Reporting Framework Design Specification

[Goals](#_7t39od803q0u)

[Summary](#_rmkrmj78azwg)

[Types of Reports](#_xdzwpmrkuwmq)

[1. Aggregated level report](#_4nt6urjwhyi)

[2. Group of patient level report](#_kx80any0rxsq)

[3. Individual patient level report](#_3nhwzbm8xyxp)

[Basic Architecture](#_gr2j32923mcd)

[1. Transaction database](#_6aa8j7kf5xim)

[2. Data Warehouse](#_pjwwo4muftt)

[3. Concept dictionary](#_m3emzt96s4i2)

[4. Visualization and reporting tools](#_vlybbmsb3xad)

[Aggregated level report](#_eeetpdgqltap)

[Group of patient level report](#_78tbvz297mb6)

[Individual patient level report](#_o8wza91ii4u0)

[Glossary](#_hbmkq1u1m4d7)

# Goals

Improve the flexibility of the iSante reporting system to:

* remove reporting load from developers
* empower the users
* transition to Haiti

**Release Targets**

April 15 Release

* Google Public Data Explorer with current concepts + malaria indicators

June 30

* Data Warehouse description
* Partial Data Warehouse
* Simple reporting generation system for Group reports

# Summary

Currently, for every report that the end user wants, a developer has to code the appropriate queries. Ideally, the end user should be able to build the reports he wants by selecting the concepts and the template. The report would then be generated on the fly.

Another functionality that is expected is the ability to drill down through the data to get to the desired datapoint.

# Types of Reports

There exists 3 levels of reports that an end user would like to generate.

## 1. Aggregated level report

GRAIN: national>department>clinic

An aggregated report that enables drilldown and cross-comparisons through the data (from national to department to clinics) and cross comparison between concepts. If data is de-identified properly, this can be outsourced to an external visualization framework.

The goal is to have higher level analytics and numbers. It can be useful for epidemiologists or data quality experts.

Examples of queries:

* How many patients were there on 12/12/2011 who were active in Jacmel
* How many patients were there on 12/12/2011 who were active in Jeremie
* Both of the previous queries at the same time for comparisons.

Msin users: Data Quality Analysis, Institutions

Remark: I-TECH in general needs to be good with indicator based data/exploration

Remark: These reports can help generating group of patients reports  based on the selection (see Google Public Data Explorer: Link Parser)

## 2. Group of patient level report

GRAIN: clinc>patient

A list of identified patients that are selected by a set of features. This should be an iSante feature.

Ideally each individual patient should be clickable in order to access to more details.

Examples of structured queries:

* List all the patients that are males, HIV+, have malaria and are less than 35 in Jacmel.
* List all the patients that had an encounter in the past 3 days.

Main users: MDs, Nurses, Data Clerks

Remark: this is an openMRS feature that has poor performance.

## 3. Individual patient level report

GRAIN: patient

A tailored report with data concerning the selected patient. This should be an iSante feature.

Examples of information displayed in the report:

* demographics
* clinical info
* last encounters
* next expected visit
* status

Main users: MDs, nurses

se

## 2. Data Warehouse



A Data Warehouse (DW) for reporting overall would allow for optimisation of the reporting queries and ease of understanding of the available data for the end users.

**Benefits of a DW**

* present data to the user in a planned/thought out way
* performance improvement against the current transactional database

**Outputs**

* DSPL dataset for Google Public Data Explorer for Aggregated level reports 
* Datamart of CD4's with specific visualization 
* Patient and Group level reports generated by the end user

Remark: Paul Bugni does similar work with DOH/INHS, but stratifies them by different values

**Approaches to creating the DW**

1. Starting with the current tables that are in Data Warehouse format

Some tables are already calculated every day for the past 24 months for each patient for reporting purposes.

Currently done:

* patient statuses (patientStatusTemp)
* regimen (pepfarTable)
* cd4 accumulated and normalized (cd4Table)
* normalized discontinuation information (discTable)

Straightforward process to create temp tables but no regular schedule for computing these tables yet:

* age (function fillPidAgeTable in backend/sharedRptFunctions.php)
* pregnancy (function fillPregnancyTable in backend/sharedRptFunctions.php)
* Incidence (function setupIncidenceTable in backend/sharedRptFunctions.php)

To do:

* Prevalence
* visit/encounter history (encValidAll view)
* drug prescription history (a\_drugTable view)
* drug dispensation history (a\_medsDispensed view)
* drug discontinuation history (drugSummaryAll table)
* lab order history (a\_labsOrdered view?)
* lab result history (a\_labsCompleted view?) [probably include CD4 results here, or make it separate?]
* weight history (multiple locations including obs, a\_vitals, others?)
* height history (multiple locations including obs, a\_vitals, others?) [maybe combine weight and height histories and add a calculated BMI history all into one table/view/whatever we end up using?]
* immunization history (multiple locations including obs, immunizations, others?)
* symptom history (multiple locations including obs, a\_symptoms, others?)
* risk assessment history (multiple locations including obs, a\_riskAssessments, others?)
* physical exam history (multiple locations including obs,
* eligibility for ART (eligibility table)
* eligibility for cotrimoxazole therapy (also in eligibility table)
* eligibility for PMTCT prophylaxis (multiple locations including obs, a\_medicalEligARVs, others?)
* tb status history (multiple locations including obs, a\_tbStatus, others?)
* psychomotor development history (multiple locations including obs, a\_vitals, others?)
* parity (T-P-A-L) history for women (multiple locations including obs, a\_vitals, others?)
* family planning history (multiple locations including obs, a\_vitals, others?)

2. Using an existing tool like pentaho



Pentaho provides a platform for

1) describing your current schema

2) describing a target warehouse schema

3) describing and executing a process for transforming transaction data to warehouse data

4) presentation and visualization tools for the data.

<http://www.pentaho.com/>

**Step By Step setup for Data Warehouse Job**

1. go to <http://kettle.pentaho.com/> and download the latest stable release of Kettle in the Recent News and Releases section (can be done directly to the server with wget <http://downloads.sourceforge.net/project/pentaho/Data%20Integration/4.2.0-stable/pdi-ce-4.2.0-stable.tar.gz>)
2. extract it to the desired directory on the server (any directory would work) with tar -xvf pdi-ce-4.2.0-stable.tar.gz
3. cd data-integration
4. chmod +x \*.sh
5. upload the historique.tar archive and extract it in the data-integration directory
6. Import the historique/database/historique.sql file into the database
7. You can now run the Snapshot Job as follows: ./kitchen.sh -file="./historique/snapshot.kjb" -level=Minimal -param:JOB\_DATE=2012-02-01 -param:DATABASE=**<database name>** -param:USER=**<database user>** -param:PASSWORD=**<database password>**

The job date should be the date of the day where the job is executed. It is the date that will be stored in the dw\_snapshots table in the database.

More info on Kitchen: <http://wiki.pentaho.com/display/EAI/Kitchen+User+Documentation>

**Architecture of the Data Warehouse**

Snapshot based

iSante/Gheskio/PIH clinical summaries (CDA’s) --> ETL --> DW

Data Warehouse created by daily snapshots of the current patient information available in the iSante Database.

Data Warehouse sources

* patient
* discTable (discontinued reason)

Data Warehouse tables

dw\_daily\_patient

* snapshotDate
* patientID
* clinic
* department
* gender
* hiv
* status only applies to HIV positive
* discontinued reason
* pregnant ? oui/non relative to date of snapshot?
* age ? probably want dob
* conditions ? (would be an array of conceptIDs) current active conditions, also drugs, labs, etc???
* regimens ? (would be an array of treatments) patient really only changes regimen when either it is not effective or drugs aren’t available, so only one regimen at a given date and only one *current* regimen

? : means that it is not done yet

Taking a snapshot of the database at time t means filling these tables using current data from the database.

A daily snapshot would be taken.

The estimated weight of a snapshot is 15Mb.

The estimated time of computation for a snapshot is 1h.

Every month, only the last day of the month is kept. you mean you keep daily for only the current month?

A year’s worth of snapshots would weigh 1Gb.

questions:

* can a patient have multiple encounters on a same day? yes, even of the same type. that is why we have seqNo in the encounter table
* can a patient have multiple regimens at the same time? no--see above
* should we have a table per snapshot date or keep all the dates in the same big table?

Log based

Paul Bugni for INHS does it slightly differently - HL7 messages (rather than clinical summaries/CDA’s) to a series of logs (lab log, ER visit log, pharm log, etc) --> ETL --> DW

messages are event-level, about a specific patient that had an event occur.

**Data Warehouse stub for Google Public Data**

remark: to execute the Data Crunching queries, there is a need for more computing time. The php.ini file should be edited with this line:

max\_execution\_time = 1800

Lookup Tables (Clinic, Department, HIV, gender, PatientStatus)

Compulsory fields:

* conceptID
* nameEn
* nameFr

Optional Fields:

* parent concept
* longitude
* latitude

Slice Tables

Concept fields:

* gender
* status

Time granularity:

* day (monthly values)

Metrics:

* population
* encounters
* percentages

## 3. Concept dictionary

Forms

Data elements

Reports

ddquery.php

## 4. Visualization and reporting tools

Report header:

sex[slice\_role=dimension;type=string;rollup=true],age[slice\_role=dimension;type=string;rollup=true],patientStatus[slice\_role=dimension;type=string;rollup=true],hivPositive[slice\_role=dimension;type=string;rollup=true],discontinuedReason[slice\_role=dimension;type=string;rollup=true],population[slice\_role=metric;aggregation=SUM],network[slice\_role=dimension;type=string;rollup=true],date[type=date;format=yyyy-MM-dd]

11100;1;;;0;;;13;COAG MSPP;2009-04-30

### Aggregated level report



**Google Public Data Explorer**



The goal was to generate reports that would allow drill down and cross comparison using an external framework.

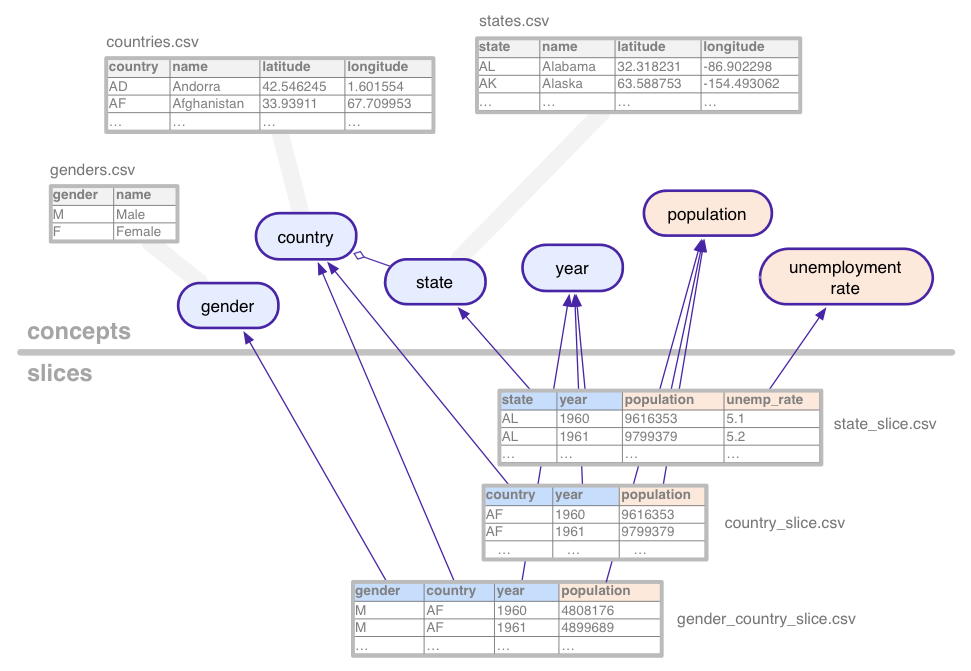
The Google Public Data Explorer has been chosen for the following reasons:

* free
* used by international organizations (OECD, UN, IMF)
* quality of visualizations
* reasonable customer support
* open source metadata XML format for visualization (aka DSPL)

The work was to extract data from the iSante database and convert it into a format that would be readable by the Google Public Data Explorer.

DSPL

DSPL is a data and metadata format designed from the ground up to support powerful, interactive visualizations like those in the [Google Public Data Explorer](http://www.google.com/publicdata). See the [overview](https://developers.google.com/public-data/overview?hl=fr) for more details and to get started.

Indicators are organized in terms of concepts (blue), metrics (red) and slices that are the visualizable csv tables.

Features

* drill down feature - national to department to clinic level
* comparison notion - which concepts are you comparing by
* filters notion - filter data by criteria
* cross metrics –
  + ex: 3 dimensions - geolocation, size of clinic, number of encounters for a given month
  + ex: 5 dimensions - # active on arv, # new on arv, size of circle is # of pats, color of circle is # of encounters -- timeline to see trend over time

Security & privacy

* data is de-identified
* visibility can be private / invite only
* collaborators can update the data in the dataset
* owners can update and change sharing and admin users

**De-Identification (Privacy)**

The need for de-identification of the data arises with the will to use external resources to visualize and generate aggregated level reports 

De-identification takes place between the Data Warehouse and the external visualization tool (eg Google Public Data Explorer).

There exists two types of de-identification procedures :

**1. Basic de-identification**

Removing information from the 18 HIPAA identifiers (name, addr, zip code...) so that data in the dataset is not identifiable.

**2. Preventing re-indentification**

Problem

Matrix of patient attributes can also lead to identification of patients where the cross point leaves only 1-2 patients within that point. The patients can then be reidentified and it is possible to deduce every information about them.

Example

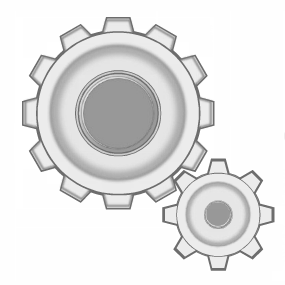
Ex - what about when there is only 1-2 patients per clinic for certain categories of patients (ex, adolescents being treated in a certain clinic)?

Technical solution

If less than 5 patients within cross point then it is reported as 0 OR random number between 1 and 5.

Geographic data: add fuzziness around where the patient is located. i.e. randomly move the patient to a nearby location.

# Current stage

iSante side Implementation

1. The DSPL file dataset.xml, written manually, describes the dataset in terms of:
   1. concepts (with hierarchy)
   2. slices
   3. basic information (provider, title...)
   4. localization to French
2. All the files (dataset.xml + all the csv slices) are archived into a zip that is then uploaded to<http://www.google.com/publicdata/admin>
3. Google computes the data and pregenerates the visualizations.
4. If the operation succeeds, it is possible to open the Public Data Explorer.

Current Dataset

Metrics

* number of patients (population)
* number of encounters (encounter)

Concepts

- For the number of patients

* department
  + clinic
* gender
* pregnancy
* status in the clinic (new in clinic, new on arv, inactive patient... etc)

Remark: Bill's point about status in the clinic is that these are very EMR-centric. Only seeing what the patient status is in the EMR.

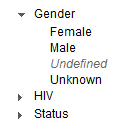
**Next Steps**

* It should be possible to add new concepts and metrics.
* A Link Parser should enable the conversion of an aggregated view 

### Group of patient level report

**Design proposition**

It would be possible to reuse the concepts developed for the Google Public Data Explorer to create a query builder for groups of patients queries.



Selecting these filters and then clicking the QUERY button would retrieve the list of clickable patient ID that match these filters.

End users would be able to get used to these concepts intuitively through the Public Data Explorer.

Based on the openMRS cohort builder

Query builder:

* Concept/Observation
* Patient Attributes
* Encounter
* Program Enrollment Drug Order
* Composition

We would have to distinguish between hard coded values and concept style values.

### Individual patient level report

**Design proposition**

First phase: making it possible to select the concepts to display in the report (either with a checkbox list of the available concepts or through a concept dictionary system).

Second phase: choose a template to display these concepts.

Remark: A prototype of the two previous reports has been created that will parse a link built with the google public data explorer and generate the corresponding SQL query.

See support/google-visualization/PublicDataLinkParser.py

# Glossary

**Dimension** is a term in data management and data warehousing that refers to logical groupings of data such as geographical location, customer information, or product information.

**Slowly Changing Dimensions (SCDs)** are dimensions that have data that changes slowly, rather than changing on a time-based, regular schedule.

# Appendix

## Installing Pentaho Kettle

Download kettle stable version from kettle.pentaho.com

### On Linux

go in the unzipped data-integration folder

start a terminal

cd data-integration

type the command: sh spoon.sh

If it doesn’t work, install java first

Then create a tunnel to the MySQL database

ssh -L3306:localhost:3306 user@server.com

### On Windows

go in the unzipped data-integration folder

start a terminal

double clic on spoon.bat in the data-integration folder

If it doesn’t work, install java first

Then create a tunnel to the MySQL database

ssh -L3306:localhost:3306 user@server.com

To do this you might need to install Putty.