni-mart
# check the logs
tail -100f /var/log/bahmni-mart/bahmni-mart.log
```

Check if following URLs are accessible

Metabase

```
docker ps
docker stop <METABASE CONTAINER ID>
docker start metabase
http://<host>:9003/
```

Bahmni-Mart Setup with MySQLWorkbench & DBeaver

Prerequisites:

1. Generate public key using

```
ssh-keygen -t rsa
```

2. For latest DBeaver, Run this on your key to convert it to RSA private key.

```
ssh-keygen -p -m PEM -f ~/.ssh/id_rsa
```

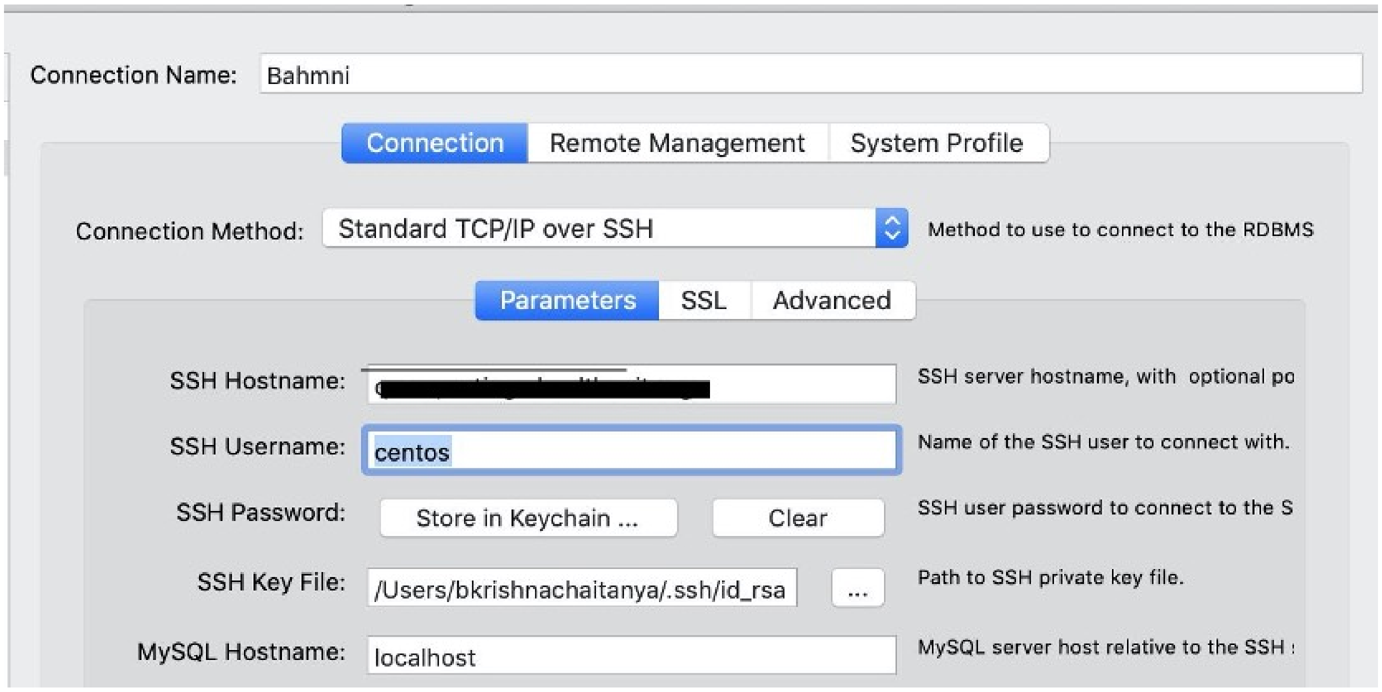
MYSQL:

Tools:

```
1.MySQLWorkbench
2.DBeaver
```

Connecting using MySQLWorkbench:

```
Change the connection method to Standard TCP/IP over SSH, by default,
connection method will be set to Standard (TCP/IP)
Hostname: “[redacted]
Username: “centos”
SSH key file path: “provide the path to id_rsa file”
MySQL Hostname: “localhost”
MySQL Server Port: “3306”
Username: “root”
```


Connection Name: Bahmni

Connection Method: Standard TCP/IP over SSH — Method to use to connect to the RDBMS

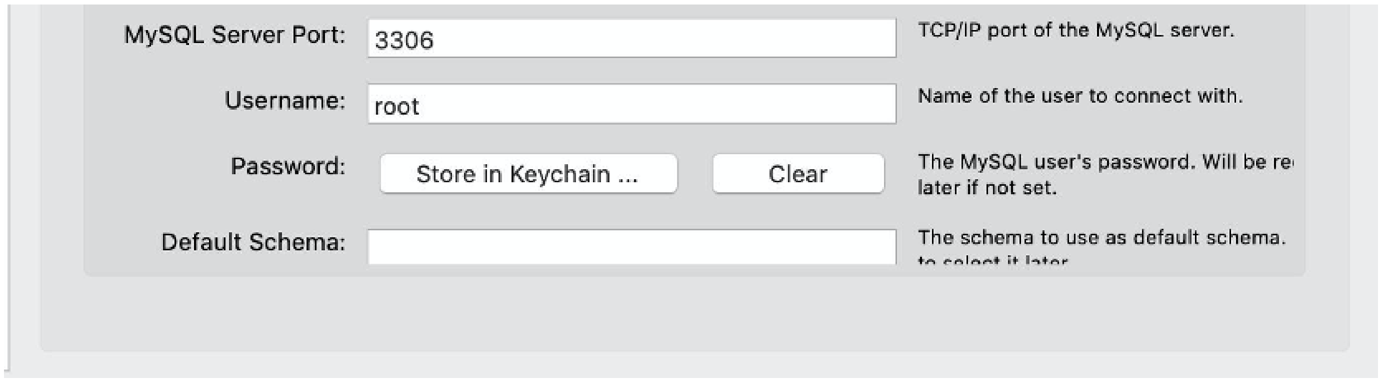
SSH Hostname: [redacted] — SSH server hostname, with optional po

SSH Username: centos — Name of the SSH user to connect with.

SSH Password: Store in Keychain ... Clear — SSH user password to connect to the S

SSH Key File: /Users/bkrishnachaitanya/.ssh/id_rsa ... — Path to SSH private key file.

MySQL Hostname: localhost — MySQL server host relative to the SSH :

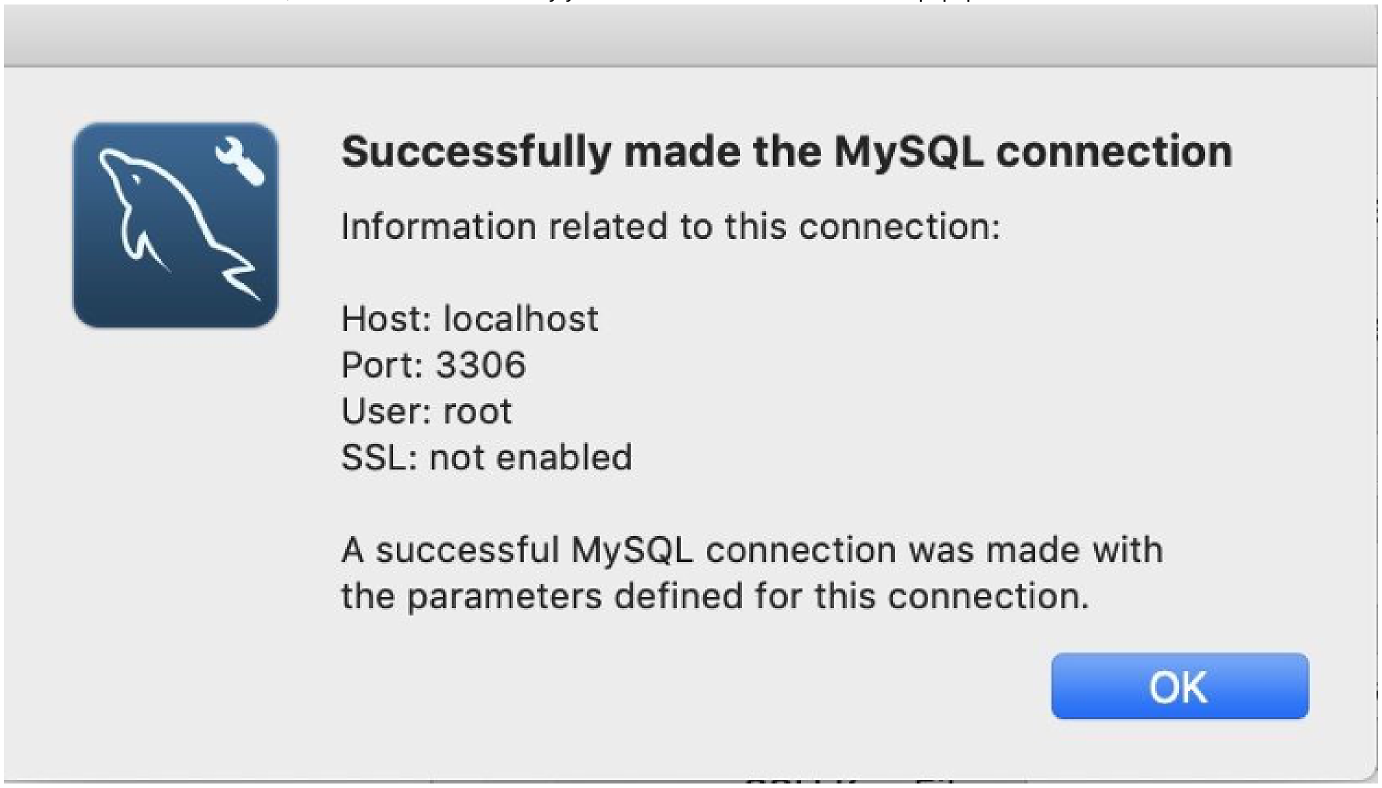

MySQL Server Port: 3306 TCP/IP port of the MySQL server.

Username: root Name of the user to connect with.

Password: Store in Keychain ... Clear The MySQL user's password. Will be re later if not set.

Default Schema: The schema to use as default schema. to select it later

Once all the details are added, Test Connection and ideally you see the “Connection Successful” popup.


Successfully made the MySQL connection

Information related to this connection:

Host: localhost
Port: 3306
User: root
SSL: not enabled

A successful MySQL connection was made with the parameters defined for this connection.

POSTGRESS:

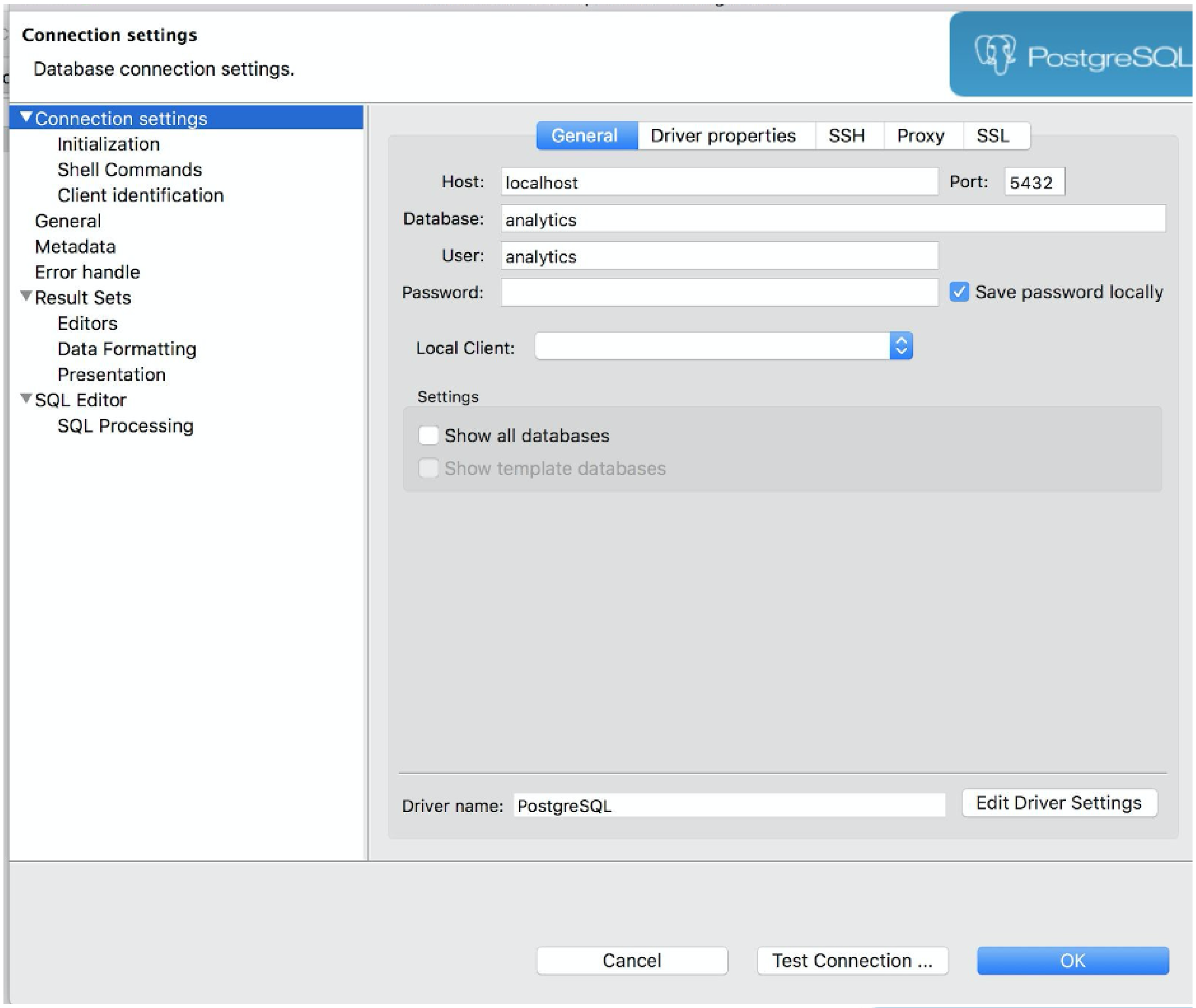
Tools:

DBeaver

CONNECTING TO POSTGRESS USING DBEAVER:

```
Select postgress DB while creating a new connection
Select SSH tab and check “Use SSH Tunnel” option
Host/IP: [redacted]
Port: 22
User Name: “centos”
Authentication Method: “Public Key”
Private Key: “provide the path to id_rsa file”
```

Connection "mart-qa-tbemr" configuration


Connection settings

Database connection settings.

- Connection settings
  - Initialization
  - Shell Commands
  - Client identification
- General
- Metadata
- Error handle
- Result Sets
  - Editors
  - Data Formatting
  - Presentation
- SQL Editor
  - SQL Processing

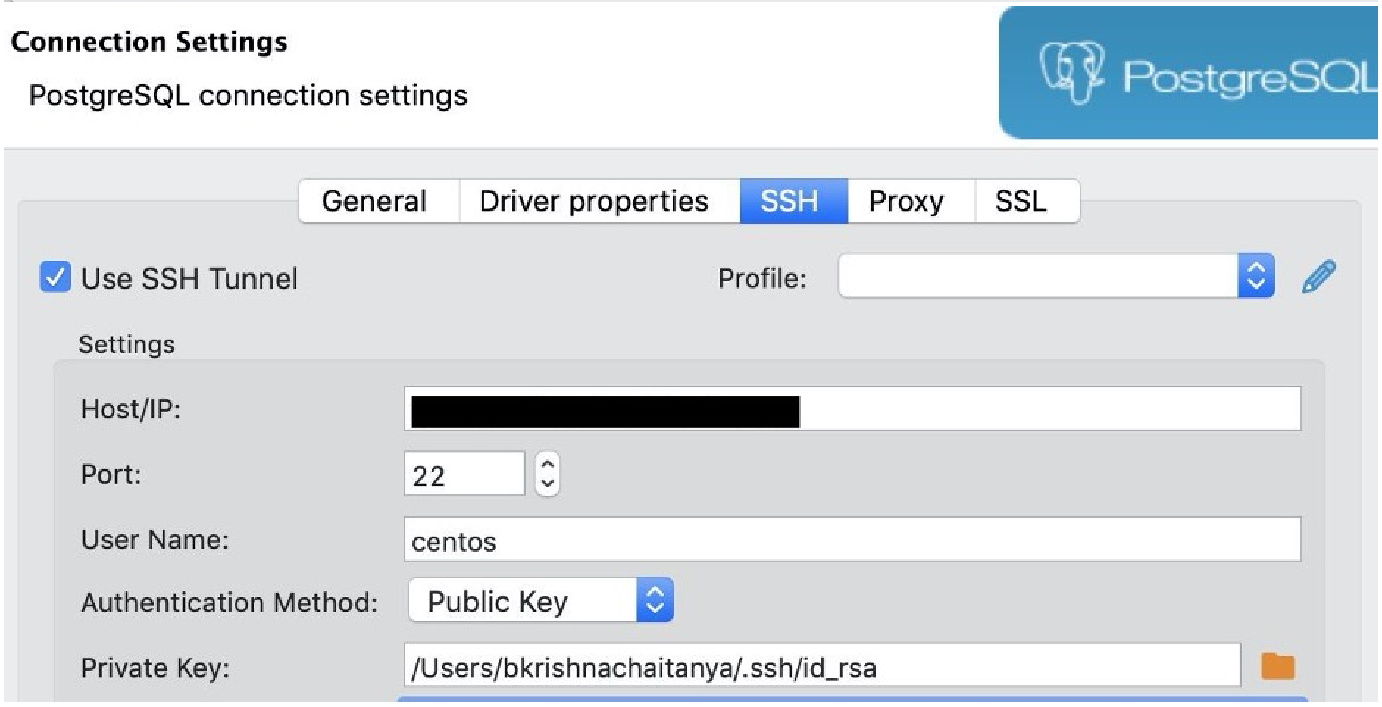
General | Driver properties | SSH | Proxy | SSL

Host: localhost Port: 5432
Database: analytics
User: analytics
Password: [✓] Save password locally
Local Client:

Settings
[ ] Show all databases
[ ] Show template databases

Driver name: PostgreSQL Edit Driver Settings

Cancel | Test Connection ... | OK

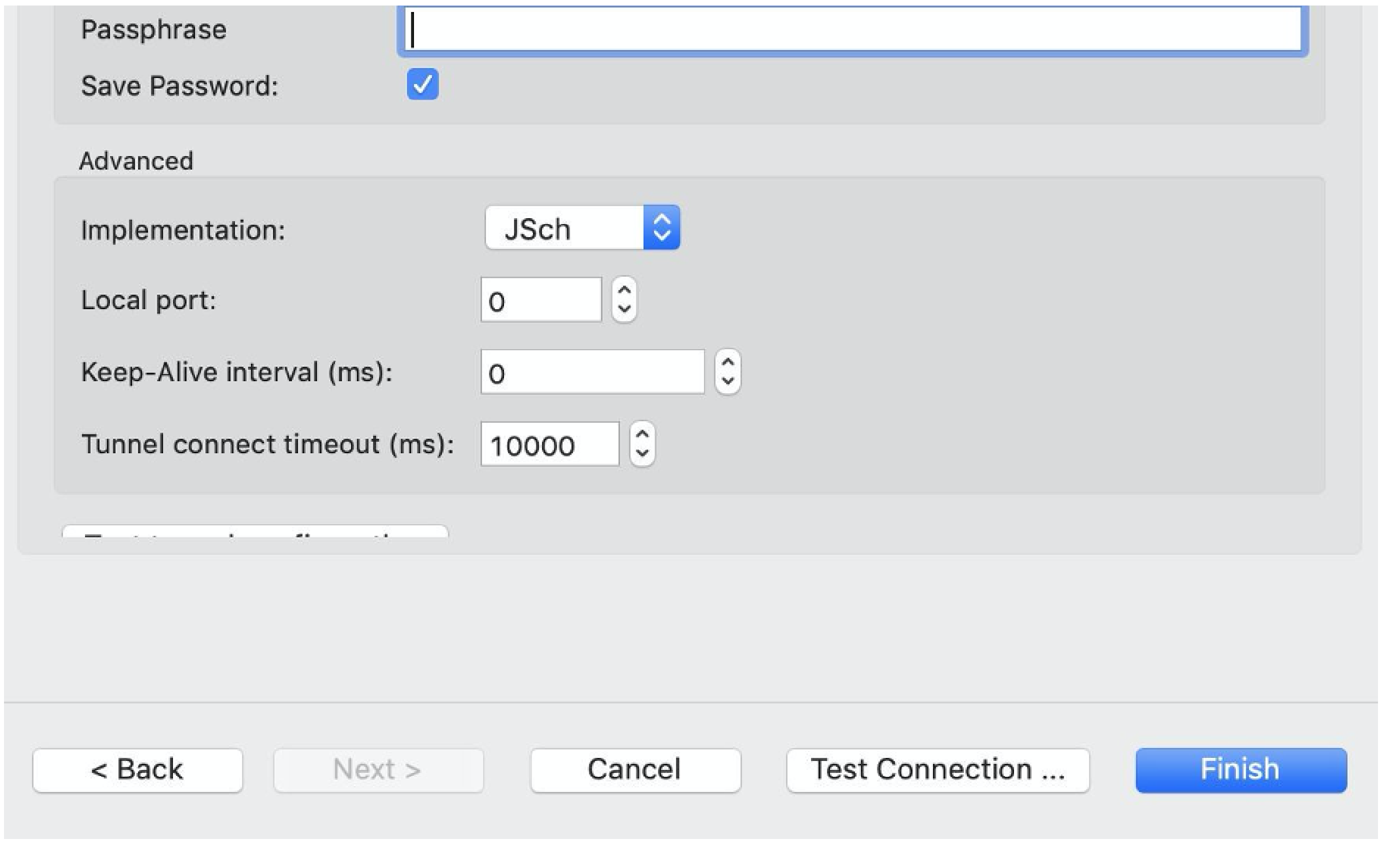

Connection Settings

PostgreSQL connection settings

General | Driver properties | SSH | Proxy | SSL

[✓] Use SSH Tunnel Profile:

Settings
Host/IP: [redacted]
Port: 22
User Name: centos
Authentication Method: Public Key
Private Key: /Users/bkrishnachaitanya/.ssh/id_rsa


Passphrase
Save Password: ✓

Advanced

Implementation: JSch
Local port: 0
Keep-Alive interval (ms): 0
Tunnel connect timeout (ms): 10000

Once all the details are added, Test Connection and ideally you see the “Connection Successful” popup.

Bahmni-Mart Json File

This is the default bahmni_mart json file, where we could find reference for all the jobs mentioned below

```
programs
patients
Appointments
Bed Management
Location
Operation Theater
Person
Provider
Visits And Encounters
Medication And Orders
Diagnoses And Conditions
Bacteriology Data
MetaData Dictionary
Obs Data
Form2 Obs Data
Registration Second Page
Disposition Data
```

```
{
    "jobs": [
      {
        "name": "Programs",
        "type": "programs",
        "chunkSizeToRead": "500"
      },
      {
        "name": "Patients",
        "type": "patients",
        "chunkSizeToRead": "500"
      },
      {
        "name": "Appointments",
        "type": "appointments",
        "chunkSizeToRead": "500",
        "groupedJobConfigs": [
          {
            "tableName": "appointment_service_default",
            "columnsToIgnore": [
            ]
          }
        ]
      },
      {
        "name": "Bed Management",
        "type": "bedManagement",
        "chunkSizeToRead": "500"
      },
      {
        "name": "Location",
        "type": "location",
        "chunkSizeToRead": "500"
      },
      {
        "name": "Operation Theater",
        "type": "operationTheater",
        "chunkSizeToRead": "500"
      },
      {
        "name": "Person",
        "type": "person",
        "chunkSizeToRead": "500",
        "groupedJobConfigs": [
          {
            "tableName": "person_attributes",
            "columnsToIgnore": [
              "primaryContact",
              "secondaryContact",
              "primaryRelative",
              "familyNameLocal",
            "givenNameLocal",
            "middleNameLocal"
          ]
        },
        {
          "tableName": "person_details_default",
          "columnsToIgnore": [
            "prefix",
            "given_name",
            "middle_name",
            "family_name_prefix",
            "family_name",
            "family_name2",
            "family_name_suffix"
          ]
        }
      ]
    },
    {
      "name": "Provider",
      "type": "provider",
      "chunkSizeToRead": "500"
    },
    {
      "name": "Visits And Encounters",
      "type": "visitsAndEncounters",
      "chunkSizeToRead": "500"
    },
    {
      "name": "Medication And Orders",
      "type": "medicationAndOrders",
      "chunkSizeToRead": "500",
      "groupedJobConfigs": [
        {
          "tableName": "medication_data_default",
          "columnsToIgnore": [
            "instructions",
            "stop_notes"
          ]
        }
      ]
    },
    {
      "name": "Diagnoses And Conditions",
      "type": "diagnosesAndConditions",
      "chunkSizeToRead": "500"
    },
    {
      "name": "Bacteriology Data",
      "conceptReferenceSource": "",
        "type": "bacteriology"
      },
      {
        "name": "MetaData Dictionary",
        "type": "metadata",
        "conceptReferenceSource": ""
      },
      {
        "name": "Obs Data",
        "type": "obs",
        "incrementalUpdateConfig": {
          "updateOn": "encounter_id",
          "eventCategory": "Encounter",
          "openmrsTableName": "encounter"
        },
        "separateTableConfig": {
          "enableForAddMoreAndMultiSelect": true,
          "separateTables": [
          ]
        },
        "conceptReferenceSource": "",
        "ignoreAllFreeTextConcepts": true,
        "columnsToIgnore": [
          "Image",
          "Video"
        ]
      },
      {
        "name": "Form2 Obs Data",
        "type": "form2obs",
        "incrementalUpdateConfig": {
          "updateOn": "encounter_id",
          "eventCategory": "Encounter",
          "openmrsTableName": "encounter"
        },
        "separateTableConfig": {
          "enableForAddMoreAndMultiSelect": true,
          "separateTables": [
          ]
        },
        "conceptReferenceSource": "",
        "ignoreAllFreeTextConcepts": true,
        "columnsToIgnore": [
          "Image"
        ]
      },
      {
        "name": "Registration Second Page",
        "type": "reg",
        "columnsToIgnore": [],
        "separateTableConfig": {
          "enableForAddMoreAndMultiSelect": true,
          "separateTables": []
        },
        "incrementalUpdateConfig": {
          "updateOn": "encounter_id",
          "eventCategory": "Encounter",
          "openmrsTableName": "encounter"
        }
      },
      {
        "name": "Disposition Data",
        "type": "disposition",
        "columnsToIgnore": [],
        "incrementalUpdateConfig": {
          "updateOn": "encounter_id",
          "eventCategory": "Encounter",
          "openmrsTableName": "encounter"
        }
      }
    ],
    "procedures": [
      {
        "name": "Discharge Date Procedure",
        "sourceFilePath": "classpath:procedureSql/dischargeDateProc.sql"
      },
      {
        "name": "Age Group Procedure",
        "sourceFilePath": "classpath:procedureSql/ageGroupProc.sql"
      }
    ],
    "views": [
      {
        "name": "patient_program_view",
        "sourceFilePath": "classpath:viewSql/patientProgramView.sql"
      },
      {
        "name": "patient_program_state_view",
        "sourceFilePath": "classpath:viewSql/patientProgramStateView.sql"
      },
      {
        "name": "patient_visits_encounters_view",
        "sourceFilePath": "classpath:viewSql/patientVisitsEncountersView.
sql"
      },
      {
        "name": "appointment_admin_panel_view",
        "sql": "SELECT * FROM appointment_service_default LEFT OUTER JOIN
service_availability_default USING (appointment_service_id,
service_name)"
```

```
    },
    {
      "name": "patient_details_view",
      "sourceFilePath": "classpath:viewSql/patientDetailsView.sql"
    },
    {
      "name": "patient_information_view",
      "sourceFilePath": "classpath:viewSql/patientInformationView.sql"
    },
    {
      "name": "bed_management_view",
      "sourceFilePath": "classpath:viewSql/bedManagementView.sql"
    },
    {
      "name": "bed_management_locations_view",
      "sourceFilePath": "classpath:viewSql/locationWiseDischarge.sql"
    },
    {
      "name": "patient_bed_view",
      "sourceFilePath": "classpath:viewSql/patientBedView.sql"
    },
    {
      "name": "patient_operation_theater_view",
      "sourceFilePath": "classpath:viewSql/patientOperationTheaterView.
sql"
    },
    {
      "name": "patient_appointment_view",
      "sourceFilePath": "classpath:viewSql/patientAppointmentView.sql"
    },
    {
      "name": "patient_program_medication_view",
      "sourceFilePath": "classpath:viewSql/patientProgramMedicationView.
sql"
    },
    {
      "name": "patient_diagnosis_condition_view",
      "sourceFilePath": "classpath:viewSql
/patientDiagnosisConditionView.sql"
    },
    {
      "name": "patient_bed_tags_history_view",
      "sourceFilePath": "classpath:viewSql/patientBedTagView.sql"
    }
  ]
}
```

Add Custom View Sql to bahmni-mart.json

- 1. Add Direct Query:

As we know view is a virtual table based on the result-set of an SQL statement. All the existing views that we have in bahmni-mart.json are bundled with bahmni-mart.rpm file. But if we want to add a new view query to the bahmni-mart.json file, we have 2 ways to implement.

1. Add direct query
2. Give path of the query

1. ADD DIRECT QUERY:

If we have simple view query to get executed on analytics database then we can directly add it in the bahmni-mart.json with the table name as show below

```
{
 "name":
"patient_program_medication_view_test",
 "sql": "SELECT pd.person_id AS patient_id,
ppd.program_id, md.patient_program_name AS
program_name, ppd.date_enrolled, ppd.
date_completed, ppd.program_outcome, pd.
gender, pd.birthyear            AS birth_year,
EXTRACT(YEAR FROM (SELECT age( md.start_date,
TO_DATE(CONCAT('01-01-', pd.birthyear), 'dd-MM-
yyyy')))) AS age_at_medication, age_group(md.
start_date, TO_DATE(CONCAT('01-01-', pd.
birthyear), 'dd-MM-yyyy')) AS
age_group_at_medication, pd.dead, pa.*, md.
patient_program_id, md.encounter_id, md.
encounter_type_name, md.order_id, md.
orderer_name, md.coded_drug_name, md.
non_coded_drug_name, md.dose, md.dose_units,
md.frequency, md.route, md.
start_date           AS medication_start_date,
md.calculated_end_date  AS
medication_calculated_end_date, md.
date_stopped         AS
medication_stopped_date, md.stop_reason, md.
duration, md.duration_units, md.quantity, md.
quantity_units, md.dispense             AS
is_dispensed, md.visit_id, md.visit_type FROM
person_details_default pd LEFT JOIN
person_attributes pa ON pa.person_id = pd.
person_id LEFT JOIN medication_data_default md
ON md.patient_id = pd.person_id LEFT OUTER
JOIN patient_program_data_default ppd ON ppd.
patient_id = md.patient_id and ppd.
patient_program_id = md.patient_program_id"
}
```

Once we add the above configuration in the mart file and run the bahmni-mart command, we will be able to see the corresponding table in the analytics database.

2. GIVE PATH OF THE QUERY

If we have sql queries which is very big and can't be included in the bahmni-mart.json file then it requires creating new sql file for it. Once we create/add sql file to bahmni instance in a specific path, we have to update the absolute path of sql query in the bahmni-mart.json. After that running the bahmni-mart will create the corresponding table in the analytics database.

```
{
 "name":
"patient_program_medication_view_test",
 "sourceFilePath":"file:/home/bahmni/viewSql
/patientProgramMedicationViewTest.sql"
}
```

In the above sample comfiguration, we have the sql file added in `/home/bahmni/viewSql` directory.

POINTS TO NOTE:

- This query must be a view query which will run on analytics tables.
- We can't use these sql queries to run on openmrs tables to create the table in analytics database. For that we must add sql query to bahmni-mart and it will be bundled with the mart rpm.

Modules

Module Name	Description

Programs	`Configuration`
	`Existing OpenMRS tables`
	`Flattened Mart Tables`
	`Mart Views`

Patients	`Configuration`
	`Existing OpenMRS table`
	`Mart Tables`
	`Mart Views`

Appointment Scheduling	`Configuration`
	`Existing OpenMRS tables`
	`Flattened Mart Tables`
	`Mart Views`

Bed Management	

	Configuration Existing OpenMRS tables Flattened Mart Tables Mart Views
Location	Configuration Existing OpenMRS tables Flattened Mart Tables
Operation Theatre	Configuration Existing OpenMRS tables Flattened Mart Tables
Person	Configuration Existing OpenMRS tables Flattened Mart Tables
Provider	Configuration Existing OpenMRS tables Flattened Mart Tables
Visits and Encounters	Configuration Existing OpenMRS tables Flattened Mart Tables Mart Views
Medication and orders	Configuration Existing OpenMRS tables Flattened Mart Tables

Diagnosis and condition	`Configuration`
	`Existing OpenMRS tables`
	`Flattened Mart Tables`
	`Mart Views`

	`Existing OpenMRS tables:`
	`Flattened Mart Tables:`
	`Usable Configurations:`

Programs Module

This module is similar to the Clinical module present in the product.

- Configuration
  - incrementalUpdateConfig is applicable (Refer Incremental Update)
  - ColumnsToIgnore apply for this module (Refer Appendix)
- Existing OpenMRS tables
- Flattened Mart Tables
- Mart Views

CONFIGURATION

incrementalUpdateConfig is applicable (Refer Incremental Update)

ColumnsToIgnore apply for this module (Refer Appendix)

```
"jobs": [
{
      "name": "Programs",
      "type": "programs",
      "chunkSizeToRead": "500"
      "groupedJobConfigs": [
        {
          "tableName": "program_outcomes_default",
          "columnsToIgnore": [
          ]
        }
      ]
}
```

EXISTING OPENMRS TABLES

```
Program
Concept
```

```
Concept_set
Concept_name
Concept_view
Patient_program
Episode_patient_program
Location
Patient
Patient_state
Program_workflow_state
Program_workflow
```

FLATTENED MART TABLES

```
programs_default
program_outcomes_default
patient_program_data_default
program_workflow_default
program_workflow_states_default
program_attributes
```

a) **programs_default** - This table gives all the programs in an implementation

Openmrs tables used: **program**

Column Name	Description
program_id	Program id as in Openmrs
program_name	Name of the Program
program_description	Description of the Program
creator_id	Creator ID of the Program
creator_name	Name of the Creator
date_created	Date on which the Program was created
date_changed	If any modifications to the Program is done, the date the changes were done is captured here
changed_by_id	ID by whom the change was done
changed_by_name	Name by whom the change was done

b) program_outcomes_default - This table lists all the outcomes for a program. Each outcome of a program will be a separate row
Openmrs tables used: program, concept, concept_set, concept_name

Column Name	Description
program_id	Program id as in Openmrs
program_name	Name of the Program
program_outcome	Outcome of the program

c) patient_program_data_default - This table gives the data about when the patient is enrolled into a program, completion date of the program and other information like location at which the patient enrolled into a program etc.
Openmrs tables used: patient_program, episode_patient_program, concept_name, location

Column Name	Description
patient_id	Database id of patient
program_id	Program id as in Openmrs
patient_program_id	Unique id generated when a patient is enrolled in to a program from Openmrs table patient_program
date_enrolled	Date enrolled in to the program
age_during_enrollment	Age of the patient during program enrollment (Date difference of birth date and program enrollment date)
date_completed	Date when the patient completed the program
age_during_completion	Age of the patient during program completion (Date difference of birth date and program completion date)
location_id	Id of the location where the patient is enrolled to a program
location_name	Name of the location where the patient is enrolled to a program
program_outcome	Outcome of the program for patient
creator_id	Id of the user who enrolled the patient in to program
creator _name	Name of the user who enrolled the patient in to program
date_created	Date on which the entry was made
date_changed	If any modifications to the patient program enrolment is done, the date the changes were done
changed_by_id	ID by whom the change was done
changed_by_name	Name by whom the change was done
voided	If patient information is deleted, voided shows true. (From Openmrs table patient_program)

d) program_workflow_default - This table lists all the workflows in a program. This is a metadata table that does not capture any user entered information. Every program can have a workflow that determines all the states a patient can transition to in that program.

Openmrs tables used: program_workflow, concept_view

Column Name	Description
program_id	Program id as in Openmrs
program_workflow_id	Id of the workflow
program_workflow_name	Name of the workflow

f) program_workfow_states_default - This table lists all the states in workflows in a program. This is a metadata table that does not capture any user entered information. Every program can have a workflow that determines all the states a patient can transition to in that program.

Openmrs tables used: program_workflow_state, program_workflow, users, concept_view

Column Name	Description
program_workflow_state_id	Unique id generated for a program workflow state
program_workflow_id	Unique id generated for program workflow
state_name	Name of the state
program_workflow_name	Name of the program workflow
initial	True if the state is initial state else False
terminal	True if the state is terminal state else False
creator_id	Id of the creator of the state

creator_name	Name of the creator of the state
date_created	Date on which the state was created
date_changed	Date on which if any changes are made
changed_by_id	Id of the user who made changes
changed_by_name	Name of the user who made changes

g) **program_attributes -**This table gives all the program attributes enrolled for a particular patient.

Openmrs tables used: program_attribute_type, patient_program_attribute

Mart Table:- program_attributes

Column Name	Description
patient_program_id	Unique id generated in DB when a patient is enrolled to a program
program_attribute 1	
program_attribute 2	

MART VIEWS

a) **patient_program_view**

This view gives all the person details along with the program details that patient has been enrolled to. If a patient is enrolled into 2 programs (or twice in to the same program), there will be 2 rows corresponding to that patient id. The data related to programs will be different in two rows where as the person details data will be duplicated. This gives person details like gender, birthdate, person address, person attributes captured in registration page and program data like date enrolled in to the program, date of completion and program outcome. Patient details also includes (Patient’s age during program enrollment) and age_group_at_progage_at_programram (Patient’s age group during program enrollment).

Mart tables used: person_details, person_address, person_attributes, patient_program_data

b) **patient_program_state_view**

This view gives all the person details along with the program details that patient has been enrolled to. If a patient is enrolled into 2 programs (or twice in to the same program), there will be 2 rows with different patient_program_id corresponding to that patient id. The data related to programs will be different in two rows where as the person details data will be duplicated. This gives person details like gender, birthdate, person address, person attributes captured in registration page and program data like date enrolled in to the program, date of completion and program outcome. Patient details also includes (Patient’s age during program enrollment) and age_group_at_progage_at_programram (Patient’s age group during program enrollment).

Mart tables used: person_details, person_address, person_attributes, patient_program_data

Note: **patient_program_view** doesn’t have state transition rows where as **patient_program_state_view** have it.

c) **patient_program_info_view**

This view gives all the programs a patient is enrolled in to and all the state transitions that a patient has undergone as part of the program. Every patient state that details state name, start date of the state and end date of the state will be a different row and all the program related information like program name, date enrolled in to program will be duplicated.

Mart tables used: programs, patient_program_data, patient_state

---

Patients

- Configuration
  - incrementalUpdateConfig is applicable (Refer Incremental Update)
  - ColumnsToIgnore apply for this module (Refer Appendix)
- Existing OpenMRS tables
- Flattened Mart Tables
- Mart Views

CONFIGURATION

incrementalUpdateConfig is applicable (Refer Incremental Update)

ColumnsToIgnore apply for this module (Refer Appendix)

```
{
      "name": "Patients",
      "type": "patients",
      "chunkSizeToRead": "500"

}
```

EXISTING OPENMRS TABLES

```
patient
patient_program
patient_state
person
person_name
patient_identifier
users
program
program_workflow_state
concept_view
```

FLATTENED MART TABLES

```
patient_state_default
patient_allergy_status_default
patient_identifier
```

a) patient_state_default - This table gives details about patient state transitions in a program.

Openmrs tables used: patient, patient_program, program, patient_state, program_workflow_state, concept_view

Column Name	Description
patient_state_id	Unique column id from patient_state table from openmrs
patient_program_id	Unique id generated in DB when a patient is enrolled to a program
patient_id	Patient id from openmrs DB
program_id	Program ID from the program table
program_name	Name of the program to which the patient is enrolled
state	Program state id
state_name	Name of the program state

start_date	Start date of the patient state
end_date	End date of the patient state
creator_id	Id of the creator who started the patient sate
creator_name	Name of the creator who started the patient state
date_created	Date on the which the patient moved to this state
date_changed	Date on which any changes to the state was made
changed_by_id	Id of the user who made changes
changed_by_name	Name of the user who made changes

b) **patient_allergy_status_default -** This table displays allergy status information of each patient, present in the person table

Openmrs tables used: patient

Mart Table :- **patient_allergy_status**

Column Name	**Description**
patient_id	Patient id from openmrs DB
allergy_status	Allergy status of the patient, if the value is not provided the “Unknown” will be displayed by default.

c) **patient_identifier -** This is an EAV table. Columns are the pivoted values in identifier_type_table.

Openmrs tables used: patient, person, patient_identifier

Mart Table :- **patient_identifier**

Column Name	**Description**
person_attribute_type_id	Id of person attribute type
name	Name of the person attribute
description	Description of the person attribute

MART VIEWS

a) **patient_details_view** :

This view provides those patient details which help in identifying a patient in terms of Nationality, camp, and special needs like caretaker requirements, legal guardian, etc and attributes pertaining to those needs

Mart tables used: person_details, person_attributes

b) Patient_information_view:

This view provides all the patient attributes, barring the PII, as captured in the Registration module.

Mart tables used : patient_identifier, person_details, person_attributes, person_address

Appointment Scheduling

- Configuration
  - ColumnsToIgnore apply for this module (Refer Appendix)
  - incrementalUpdateConfig is applicable (Refer Incremental Update)
- Existing OpenMRS tables
- Flattened Mart Tables
  - Note : The below Mart tables are created by customSql job type.
- Mart views

CONFIGURATION

ColumnsToIgnore apply for this module (Refer Appendix)

incrementalUpdateConfig is applicable (Refer Incremental Update)

```
{
      "name": "Appointments",
      "type": "appointments",
      "chunkSizeToRead": "500",
      "groupedJobConfigs": [
        {
          "tableName": "appointment_service_default",
          "columnsToIgnore": [
          ]
        }
      ]
}
```

EXISTING OPENMRS TABLES

```
Patient_appontment
Appointment_service
appointment_speciality
```

FLATTENED MART TABLES

Note : The below Mart tables are created by customSql job type.

```
patient_appointment_default
appointment_service_default
appointment_speciality_default
service_availability_default
```

a) patient_appointment_default

Column Name	Description
patient_id	Patient identification number.
appointment_id	Id reference for an appointment
appointment_location	Location where the appointment is created
appointment_provider	Creator of the appointment
appointment_service	Service of the appointment
appointment_service_duration	Service duration of the appointment
appointment_service_type	Service type of the appointment

appointment_service_type_duration	Service type duration of the appointment
appointment_speciality	Speciality of the appointment
appointment_start_time	Start_time of the appointment
appointment_status	Status of the appointment
appointment_end_time	End time of the appointment
appointment_kind	Refers whether it is WalkIn or Scheduled appt.
comments	Comments given during appointment creation

b) appointment_service_default

Column Name	Description
appointment_service_id	Id of appointment service
location_name	Location of the appointment
service_description	Description of the service
service_duration	Duration of the service
service_starttime	Start time of the service
service_endtime	End time of the service
service_max_load	Max load of the service
service_name	Name of the service
service_type	Type of the service
service_type_duration	Duration of service type
speciality	Speciality of the service

c) appointment_speciality_default

Column Name	Description
speciality_id	Id of speciality
speciality	Appointment speciality

d) service_availability_default

Column Name	Description
appointment_service_id	Id of appointment_service
availability_start_time	Start time of the availability
availability_end_time	End time of the availability
availability_day_of_week	Day on which service is available
availability_max_load	Max load of the availability
service_location	Location of the service
service_name	Name of the service

MART VIEWS

a) patient_appointment_view

This view combines person_details_default, person_attributes and patient_appointment_default tables. Additionally, age_during_appointment (Date difference between patient_appointment_startdate and birthdate) and the corresponding age group.

b) appointment_admin_panel_view

This view combines appointment_service_default and service_availability_default

Bed Management

- Configuration
  - ColumnsToIgnore apply for this module (Refer Appendix)
  - incrementalUpdateConfig is applicable (Refer Incremental Update)
- Existing OpenMRS tables
- Flattened Mart Tables
- Mart Views

CONFIGURATION

ColumnsToIgnore apply for this module (Refer Appendix)

incrementalUpdateConfig is applicable (Refer Incremental Update)

```
{
      "name": "Bed Management",
      "type": "bedManagement",
      "chunkSizeToRead": "500"
}
```

EXISTING OPENMRS TABLES

```
Bed
Bed_tag_map
Bed_tag
Bed_patient_assignment_map
Bed_type
Bed_location_map
Location
Encounter
Visit
Visit_type
```

FLATTENED MART TABLES

Note : The below Mart tables are created by customSql job type.

```
bed_patient_assignment_default
bed_tags_default
current_bed_details_default
```

a) **bed_patient_assignment_default**

Column Name	Description
patient_id	Patient identification number.
bed_id	Id reference for a bed
bed_number	Number assigned by hospital for a bed for their identification.
date_started	Date time column that gives occupancy duration of patient on different encounters
date_stopped	Date time column that gives occupancy duration of patient on different encounters
location	location of the patient in bed management module
encounter_id	Encounter on which the bed was occupied by the patient
encounter_datetime	Time of encounter
visit_id	Id of the visit
visit_type	Type of visit of the patient

b) bed_tags_default

Column Name	Description
bed_tag_map_id	Id reference for bed tag map
bed_id	Id reference for a bed
bed_location	Location of bed in the hospital
bed_number	Number assigned by hospital for a bed for their identification
bed_status	Status of the bed (eg :Available, Occupied)
bed_tag_name	Tag for bed (eg : Lost, Isolation, Strict Isolation, Reserved for CT)
date_created	tag start date
date_changed	tag changed date
date_stopped	Date stopped is date voided

c) current_bed_details

Column Name	Description
bed_id	Id reference for a bed
bed_location	Location of bed in the hospital
bed_number	Number assigned by hospital for a bed for their identification.
bed_type	Type of bed

MART VIEWS

a) bed_management_view

This view provides the details of all the beds that a patient had been assigned to along with details like start and end date & time. It also provides information about the status of each bed when assigned to the patient and its location in the hospital. Since a patient may be assigned to different beds at different times there will be multiple rows for same patient.

Column Name	Description
patient_id	Patient identification number.

visit_id	Id reference for visit
bed_number	Number assigned by hospital for a bed for their identification.
date_started	Date time column that gives occupancy duration of patient on different encounters
date_stopped	Date time column that gives occupancy duration of patient on different encounters
location	location of the patient in bed management module
encounter_type_name	Encounter on which the bed was occupied by the patient
bed_tag_name	tag name assigned to the bed
tag_start_date	tag start date
tag_end_date	tag end date

b) patient_bed_tags_history_view

Column Name	Description
patient_id	Id reference for a patient
visit_id	Patient visit id
encounter_id	Id reference for an encounter
bed_number	Bed number
location	Location in hospital
age_at_bed_assignment	Age of patient
age_group_at_bed_assignment	Under which age range the patient belongs
action	action performed (Movement/Discharge etc..)
assigned_on	Assigned date
discharged_on	Patient discharge date
bed_tags	Bed tag, if tagged
bed_tag_created	Date bed was tagged
bed_tag_removed	Date bed tag was removed

c) patient_bed_view

Column Name	Description
age_at_bed_assignment	Age of the patient during bed assignment
bed_assigned_date	Date of bed assignment
bed_discharged_date	Date of discharge from bed
birth_year	Year of birth of the patient
dead	True (Patient is Dead) False(Patient is not Dead)
gender	Gender of the patient
location	Location of the bed
nationalIdentificationNumber	Patient attribute
patient_id	Id of the Patient

patientAddress	Patient attribute
patientAddressLine2	Patient attribute
patientCountry	Patient attribute
patientDistrict	Patient attribute
person_id	Person id of the patient
telephoneNumber	Patient Attribute
visit_id	Visit id of the patient

Location

CONFIGURATION

- configuration
    - incrementalUpdateConfig is applicable (Refer Incremental Update)
- Flattened Mart Tables

incrementalUpdateConfig is applicable (Refer Incremental Update)

```
{
      "name": "Location",
      "type": "location",
      "chunkSizeToRead": "500"
}
```

Existing OpenMRS tables

```
location
location_tag_name
location_tag_map
location_attribute
location_attribute_type
```

FLATTENED MART TABLES

```
location_default
location_tag_map_default
location_attribute_details_default
```

a) location_default - This table provides the location and all the details of locations.

Openmrs tables used: location

Column Name	Description
location_id	Unique column id from location table from openmrs
name	Name of the location

description	Description for that particular location
address1 - address15	Represents different fields in address (Configurable)
city_village	City/Village of the patient
state_province	State/Province of the patient
postal_code	Postal code of the patient’s residential area
country	Country of patient’s residence
county_district	District of the patient’s residence
latitude	Latitude of the area of residence
longitude	Longitude of the area of residence
start_date	Date when the patient started living in that area
end_date	Date when the patient stopped being in that area
parent_location	parent locations id

b) **location_tag_map_default** - This table provides the location tag map details

Openmrs Tables used :- location, location_tag_name, location_tag_map

Column Name	**Description**
location_id	Unique column id from location table in openmrs
location_tag_id	Unique column id from location_tag table in openmrs
location_tag_name	Name of location from location_tag table in openmrs
location_tag_description	Location Tag description from location_tag table in openmrs

c) **location_attribute_details_default -**

Openmrs Tables used :- location, location_attribute , location_attribute_type

Column Name	**Description**
location_attribute_id	Unique column id from location_attribute table in openmrs
attribute_type_id	location attribute type id from location_attribute table in openmrs
value_reference	value_reference from location_attribute table in openmrs
location_attribute_type_name	value_reference from location_attribute table in openmrs
location_attribute_type_description	location_attribute_type description from location_attribute table in openmrs
location_attribute_type_datatype	location_attribute_type_datatype from location_attribute table in openmrs
location_attribute_type_datatype_config	location_attribute_type_datatype_config from location_attribute table in openmrs
location_attribute_type_preferred_handler	location_attribute_type_preferred_handler from location_attribute table in openmrs
location_attribute_type_handler_config	location_attribute_type_handler_config from location_attribute table in openmrs
location_attribute_type_min_occurs	location_attribute_type_min_occurs from location_attribute table in openmrs
location_attribute_type_min_occurs	location_attribute_type_min_occurs from location_attribute table in openmrs

Operation Theatre

- Configuration
  - incrementalUpdateConfig is applicable (Refer Incremental Update)
- Existing OpenMRS tables
- Flattened Mart Tables

- Mart Views

CONFIGURATION

incrementalUpdateConfig is applicable (Refer Incremental Update)

```
{
      "name": "Operation Theater",
      "type": "operationTheater",
      "chunkSizeToRead": "500"
}
```

EXISTING OPENMRS TABLES

```
Surgical_block
Surgical_appointment
Surgical_appointment_attribute
Surgical_appointment_attribute_type
Provider
Person_name
Location
```

FLATTENED MART TABLES

```
surgical_block_default
surgical_appointment_default
surgical_appointment_attributes
surgical_appointment_attribute_type_details_default
```

a) surgical_block_default

Column Name	Description
surgical_block_id	Identifier for the table
primary_provider_name	Name of the Surgeon performing the surgery
creator_name	Name of creator for surgical block
location_name	Location of the operation theatre
block_starttime	Start time for surgery block
block_endtime	End time for surgery block
date_created	Date of creation for surgical block
date_changed	Changed date if modified
changed_by	Name of person who changed the surgical block

b) surgical_appointment_default

Column Name	Description
surgical_appointment_id	Table Identifier
surgical_block_id	Identifier for surgical block related to this surgery
patient_id	Identifier for the patient undergoing surgery
sort_weight	Weight will be used to order the appointment
status	Status of the surgery
actual_start_datetime	Actual Surgery start time
actual_end_datetime	Actual time at which surgery ended
notes	Notes added after surgery
date_created	Date when the surgery appointment was created
date_changed	If modified the date of modification of the surgery appointment
creator_name	Name of the person who created the surgery appointment
changed_by	Name of the person who modified the surgery appointment

c) surgical_appointment_attributes

Column Name	Description
surgical_appointment_id	Identifier for the surgical appointment
procedure	Name of the surgical procedure
estTimeHours	Estimated time for surgery in hours
estTimeMinutes	Estimated time for surgery in minutes extra to the above hours
cleaningTime	Time for cleaning after the surgery
otherSurgeon	Name of additional surgeon
surgicalAssistant	Name of assistant
anaesthetist	Name of the doctor who performs anesthesia
scrubNurse	Name of scrub nurse for the surgery
circulatingNurse	Name of circulating nurse (nurse from other location)
notes	Notes added after surgery is done

d) surgical_appointment_attribute_type_details_default

Column Name	Description
name	Name of the attributes used in the surgical appointment table
description	Description of the attributes used

MART VIEWS

a) patient_operation_theater_view

This provides a comprehensive view of operation theatre block details such as creation date, surgeries scheduled, location of the block, surgeon assigned, procedure to be carried out, etc. Since a patient can have multiple surgeries scheduled at different times there can be multiple rows , but of different data for the same patient. Patient details includes age_at_surgery (Age of the patient during surgery) and age_group_at_surgery (Age group of the patient during surgery). The surgical block date corresponds to all the surgeries planned in that block for that day. Whenever a surgery is postponed the block date will give the actual scheduled time of the surgery.

Person

- configuration
    - ColumnsToIgnore apply for this module (Refer Appendix)
    - incrementalUpdateConfig is applicable (Refer Incremental Update)
- Existing OpenMRS tables
- Flattened Mart Tables

CONFIGURATION

ColumnsToIgnore apply for this module (Refer Appendix)

incrementalUpdateConfig is applicable (Refer Incremental Update)

```
{
      "name": "Person",
      "type": "person",
      "chunkSizeToRead": "500",
      "groupedJobConfigs": [
        {
          "tableName": "person_details_default",
          "columnsToIgnore": []
        }
        ]
}
```

EXISTING OPENMRS TABLES

```
person
person_name
Patient_identifier
person_address
person_attribute_type
address_hierarchy_level
```

FLATTENED MART TABLES

```
person_details_default
person_address_default
person_attribute_info_default
address_hierarchy_level_default
```

a) person_details_default - This table provides all the person related information.

openmrs tables used:- person, person_name

Column Name	Description
person_id	Id reference to the person, from person table in openmrs
person_name_id	Name of person from person_name table in openmrs
preferred	Preferred from person_name table in openmrs
gender	gender of person from person table in openmrs
birthyear	year of person birth
birthtime	Time of person birth
birthdate_estimated	provided date of birth is estimated or accurate, is indicated with birth_estimated (true/false)
age	age of person from person table
age_group	Under which age range the patient belongs
dead	death status from person table (true of false)
death_date	death date in case
deathdate_estimated	if death date is not accurate then this field indicates the same (boolean value true/false)
cause_of_death	cause of death from person table

b) person_address_default - This table provides all the address information of persons.

openmrs tables used:- person_address

Column Name	Description
person_id	Patient identifier
preferred	Preferred address of the patients if they have multiple addresses
address1 - address15	Represents different fields in address (Configurable)
city_village	City/Village of the patient
state_province	State/Province of the patient
postal_code	Postal code of the patient’s residential area
country	Country of patient’s residence
county_district	District of the patient’s residence
latitude	Latitude of the area of residence
longitude	Longitude of the area of residence
start_date	Date when the patient started living in that area
end_date	Date when the patient stopped being in that area

c) person_attribute_info_default - This table provides all the information related person attributes

openmrs table used:- person_attribute_type

Column Name	Description
person_attribute_type_id	Id reference to the person attribute, from person_attribute_type table in openmrs
name	name of the person attribute type
description	description of the person attribute type

d) address_hierarchy_level_default- This table provides us the address hierarchy level information.

openmrs table used:- address_hierarchy_level

Column Name	Description
address_hierarchy_level_id	Id reference to the address, from address_hierarchy_level table in openmrs
name	name of the address field
parent_level_id	parent_level_id of address field
address_filed	To which address filed the data is filled in.

e) person_attributes : This table contains all the patient attributes that where used in the registration page. Columns will be generated dynamically based on the attributes that were used as a part of that specific implementation.

Eg : givenName, familyName, isCaretakerRequired, caretakerGender

Provider

- Configuration
    - ColumnsToIgnore apply for this module (Refer Appendix)
    - incrementalUpdateConfig is applicable (Refer Incremental Update)
- Existing OpenMRS tables

CONFIGURATION

ColumnsToIgnore apply for this module (Refer Appendix)

incrementalUpdateConfig is applicable (Refer Incremental Update)

```
 {
      "name": "Provider",
      "type": "provider",
      "chunkSizeToRead": "500"
}
```

EXISTING OPENMRS TABLES

```
provider
provider_attribute
provider_attribute_type
```

Flattened Mart Tables

```
provider_default
provdier_attributes
provider_attribute_details_default
```

a) **provider_default:-** This table gives all the provider information in an implementation

Openmrs tables used: provider

Column Name	Description

provider_id	provider id as in Openmrs
person_id	person id as in Openmrs
name	name of the provider
identifier	provider identifier for that provider (Not patient identifier)
provider_role_id	role_id for that provider

b) **provider_attribute:-** This table gives all the provider attribute information in an implementation

Openmrs tables used: provider:- provider, provider_attribute, provider_attribute_type

Column Name	**Description**
provider_id	provider id as in Openmrs
All other provider attributes as separate columns.	

c) provider_attribute_details_default - This table displays all the provider attribute details

openmrs tables used:- provider, provider_attribute, provider_attribute_type.

Column Name	**Description**
provider_attribute_id	Id reference for provider attribute (provider_attribute table in openmrs)
attribute_type_id	Id reference for attribute type (provider_attribute table in openmrs)
provider_id	Id reference for provider (provider table in openmrs)
value_reference	value reference is from provider_attribute table in openmrs
provider_attribute_type_name	attribute type name from provider_attribute_type table in openmrs
provider_attribute_type_description	Description of attribute type from provider_attribute_type table in openmrs
provider_attribute_datatype	Date type of attribute type from provider_attribute_type table in openmrs
provider_attribute_datatype_config	Date type config of attribute type from provider_attribute_type table in openmrs
provider_attribute_type_preferred_handler	preferred_handler of attribute type from provider_attribute_type table in openmrs
provider_attribute_type_handler_config	handler_type of attribute type from provider_attribute_type table in openmrs
provider_attribute_type_min_occurs	min_occurs of attribute type from provider_attribute_type table in openmrs
provider_attribute_type_max_occurs	max_occurs of attribute type from provider_attribute_type table in openmrs

Visits and Encounters

- Configuration
    - ColumnsToIgnore apply for this module (Refer Appendix)
    - incrementalUpdateConfig is applicable (Refer Incremental Update)
- Existing OpenMRS tables
- Flattened Mart Tables
- Mart Views

CONFIGURATION

ColumnsToIgnore apply for this module (Refer Appendix)

incrementalUpdateConfig is applicable (Refer Incremental Update)

```
{
      "name": "Visits And Encounters",
      "type": "visitsAndEncounters",
      "chunkSizeToRead": "500"
}
```

EXISTING OPENMRS TABLES

```
visit_type
visit
visit_attribute
visit_attribute_type
encounter
encounter_type
encounter_provider
encounter_role
episode_encounter
provider
person_name
patient
location
```

FLATTENED MART TABLES

```
patient_visit_details_default
visit_attribute_details_default
visit_attributes
patient_encounter_details_default
```

a) Visits

i) **patient_visit_details_default**

Column Name	Description
visit_id	Id reference for a visit
patient_id	Id reference for a patient
visit_type_id	Id reference for visit type
visit_type_name	Name of visit [IPD/ OPD/ Special OPD / Emergency etc]
visit_type_description	Description of visit type
visit_start_date	Start date of visit
visit_end_date	End date of visit

indication_concept_id	indication_concept_id from visit table in openmrs
location_id	Id reference for location
location_name	Name of location [general ward / Registration Desk, Labor ward etc]

ii) **visit_attribute_details_default**

Column Name	Description
visit_id	Id reference for a visit
visit_attribute_id	Id reference for a visit attribute
value_reference	Value for the attribute [OPD / IPD]
visit_attribute_type_id	Id reference for a visit attribute type
visit_attribute_type_name	Visit type [Visit status / Admission status]
visit_attribute_type_description	visit attribute type description from visit_attribute_type openmrs table
visit_attribute_type_datatype	visit_attribute_type_datatype from visit_attribute_type openmrs table
visit_attribute_type_datatype_config	visit_attribute_type_datatype_config from visit_attribute_type openmrs table
visit_attribute_type_preferred_handler	visit_attribute_type_preferred_handler from visit_attribute_type openmrs table
visit_attribute_type_handler_config	visit_attribute_type_handler_config from visit_attribute_type openmrs table
visit_attribute_type_min_occurs	visit_attribute_type_min_occurs from visit_attribute_type openmrs table
visit_attribute_type_max_occurs	visit_attribute_type_min_occurs visit_attribute_type openmrs table

iii) visit_attributes

Column Name	Description
visit_id	Id reference for a visit
Visit_Status	Status of visit [OPD / IPD]
Admission_Status	Status of admission [Admitted / Discharged etc]

b) Encounter

i) patient_encounter_details_default

Column Name	Description
patient_id	Id reference for a patient
visit_id	Id reference for a visit
episode_id	Id reference for a episode
encounter_id	Id reference for a encounter
encounter_type_id	Id reference for a encounter type
encounter_type_name	Type of encounter [Consultation / REG etc]
encounter_type_description	Description of encounter [Consultation encounter/Registration encounter etc]
edit_privilege	edit_privilege from encounter_type table in openmrs
view_privilege	view_privilege from encounter_type table in openmrs
location_name	Name of location [Labour ward/ General ward]

form_id	form_id from encounter table in openmrs
encounter_datetime	Date and time of encounter
encounter_role_id	Id reference for encounter role from encounter_role table in openmrs
encounter_role_name	name of encounter role encounter_role in openmrs
provider_id	Id reference of provider from encounter_provider table in openmrs
provider_name	provider name from provider table in openmrs
encounter_role_description	description of encounter role from encounter_role description table.

MART VIEWS

a) **Patient_visits_encounters_view**

This view allows the user to see the patient details data along with with all the visit and its details. It has also has the details of all the encounters at every encounter location. Each patient might have multiple visits hence the view have may have multiple rows of same patient data.

Medication

- Configuration
    - ColumnsToIgnore apply for this module (Refer Appendix)
    - incrementalUpdateConfig is applicable (Refer Incremental Update)
- Existing OpenMRS tables
- Flattened mart tables

The medicines ordered for the patients will be flattened into one table namely **medication_data**.

CONFIGURATION

ColumnsToIgnore apply for this module (Refer Appendix)

incrementalUpdateConfig is applicable (Refer Incremental Update)

```
{
      "name": "Medication And Orders",
      "type": "medicationAndOrders",
      "chunkSizeToRead": "500",
      "groupedJobConfigs": [
        {
          "tableName": "medication_data_default",
          "columnsToIgnore": [
          ]
        }
      ]
}
```

EXISTING OPENMRS TABLES

```
Drug_order
Drug
Orders
Order_frequency
Patient
Patient_program
Program
Encounter
Provider
Person_name
Obs
Concept
Concept_name
Location
```

FLATTENED MART TABLES

```
medication_data_default
```

a) **medication_data_default**

Column Name	Description
patient_id	Patient id
program_id	Program Id of the patient for which he is enrolled to.
patient_program_id	Patient program id
patient_program_name	Patient program name Eg: Reconstructive Surgery
order_id	Drug order id
coded_drug_name	Drug coded name Eg: FERROUS sulphate 256mg (=80 mg iron)
non_coded_drug_name	Drug non-coded name Eg: Candesartan
dose	Dose of the drug Eg: 1, 500
dose_units	Dosage units Eg: Tablets(s), mg
frequency	Drug consumption frequency Eg: Twice a day
route	Drug consumption route Eg: Oral, Intravenous
start_date	Medication start date

calculated_end_date	Medication end date
date_stopped	Medication stopped date
stop_reason	Medication stopped reason
duration	Medication duration(in quantity_units)
duration_units	Duration units Eg: Week(s), Day(s)
quantity	Drugs quantity for whole duration(in quntity_units) Eg: 14, 4800
quantity_units	Drugs quantity in units Eg: Tablet(s), mg
additional_instructions	Any other instructions given on medication
dispense	Whether drugs dispensed or not Eg: Yes
encounter_id	Id reference to encounter from encounter table in openmrs
orderer_id	Id reference to orderer from orders table in openmrs
orderer_name	name of orderer from order_type table in openmrs
visit_id	Id reference to the visit from visit table in openmrs
visit_type	Type of visit from visit_type table in openmrs
encounter_type_id	Id reference to encounter type from encounter_type table in openmrs
encounter_type_name	name of encounter type from encounter_type table in openmrs

Orders

- configuration
    - ColumnsToIgnore apply for this module (Refer Appendix)
    - incrementalUpdateConfig is applicable (Refer Incremental Update)
- Existing OpenMRS tables

The orders placed for the patients will be flattened into analytics database with orderable name (Eg: Lab Samples, Radiology, etc.). A separate table will be created for each orderable with same structure. The following is the example for lab samples.

CONFIGURATION

ColumnsToIgnore apply for this module (Refer Appendix)

incrementalUpdateConfig is applicable (Refer Incremental Update)

```
{
      "name": "Medication And Orders",
      "type": "medicationAndOrders",
      "chunkSizeToRead": "500",
      "groupedJobConfigs": [
        {
          "tableName":
          "columnsToIgnore": [
          ]
```

```
            }
        ]
}
```

EXISTING OPENMRS TABLES

```
Orders
Visit
Visit_type
Concept
Concept_set
Concept_view
```

Flattened Mart Tables

```
lab_samples
```

a) lab_samples

Column Name	Description
patient_id	Id of patient to whom order belongs
date_created	Order created dated
encounter_id	Encounter id
visit_name	Visit name Eg: Clinic
type_of_test	Type of test: Eg. Blood
panel_name	Different panels in a type of test. Eg: Hematology, Anaemia Panel are two panels in Blood
test_name	Test corresponds to the panel Eg: HPLC, Coombs Test (Direct), Coombs Test (Indirect), G6PD are four tests in Hematology

Diagnosis

- ColumnsToIgnore, ignoreAllFreeTextConcepts apply for this module (Refer Appendix)

Existing OpenMRS tables

Mart Tables

Mart Views

If one implementation has Diagnosis, **Visit_diagnosis** table will get created with existing data otherwise an empty table will be generated since **Visit Diagnosis** comes with Bahmni product by default.

Will get all the child concepts of **Visit Diagnosis** as columns.

ColumnsToIgnore, ignoreAllFreeTextConcepts apply for this module (Refer Appendix)

To generate this table we took product dump as reference.

Configuration

```
{
      "name": "Diagnoses And Conditions",
      "type": "diagnosesAndConditions",
      "chunkSizeToRead": "500"
}
```

EXISTING OPENMRS TABLES

```
Obs
Concept
Concept_name
location
visit
patient_program
```

MART TABLES

```
visit_diagnoses
```

a) **visit_diagnoses**

Column Name	Description
id_visit_diagnoses	Obs id
patient_id	Id reference for patient from patient table in openmrs
encounter_id	In which encounter data is captured
visit_id	Id reference to visit from visit table in openmrs
obs_datetime	Obs date time from obs table in openmrs
date_created	date_created from visit table in openmrs
date_modified	date_created from visit table in openmrs
location_id	Id reference to location from location table.
location_name	Name of location from location table in openmrs
program_id	Id reference to program from program table in openmrs
program_name	Name of program from program table in openmrs
patient_program_id	Id reference to patient program from patient_program table in openmrs
bahmni_diagnosis_status	
bahmni_diagnosis_revised	
bahmni_initial_diagnosis	

coded_diagnosis	Diagnosis name which exists in database
diagnosis_certainty	
diagnosis_order	

Mart Views

a) patient_diagnosis_condition_view

Column Name	Description
age_at_condition	Age of patient
age_group_at_condition	Age group under which the patient falls
birth_date	Birth date of patient
coded_diagnosis	Diagnosis value
condition_date_created	Date on which condition was recorded
condition_end_date	Date on which condition was removed
condition_id	Id reference for condition
condition_name	Type of condition
condition_onset_date	When condition was set in
creator	Who created the record
dead	
diagnosis_certainty	
diagnosis_order	
encounter_id	Id reference for encounter
end_reason	
gender	Gender for patient
nationalIdentificationNumber	ID for patient (such as passport)
non_coded_diagnosis	
obs_datetime	
patient_id	Reference ID for the patient
patientAddress	Patient Address
patientAddressLine2	Patient Address
patientCountry	Patient Address
patientDistrict	Patient Address
person_id	Reference ID for the person
previous_condition_id	ID for previous condition
status	Condition status
telephoneNumber	

Conditions

- ColumnsToIgnore apply for this module (Refer Appendix)

Existing OpenMRS tables
Mart Tables

If one implementation has Conditions, **conditions** table will get created with existing data otherwise an empty table will be created since **Conditions** comes with Bahmni product by default.

ColumnsToIgnore apply for this module (Refer Appendix)

To generate this table we took product dump as reference

Configuration

```
{
      "name": "Diagnoses And Conditions",
      "type": "diagnosesAndConditions",
      "chunkSizeToRead": "500"
}
```

EXISTING OPENMRS TABLES

```
Conditions
Concept_view : For concept name
```

MART TABLES

a) **conditions_default**

Column Name	Description
condition_id	Unique id for the table
previous_condition_id	
patient_id	To know for which patient these details are entered
status	
condition_name	Will have only one column for non-coded and coded condition names. Openmrs has two columns one is for coded and other is for non-coded
is_coded_condition_name	This column tell us whether a condition is coded or not
onset_date	
additional_detail	
end_date	
end_reason	
creator	Who has entered these conditions
date_created	Date of creation of these conditions

Bacteriology Data

All observations recorded against the concept Bacteriology Concept Set and it’s children will be in one table. All multi select and add more sections under this concept will become separate tables.

CONFIGURATION

ColumnsToIgnore apply for this module (Refer Appendix)

incrementalUpdateConfig is “NOT” applicable

```
{
      "name": "Bacteriology Data",
      "conceptReferenceSource": "",
      "type": "bacteriology"
}
```

EXISTING OPENMRS TABLES

```
Obs
visit
program
patient_program
location
```

FLATTENED MART TABLES

Since we do not have add mores/ multi-selects in bacteriology form in demo environment, there will be only one table :- **bacteriology_concept_s et** which will have all the details.

```
bacteriology_concept_set
```

Some of the columns in this table are as follows. Others columns are not mentioned here

Column Name	Description
id_bacteriology_concept_set	Id reference to bacteriology concept set from
patient_id	Id reference to patient from patient table in openmrs
encounter_id	In which encounter data is captured
obs_datetime	Observation date and time
date_created	Date when Bacteriology details are filled for the first time
date_modified	Date of recent Bacteriology details updation
location_id	Location code
location_name	Location name
program_id	Program code

program_name	Name of the program
patient_program_id	Id reference to patient program from patient_program table in openmrs
Specimen_collection date	Date of collection of the specimen
specimen_sample_source_noncoded	
specimen_sample_source	
specimen_id	

TRANSLATIONS

if we need to get those values as per the translations we need to add the below **locale tag** in the "**/var/www/bahmni_config/bahmni-mart /bahmni-mart.json**" file.

```
{
      "name": "Bacteriology Data",
      "conceptReferenceSource": "",
      "type": "bacteriology",
      "locale" : "fr"
    }
```

Observations

CONFIGURATION

ColumnsToIgnore, , ignoreAllFreeTextConcepts, separateTableConfig, enableForAddMoreAndMultiSelect apply for this module (Refer Appendix)

incrementalUpdateConfig is applicable (Refer Incremental Update)

```
{
      "name": "Obs Data",
      "type": "obs",
      "incrementalUpdateConfig": {
        "updateOn": "encounter_id",
        "eventCategory": "Encounter",
        "openmrsTableName": "encounter"
      },
      "separateTableConfig": {
        "enableForAddMoreAndMultiSelect": true,
        "separateTables": [
        ]
      },
      "conceptReferenceSource": "",
      "ignoreAllFreeTextConcepts": true,
      "columnsToIgnore": [
      ]
}
```

EXISTING OPENMRS TABLES

```
Obs
```

FLATTENED MART TABLES

All forms that are under the concept ALL OBSERVATION TEMPLATES will become separate tables in the mart database. Any multi select or add more sections under these forms will become separate tables by default. One can also choose the concepts they want to have as separate tables using separate Tables configuration in obs job.

Since we have 51 forms under All Observation Template of demo environment currently, there will be 51 separate tables. Additionally all multi select, add more sections and concepts from seperateTables config will become separate tables.

Below are the columns that are common in every form table

Column Name	Description
patient_id	Id reference to the patient from patient table
encounter_id	In which encounter data is captured
obs_datetime	Observation date and time
location_id	Location code
location_name	Location name
program_id	Program code
program_name	Name of the program
date_created	Date of form creation (When filled for the first time)
date_modified	Date of recent form updation (When any form fields are filled again or modified in same encounter)

Translations

if we need to get those values as per the translations we need to add the below **locale tag** in the "**/var/www/bahmni_config/bahmni-mart/bahmni-mart.json**" file.

```
{
      "name": "Obs Data",
      "type": "obs",
      "locale": "fr"
      }
```

Forms 2.0 Documentation

We could navigate Forms 2.0 either from clinical module or programs module.

CONFIGURATION

ColumnsToIgnore, ignoreAllFreeTextConcepts, separateTableConfig, enableForAddMoreAndMultiSelect apply for this module (Refer Appendix)

incrementalUpdateConfig is applicable (Refer Incremental Update)

```
{
      "name": "Form2 Obs Data",
      "type": "form2obs",
      "incrementalUpdateConfig": {
        "updateOn": "encounter_id",
        "eventCategory": "Encounter",
        "openmrsTableName": "encounter"
      },
      "separateTableConfig": {
        "enableForAddMoreAndMultiSelect": true,
        "separateTables": [
        ]
      },
      "conceptReferenceSource": "",
      "ignoreAllFreeTextConcepts": true,
      "columnsToIgnore": [
      ]
}
```

EXISTING OPENMRS TABLES

```
Obs
```

FLATTENED MART TABLES

The Forms 2.0 module deals with only Obs table. Once the data is filled against a particular form, upon executing mart, the form observations would be populated in the mart with the table name as “**form_name**”.

Below are the columns that are common in every form table

Column Name	Description
patient_id	Id reference for patient id from patient_id table.
encounter_id	In which encounter data is captured
obs_datetime	Observation date and time
location_id	Location code
location_name	Location name
program_id	Program code
program_name	Name of the program
date_created	Date of form creation (When filled for the first time)
date_modified	Date of recent form updation (When any form fields are filled again or modified in same encounter)

TRANSLATIONS

if we need to get those values as per the translations we need to add the below **locale tag** in the "**/var/www/bahmni_config/bahmni-mart /bahmni-mart.json**" file.

```
{
      "name": "Form2 Obs Data",
      "type": "form2obs",
      "locale": "fr",
      ..
      ..
}
```

USE-CASES

Some Additional information in use-case view:-

1. **When a form has a multi select coded answer**
    a. Separate table with the question name is created and the multi select answers are inserted in each row
    b. If the same question is used in a separate form, then the data is inserted in the same table if the table is created for the multi select question.
2. **When a form has a section which is not add more**
    a. The section data is inserted in the same form table without creating a new table
3. **When a form has a section which is add more**
    a. Section data is inserted in a separate table.
    b. Table name will be formname_Section name
4. **When a form has a section which is not add more and a multi select coded answer**
    a. Section data is inserted in Form table
    b. Multi select answer is inserted in a separate table
5. **When a form has a section which is add more and a multi select coded answer**
    a. Section data is inserted in a separate section table
    b. Multi select answer is inserted in a separate table
6. **When the data is voided from the patient observation**
    a. The data is removed from the table column
7. **When the form has translations added (From form builder implementer interface) and the mart is run in english**
    a. Data is inserted according to the english translations
8. **When the form has translations added and the mart is run in french**
    a. Data is inserted according to the french translations
    b. Previous data is also changed according to french translations
9. **When the column name is added in columns to ignore, (Fully specified name of the concept)**
    a. The column is removed from the table
    b. The entire table data is removed when there is only one column in the table and the same concept is added in columns to ignore.
10. **When the entire form is reconstructed, (All the form fields are removed and now concepts are added)**
    a. Existing data is removed from the table, and the new columns are added to the table
    b. New data is added to the form table when the observation is filled to a patient

Registration Second Page

- Configuration
    - ColumnsToIgnore apply for this module (Refer Appendix)
    - incrementalUpdateConfig is applicable (Refer Incremental Update)
- Existing OpenMRS tables
- Mart Tables
- Mart Views
- Translations

Registration on the second page is a default Bahmni feature.

CONFIGURATION

ColumnsToIgnore apply for this module (Refer Appendix)

incrementalUpdateConfig is applicable (Refer Incremental Update)

```
{
      "name": "Registration Second Page",
      "type": "reg",
      "columnsToIgnore": [],
      "separateTableConfig": {
        "enableForAddMoreAndMultiSelect": true,
        "separateTables": []
      },
      "incrementalUpdateConfig": {
        "updateOn": "encounter_id",
        "eventCategory": "Encounter",
        "openmrsTableName": "encounter"
      }
}
```

EXISTING OPENMRS TABLES

```
Obs
visit
program
patient_program
location
```

MART TABLES

```
reg_nutritional_values
reg_fee_information
```

In mart database, there will be two tables based on the configuration defined as part of the Bahmni config. The default Bahmni has the config as mentioned in the Appendix . As per the configuration following tables will be created. Any other concept can be defined in the same config. To configure Registration second page in Bahmni UI check the documentation.

a) **reg_nutritional_values** (For **Nutritional Values)**

Column Name	**Column Description**
id_reg_nutritional_values	Observation Id same as obs table of openmrs database
patient_id	For which patient the data was captured
encounter_id	In which encounter the data was captured
visit_id	Id reference of visit from visit table in openmrs

obs_datetime	obs date and time from obs table in openmrs
date_created	visit date created from visit table in openmrs
date_modified	visit date changed from visit table in openmrs
location_id	Id reference to location from location table in openmrs
location_name	Name of location from location table in openmrs
program_id	Id reference to the program from program table in openmrs
program_name	Name of program from program table in openmrs
patient_program_id	Id reference to patient program from patient_program table in openmrs
height	For given encounter and patient what is the value of height
weight	For given encounter and patient what is the value of weight

b) **reg_fee_information** (**Fee Information**)

Column name	**Column Description**
id_reg_fee_information	Observation Id same as obs table of openmrs database
patient_id	For which patient the data was captured
encounter_id	In which encounter the data was captured
visit_id	Id reference of visit from visit table in openmrs
obs_datetime	obs date and time from obs table in openmrs
date_created	visit date created from visit table in openmrs
date_modified	visit date changed from visit table in openmrs
location_id	Id reference to location from location table in openmrs
location_name	Name of location from location table in openmrs
program_id	Id reference to the program from program table in openmrs
program_name	Name of program from program table in openmrs
patient_program_id	Id reference to patient program from patient_program table in openmrs
registration_fees	For given encounter and patient what is the value of registration fees
comments	For given encounter and patient what is the value of comments

Note:- If a concept is present both in Observation form and in the registration second page, and the data for the concept is added from the Registration second page as well as one of the forms then the table under registration second page will have both the values under separate encounters.

However if a value is added from forms the same will only reflect under the obs table and not in the registration second page table.

MART VIEWS

There will be a separate table for each concept set configured in extension.json. To make it easily accessible there will be a view named as **regist ration_second_page_view** combining all the tables related to registration second page in mart database

patient_id	For which patient the data was captured
encounter_id	In which encounter the data was captured
visit_id	Id reference of visit from visit table in openmrs
obs_datetime	obs date and time from obs table in openmrs

date_created	visit date created from visit table in openmrs
date_modified	visit date changed from visit table in openmrs
location_id	Id reference to location from location table in openmrs
location_name	Name of location from location table in openmrs
program_id	Id reference to the program from program table in openmrs
program_name	Name of program from program table in openmrs
patient_program_id	Id reference to patient program from patient_program table in openmrs
reg_fee_information_registration_fees	Fee collected from that patient from obs table in openmrs
reg_nutritional_values_height	Height mentioned in the RSP from obs table in openmrs
reg_nutritional_values_weight	Weight of patient mentioned in the RSP from obs table in openmrs

TRANSLATIONS

if we need to get those values as per the translations we need to add the below **locale tag** in the "**/var/www/bahmni_config/bahmni-mart /bahmni-mart.json**" file.

```
{
        "name": "Registration Second Page",
        "type": "reg",
        "locale" :"fr",
}
```

Metadata

CONFIGURATION

```
{
        "name": "MetaData Dictionary",
        "type": "metadata",
        "conceptReferenceSource": ""
}
```

Existing OpenMRS tables

```
concept
concept_set
concept_name
```

Flattened Mart Tables

```
meta_data_dictionary
```

a) meta_data_dictionary- The metadata dictionary table displays all the information of concept sets that are mapped to **All Observation Templates** OpenMrs table. If there is a concept set ‘Disposition’ and two concept set members are mapped to it, below are the sample entries in the meta_data_dictionary mart table.

fully_specified_name	question	question_datatype	description	answer	answer_code
Disposition	Disposition	Coded		Answerconcept-1	
Disposition	Disposition	Coded		Answerconcept-2	

Column Name	Description
fully_specified_name	Full Name of concept from concept_name table in openmrs
question	Short Name of concept from concept_name table in openmrs
question_datatype	Data type of the concept
description	Description/ helptext of a concept
answer	answers mapped to the coded question concepts
answer_code	answer code

Disposition

CONFIGURATION

ColumnsToIgnore apply for this module (Refer Appendix)

incrementalUpdateConfig is applicable (Refer Incremental Update)

```
  {
       "name": "Disposition Data",
       "type": "disposition",
       "columnsToIgnore": [],
       "incrementalUpdateConfig": {
         "updateOn": "encounter_id",
         "eventCategory": "Encounter",
         "openmrsTableName": "encounter"
       }
     }
]
```

EXISTING OPENMRS TABLES

```
concept
concept_set
concept_name
patient
visit
obs
```

```
location
program
patient_program
ecnounter
```

Flattened Mart Tables

```
disposition_set
```

a) dispostion_set- This table provides the information about the disposition of the patient enrolled from programs/clinical module.

Column Name	**Description**
id_disposition_set	Id reference to the disposition set
patient_id	Id reference to the patient from patient table in openmrs
encounter_id	Id reference to the encounter from encounter table in openmrs
visit_id	Id reference to the visit from the visit table in openmrs
obs_datetime	obs date time from obs table in openmrs
date_created	date of disposition got created
date_modified	date of disposition got modified
location_id	location id of the patient
location_name	location name
program_id	Id reference to what program that particular patient is assigned to
program_name	Name of the program assigned
patient_program_id	patient_program_id from patient_program openmrs table
disposition_note	Disposition note entered while enrolling for disposition
disposition	Disposition of which the patient enrolled to

TRANSLATIONS

if we need to get those values as per the translations we need to add the below **locale tag** in the "**/var/www/bahmni_config/bahmni-mart /bahmni-mart.json**" file.

```
{
      "name": "Disposition Data",
      "type": "disposition",
      "locale": "fr"
}
```

Incremental Update

Bahmni-mart supports incremental update during data flattening.This can be easily enabled by adding incremental update config in **/var/www /bahmni_config/bahmni-mart/**

bahmni-mart.json for any specific job.

> **Note**: Due to some technical differences in Bahmni, incremental update config is not suggested for bacteriology job. Find more details in openmrs-talk

Check here for example.

Key Name	Description	Default Value	Required
updateOn	Column name in the analytics database depends on which incremental update should work	N/A	yes
eventCategory	Category name present in event_records table (openmrs database)	N/A	yes
openmrsTableName	Table name in OpenMRS for given eventCategory	N/A	yes

Email Configuration for Notification Regarding Failed Jobs

A mail will be received incase of any job fails after every run of bahmni-mart application.

For getting mail you have to provide values for **'mail_subject'**, **'mail_from'** and **'mail_recipients'** in **'bahmni-mart-playbook/roles/bahmni-mart/defaults/main.yml'**. For sending mail to multiple recipients, you can mention all the recipient mail ids separated by commas (see below example).

Note - You won't get mail if values of **mail_recipients** or **mail_from** is empty.

For Example -

You have to provide values for **'mail_subject'**, **'mail_from'** and **'mail_recipients'** in **'bahmni-mart-playbook/roles/bahmni-mart/defaults /main.yml'** like below example.

mail_subject = "Notification regarding failed jobs"

mail_from = "no-reply@bahmni-mart.notifications"

mail_recipients = "recipient_one@gmail. com, recipient_two@gmail.com"

Whenever job fails you will get mail into spam folder of given recipients. For getting mail to inbox rather than going to spam folder, the recipients have to apply the below filter procedure.

<u>Filter Process for getting mail in Inbox</u>

1. You have to go to recipients mail.

1. In mail at top right corner there is Setting sign.

1. Click on that it will give dropdown. In dropdown click on the Settings.

1. Then go to Filters and Blocked Addresses tab.

1. Click on 'Create a new filter' and then provide value of

'mail_from' address which given in **bahmni-mart-playbook/roles/bahmni-mart/defaults/main.yml** file (like given in the example - no-reply@bahmni-mart.notifications) in From text block.

1. Then click on 'Create filter' and tas Never send it to Spam to that filter and click on Create filter again .

By doing this it will create a filter and mail will come to inbox.

<u>For Developers - To Run locally</u>

Your application-dev.properties file should config with below values for getting mail.

bahmni-mart.mail.subject = Notification regarding failed jobs

bahmni-mart.mail.from = "no-reply@bahmni-mart.notifications"

bahmni-mart.mail.recipients = "recipient_one@gmail. com, recipient_two@gmail.com"

applictaion-dev.properties file path -

/opt/bahmni-mart/properties/application-dev.properties

After this you have to follow above filter process for getting mail in inbox rather than in spam.

Appendix

All the configurations mentioned below need to be added in **bahmni-mart.json**

```
Obs
bacteriology
orders
diagnosis
rsp (Registration Second Page)
```

Key name	description	Default value	Required
name	Job name	N/A	Yes
type	Job type	N/A	Yes
separateTables	This config refers the different tables that need to be separated	N/A	No
conceptReferenceSource	This config is used for data masking. Here the concept reference source can be added as per the OpenMRS database	N/A	No
ignoreAllFreeTextConcepts	This config when enabled ignores all the free text concepts	True	No
columnsToIgnore	This config ignores the columns which are not needed to be shown on analytics DB	N/A	No
includeFreeTextConceptNames	This config includes the text columns of all concept names given in config		
locale	Mart flattens the tables/columns based on given locale	en	No

1. **EAV**

Key name	description	Default value	Required
name	Job name	N/A	Yes
type	Job type	N/A	Yes
chunkSizeToRead	This tells how many rows to read in the source DB before writing it into analytics DB	N/A	Yes
tableName	Target table name in analytics DB	N/A	Yes
eavAttributes			
attributeTypeTableName			
attributeTableName			
valueTableJoiningId			
typeTableJoiningId			
valueColumnName			
primaryKey	Primary key in the target table	N/A	Yes
columnsToIgnore	This config ignores the columns which are not needed to be shown on analytics DB	N/A	No

1. **Metadata**

Key name	description	Default value	Required
name	Job name	N/A	Yes
type	Job type	N/A	Yes
conceptReferenceSource	This config is used for data masking. Here the concept reference source can be added as per the OpenMRS database	N/A	Yes

1. **CSV upload**

Key name	description	Default value	Required
name	Job name	N/A	Yes
type	Job type	N/A	Yes
sourceFilePath		N/A	Yes

1. **CustomSql**

Key name	description	Default value	Required
name	Job name	N/A	Yes
type	Job type	N/A	Yes
chunkSizeToRead	This tells how many rows to read in the source DB before writing it into analytics DB	N/A	Yes
tableName	Target table name in analytics DB	N/A	Yes
columnsToIgnore	This config ignores the columns which are not needed to be shown on analytics DB	N/A	No
readerSql	This sql reads data from source database (OpenMRS)	N/A	Yes (either readerSql or sourceFilePath)
sourceFilePath	This key is an alternative for **readerSql**. In this case a file path needs to be mentioned	N/A	

Frequently(F) Asked(A) Questions(Qs)

Name of the contributers :
Himabindu Thungathurty ,Pritam Das ,Rakesh Kumar ,Supriya Muppiri ,Tarun Shettygari ,Vinisha Donthula

Installation FAQs

Q1. How can I do a dev set-up for bahmni-mart?

A: After cloning follow the below steps:-

- Update /src/main/resources/application-dev.properties and /bahmni-mart/src/test/resources/application-test.properties with ur corresponding vagrant ip and password(if required)
- Connect to the database of your local vagrant with the IDEA u are working in
- Run /scripts/dev/testMysql.sql and sudo mysql -pP@ssw0rd123 -e "GRANT ALL ON test_openmrs.* to'test_user'@'localhost';FLUSH PRIVILEGES;" in mysql database
- Run /scripts/dev/psqlTestSetup.sql and psql -U test_user test_analytics -c "CREATE SCHEMA bahmni_mart_scdf; in postgres database

Q2. How can I install mart in my local system and point to bahmni installed in Vagrant or any other environment?

A: May be the steps for gradle build and run commands we can provide. And also provide how to do the bahmni configuration.

Q3. How can I install mart without metabase using the ansible playbook?

Q4. Which version of Postgres is required for Mart as a Prerequisite?

A: For Bahmni version >=0.89 & < 0.92 -> Install PostgreSQL 92 (update the link)

For Bahmni version >= 0.92 -> Install PostgreSQL 96 (update the link)

Q5. How can I override the default mart configuration for any particular implementation?

A: You can place an overridden version of bahmni-mart.json at `/var/www/bahmni_config/bahmni-mart/` path of your implementation environment.

Q6. How can i get started with mart?

A: Bahmni Mart Installation Setup

Q7. What is the suggested DB tool to view the analytics table and the procedure to Connect mart to the DB tool?

A: Bahmni-Mart Setup with MySQLWorkbench & DBeaver

Q8.How to configure the mart jobs?

A: Bahmni-Mart Json File

Q9: Where is mart.json file located on the server?

A: ./var/www/bahmni_config/bahmni_mart/mart.json

General FAQs

Q1. How to resolve Cannot connect to mysql or cannot connect to postgres or Communication link failure

A: Restart mysql and postgres servers and make sure you are connecting with correct password

Q2: How to ignore columns in mart tables

A: While creating mart tables you don't want some fields from the openmrs table. You can add them to the "columnsToIgnore" field.

Q3: One of the jobs fails(Eg: Microbiology) with out of memory issue, what should I do?

A: Bahmni mart **reads job data** from Bahmni and keeps it in memory until it pushes to ***analytics*** database. If the data in Bahmni is very large for that specific job memory overflow issues may occur. As of now bahmni-mart doesn't have streaming support for all the jobs except ***orders***. Please request for the feature enhancement.

Q4: Can I remove any jobs in bahmni-mart.json if I don't need it?

A: Yes, it is safe to delete(or add) any unwanted job configurations from bahmni-mart.json

Q5. What is Bahmni Mart?

A: We need 1-2 line overview of Mart.

Q6. What is Metabase?

A: We need 1-2 line overview of Metabase.

Q7. What to check if any view is failing while running mart?

A: Make sure if you have corresponding configuration in mart. For example if you don't have registration second page configured for your implementation the corresponding view would be failing

Q8. What are the databases supported by Mart?

Q9: When to use Custom Sql & when to use sql files

A: When you think more than one job can use the common sql queries, you can create a separate sql file and mention the type field.

Else you can mention customSql in the type field and write the sql to be executed in readerSql field'.

- How to view the log files.
- How to stop the running mart.
- What will happen if one of my jobs fails?

Q10: Why the incremental load is not working. Its always doing the full load.

A: Incremental load depends on the atom feed events that are published by openmrs. And we have the option to turn it on or off based on the usage. And we have a scheduler to publish the events from time to time. It can be configured through openmrs admin "Manage scheduler" page. Below is the property for the same

"OpenMRS event publisher task" If incremental load is not working fine, please check the above scheduler is running fine/not.

TechnicalFAQs

Q1: How Does the data type of the values change in Bahmni mart compare to openers?

Q2: How boolean is getting stored at bahmni-mart tables?

Q3: What is the difference between form1 and form2 in mart?

A: Form 1 tables and Form 2 tables both will have the entries for the observations filled.

But Form 1 tables will have the "id_formName" column which will contain "obs id"(openmrs) of the form concept.

Form 2 tables will have the "form_field_path" column which will contain the form name.

Q3: What is form_field_path and reference_form_field_path?

A: These both columns will come into picture when a separate table got created for either add-more /multi-select concepts and sections.

"form_field_path" will contain the form name along with the control id followed by count of the control among all the controls that got added using add more(i.e. Path from section to the obs).

```
Eg:  form-name.{section_id} -
{add_more_count_of_section}/{obs_control_id} -
{add_more_count_of_obs}
```

"reference_form_field_path" will contain the path of section in the form(i.e. path from form name to the section control).

```
Eg:  form-name.{section_id} -
{add_more_count_of_section}
```

Q4: Why does my *customSQL* job fail with SQL syntax error even though the SQL is correct?

A: The issue could be with *columnsToIgnore* configuration. Mart internally parses the given SQL query in ***customSQL*** jobs using ***jsqlparser*** and removes the columns provided in ***columnsToIgnore*** . If the ***select*** part of the query is too complex(with aggregate functions etc), ***jsqlparser*** may not be too intelligent to parse the SQL query. You can solve the problem by **avoiding complex select parts in the outermost *select* clause** or update the SQL query to remove unwanted columns so *columnsToIgnore* configuration can be ignored

Q5: How to fix, mart fails with FileNotFoundException ?

```
java.io.FileNotFoundException: /var/www
/bahmni_config/bahmni-mart/bahmni-mart.json
```

A: Make sure the bahmni-mart.json file is there in the /var/www/bahmni_config folder. Otherwise run below task to create a symlink from /etc/bahmni-mart/conf path

ln -s /etc/bahmni-mart/conf bahmni-mart.

Usually when we update the config the symlink gets removed.

Q6: Why mart is failing with java.sql.SQLException and not able to connect to the databases?

A: Please check the passwords for both openmrs and analytics databases in /etc/bahmni-mart-playbook/setup.yml file. If it needs a change please update in setup.yml and re-run the mart installation.

Functional FAQs

Q1. Does Bahmni-mart considering older data or data from any specific date

A: Yes, and marker table can be updated for using data from a specific date

Q2: How to remove incremental update config(or any modifications) for any grouped jobs?

A: If you maintain your own mart code then you can go to the corresponding grouped job config and make modifications there.All grouped job config jsons will be placed under /src/main/resources/groupedJobs

(or)

If you don't have control over mart code copy the json contents from the corresponding grouped job json to your /var/www/bahmni_config/bahmni-mart/bahmni-mart.json and make the required modifications

Q2: What is the use of giving separate table config?

A: You can add configuration to create new tables for all multi-select and add more concepts by enabling enableForAddMoreAndMultiSelect   to true

Q3: What if we want to create a separate table for a concept which is neither multi-select nor add more?

A: You can configure the concept to have separate table created by giving concept in "separateTables": []

Q4. Is there a case where we should run mart in only full load instead of incremental load?

A: If you make any changes in bahmni-mart.json, run the mart in full load once and proceed.

Q5: I don’t see the data in mart analytics database though mart runs successfully

A: Mart does incremental loading using the events in Bahmni/openmrs by default. If there is any delay in raising the event or if the event skipped with unknown reasons by mart, you don't see the data. It is recommended to do a full-load of mart once in a while. You can do full load safely by truncating markers table and deleting all the tables in the analytics database.

Q6: There are some tables in mart which are not related to Bahmni anymore

A: If you do any metadata changes(Eg: changing form name) in Bahmni, the mart is not so intelligent to clear the stale tables. It is recommended to do full-load when there is a change in Bahmni metadata or any configuration is updated in bahmni-mart.json

Q7. How can I override the default mart configuration for any particular implementation?

A: You can place an overridden version of bahmni-mart.json at

```
/var/www/bahmni_config/bahmni-mart/path of
your implementation environment.
```

Q8.What is the difference between Full load and Incremental Load for Mart?

Q9.What data do i get after the full Load?

Q9: What are the different cases in which I can't see the form in mart table?

A: Following cases are possible:

- When you form has duplicate concepts, the table creation for the form will be skipped
- When the concept is more than 50 characters, postures will skip creating the table

Q10: How are the special characters handled in the analytics table?

A: Special characters are replaced with underscores (_)

Q11: Diff between full load and incremental load and how to switch between them. And what's the best option?

A:

Security FAQs

Q1.How to protect the patient sensitive data from flowing to analytics table?

A: You can add the specific concepts to columns to ignore. Those concepts and the data related to the concepts will not be displayed

Q2. Where can we configure the password for Postgres?

A: You can configure password in the playbook.

```
/etc/bahmni-mart-playbook
```

Performance FAQs

Q1: Does custom query impact on performance?

Q2.How long will it take to run mart?

A: It depends majorly on the following 2 factors:

- It depends on the number of jobs configured
- Data present in the openmrs tables.

Q3.What is the suggested way to run the bahmni-mart. (Incremental, Full Load)