IDART DATA MART

By

ZUKILE RORO

A report submitted in partial fulfillment of the requirements for the degree of BSc Honours (Computer Science)

University of the Western Cape

2010

University of the Western Cape
Department of Computer Science
Supervisor: Dr William D. Tucker

ABSTRACT

IDART DATA MART

by Zukile Roro

Supervisor: Dr William D. Tucker
Department of Computer Science

The Intelligent Dispensing of ART (iDART) is the software solution designed by Cell-Life to support the dispensing of antiretroviral drugs in the public health sector.
The purpose of this project is to combine data from multiple instances of iDART into a single data mart that can be used by Cell-Life for analysis and reporting. The data mart design will use the star schema instead of snowflake schema. The advantage of using this schema is that it reduces the number of tables in the database.

A dashboard user interface will be used. Implementing a dashboard will allow Cell-Life to find an overall view of antiretroviral drug treatments. A High-Level Design provides an overview of the system, and includes a high-level architecture diagram depicting the components and interfaces that are needed. The low level design will contain: detailed functional logic of the module in pseudo code, database tables with all elements including their type and size, all interface details with complete API references(both requests and responses), complete input and outputs for a module(courtesy 'anonimas').

ACKNOWLEDGMENTS

First and foremost I would like to thank my family for their support, without them I wouldn’t be where I am today. Then I wish to thank my supervisor Dr William D. Tucker for his kind supervision, advices and support.

Table of contents

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

ARV–AntiRetroViral

iDART – Intelligent Dispensing of ART

Dashboard – A reporting tool that presents key indicators on a single screen, which includes measurements, metrics, and scorecards.

Data mart - It is a simple form of a data warehouse that is focused on a single functional area.

ETL - Extract, Transform, and Load is a process in database usage.

GUI - Graphical User Interface

HIV – Human Immunodeficiency Virus

IDE - Integrated Development Environment is a software application that provides comprehensive facilities to computer programmers for software development.

KPI – Key Performance Indicators

OLAP– Online Analytical Processing

OLTP– Online Transactional processing

OOA – Object Oriented Analysis

OOD – Object Oriented Design

Packages-

Pentaho – The Pentaho BI Project is open source application software for enterprise reporting, analysis, dashboard, data mining, workflow and ETL capabilities for business intelligence needs.

PostgreSQL– PostgreSQL, often simply Postgres, is an object-relational database management system (ORDBMS).

RA – Requirement Analysis

Representation Term- is a word, or a combination of words, that semantically represent the data type (value domain) of a data element.

Star schema - is the simplest style of data warehouse schema.

Talend - is an open source data integration software vendor which produces several enterprise software products, including Talend Open Studio.

UIS - User Interface Specification

UR – User Requirements

Chapter 1

INTRODUCTION

Any online transaction processing (OLTP) data contains information that can help in making informed decisions about businesses. For example, one can calculate your net profits for last quarter and compare them with the same quarter of the previous year. The process of analyzing data for that type of information, and the data that results, are collectively called *business intelligence*. Because most operational databases are designed to store data, not to help analyze it, it's expensive and time consuming to extract business intelligence information from databases. The solution: an online analytical processing (OLAP) database, a specialized database designed to help extract business intelligence information from data.

In response to a request from the Desmond Tutu HIV Foundation to assist the management of anti retro-viral (ARV) dispensing, the Intelligent Dispensing of Anti-Retroviral Treatment (iDART) system was developed by Cell-life which in 2009 is in over 20 clinics dispensing drugs to more than 45,000 patients. This system is used by pharmacists to manage the supply of ARV stocks, print reports and manage collection of drugs by patients. One of many iDART sites is the ARV pharmacy at the Tsepong Wellness Centre which became the third Elton Aids Foundation sponsored health care facility to receive the iDART system. The Tsepong Wellness Centre is currently servicing over 6000 HIV+ patients.

The goal of this project is to combine data from multiple instances of iDART into a single data mart that can be used for reporting and analysis by Cell-life (see Figure 1). A data mart is a simple form of a data warehouse that is focused on a single functional area. A data warehouse incorporates information about many subject areas, often the entire enterprise/organisation while the data mart focuses on one or more subject areas. The data mart represents only a portion of an enterprise's data, perhaps data related to a business unit or work group. Data marts represent the retail level of the data warehouse, where data is accessed directly by end users.[3]

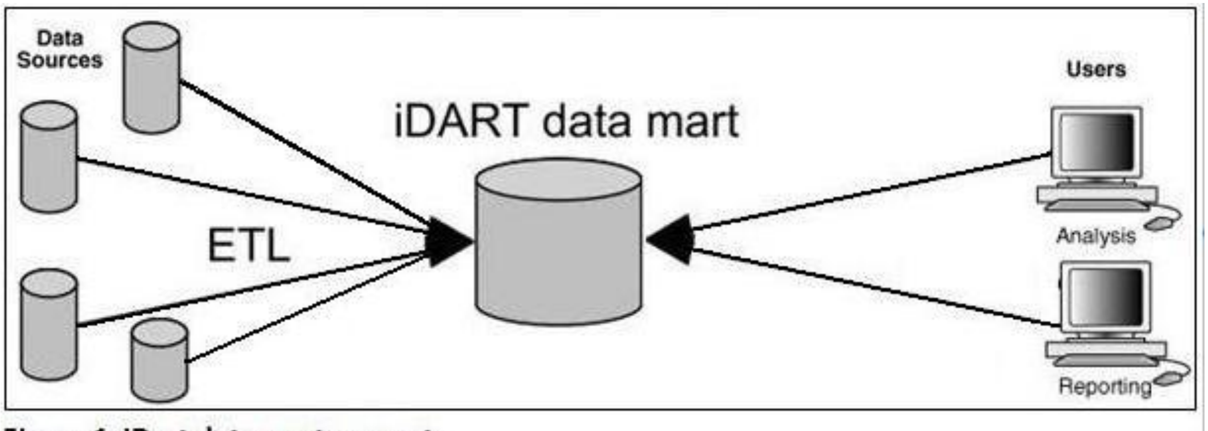


Figure 1: iDart data mart concept.

A schema is a collection of database objects, including tables, views, indexes and synonyms. Concerning the data mart design, two commonly used schemas are the star and snowflake schema. In star schema the fact is denormalised, all dimension tables are normalised and there will be primary foreignkey relationship between fact and dimension tables. For better performance we use star schema when compare to snow flake schema where fact table and dimension tables are normalised. Every dimension table there will be a look table meaning that we have to dig from top to bottom in the snowflake schema. The main advantages in star schema are that they:

- Provide a direct and intuitive mapping between the business entities being analyzed by end users and the schema design.
- Provide highly optimized performance for typical start queries
- A widely supported by a large number of business intelligence tools, which may anticipate or even require that the data mart schema contains dimension tables.

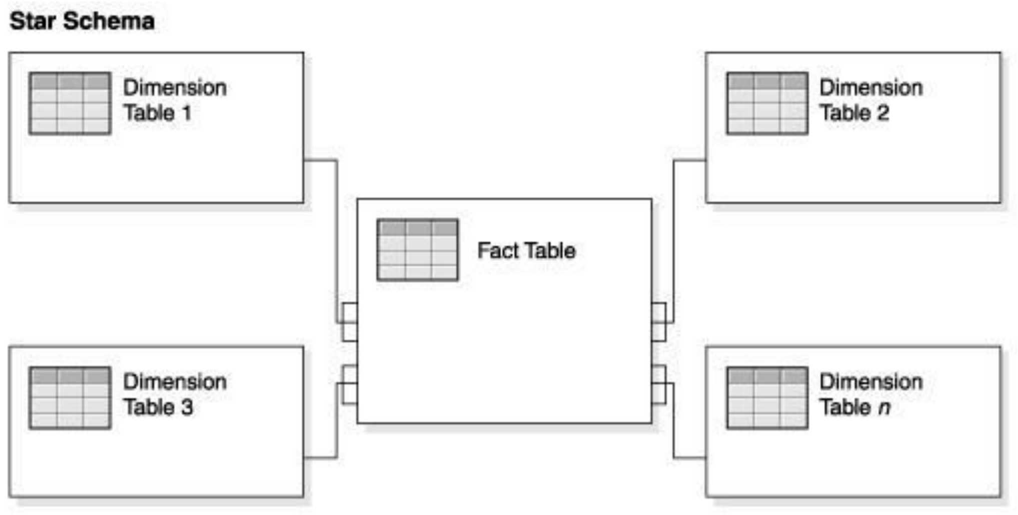


Figure 2: The Fact Table References Each Dimension Table.

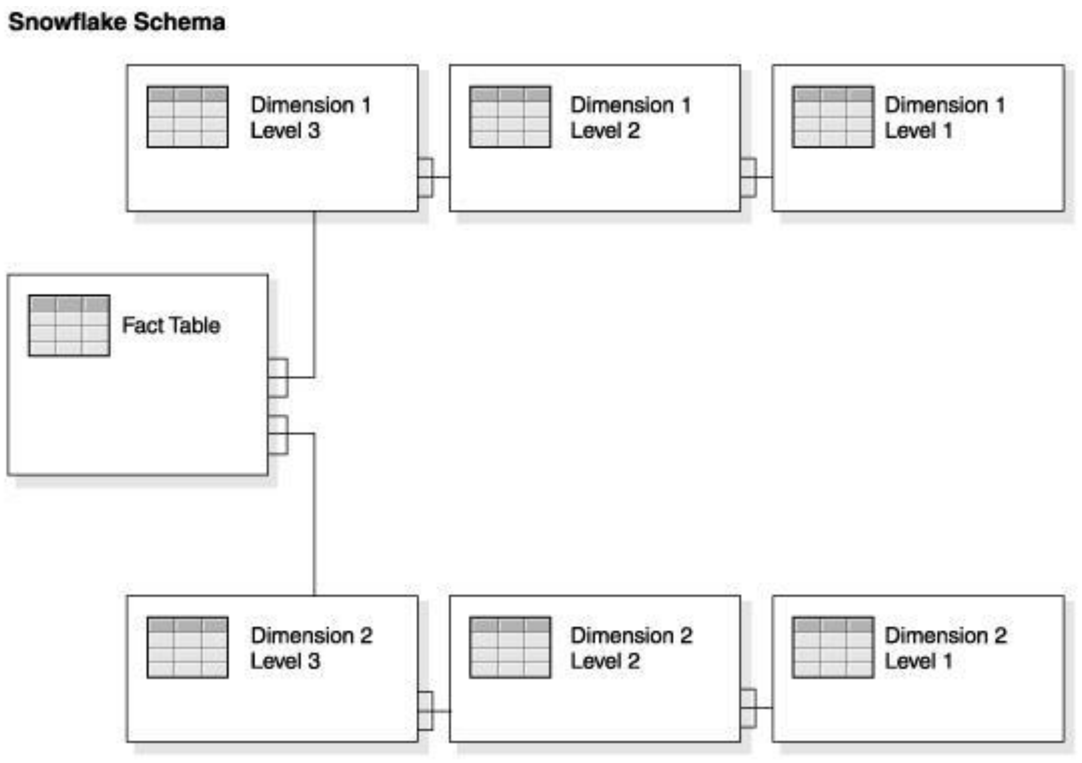


Figure 2: The Fact Table References a Dimension Table which may reference another Dimension Table.

This document is intended to guide development of iDART data mart. It also will give overview of the project, including why it was conceived, what it will do when complete. Screenshots showing how the final product will look like and behave are provided.
The object oriented view of the system is presented, analysis of the high level design and describes the objects needed to implement the system is provided.
This document also presents the object oriented design of the system, analysis of the low level design and provides details for the object oriented analysis of the system.

The rest of this document is organized as follows. Chapter 2 specifies the requirements the user expects from software solution to be constructed in this project.
Chapter 3 provides the user requirement analysis, Chapter 4 provides the user interface specification, Chapter 5 specifies the high level design, Chapter 6 the low level design.
Chapter 7 and Chapter 8

C h a p t e r   2

USER REQUIREMENTS

This chapter contains the user requirements of iDART data mart. These requirements have been derived from Cell-life’s project specification. This chapter is intended to guide development of iDART data mart. This also will give overview of the project, including why it was conceived, what it will do when complete, and the types of people we expect will use it. Section 2.1 identifies the user's view of the problem, section 2.2 tells what is expected from the software solution, section 2.3 tells what is not expected from the software solution and section 2.4 identifies general constraints for this data mart design.

2.1 User’s view of the problem

The time and expense involved in retrieving answers from databases means that a lot of business intelligence information often goes unused. Some organizations use a dozen different software packages to produce simple reports. If the report doesn't have the proper information, its creators have to start over. Also, the cost of implementing a full Data warehouse is higher than that of implementing a data mart. The iDART data mart will help minimize cost of extracting business intelligence information from iDART instances around the country.

2.2 What is expected from a software solution?

The software system is expected to provide easy access to frequently needed data and creates a collective view by a group of users.

Cell-Life expects a software solution that can be used for analysis and reporting purposes.

Cell-life would like to be able to generate the following statistics on a monthly/annual basis:

- Number of patients treated(based on packages created)
- Number of patients enroll on treatment
- Number of patients terminating treatment(including reason for termination)

  by date, site, gender and age groups (see Appendix A).

2.3 What is not expected from a software solution?

The software solution is not expected to be deployed to all the Cell-Life branches and it is not expected to be able to function in times of power failure unless a backup power supply is in place.

Also the software solution is not expected to be used by multiple business units except what it’s designed for.

2.4 General Constraints

We will work under a few number of constraints such as development environment which in this case has to be the integrated development environment (IDE). Also the database we’ll have to use is PostgreSQL, to make sure that our product (iDART data mart) is compatible with existing database which is currently in use.

C h a p t e r   3

REQUIREMENT ANALYSIS

Requirements analysis is critical to the success of a development project. [2] Requirements must be documented, actionable, measurable, testable, related to identified business needs, and defined to a level of detail sufficient for system design. Requirements can be functional and non-functional. Section 3.1 identifies the designer's interpretation of the user's requirements, Section 3.2 describes suggested the software solution and Section 3.3 identifies types of testing strategies to be used when testing the suggested software solution.

3.1 Designer's interpretation of the user's requirements

Cell-Life has clearly expressed the requirements for the iDART data mart in the previous chapter (Chapter 1). Now we will focus on the business and technical requirements needed to implement the given user requirements. Existing solutions will also be considered.

A basic desktop computer running Windows/Linux will work and a PostgreSQL Database Management System with Java. For data integration, any data integration software tool with Extract, transform and load (ETL) functionality can used. A Business intelligence (BI) Server that will provide common functions of business intelligence technologies like reporting, online analytical processing, analytics, data mining, business performance management, benchmarking, text mining, and predictive analytics. Any BI Server will work.

The basic building block to use in data mart design is the star schema because of the advantages this schema has. A star schema consist of one large central table called *fact table,* and a number of smaller tables called *dimension tables* which radiate out from the *fact table.*

After classifying data from the requirements in Chapter 1 and looking at the representation terms, facts and dimensions are as follows:

- Date, location/site and patient are dimensions
- Number of patient treated, enrolled for treatment, terminating treatment are facts.

3.2 Suggested solution

The suggested solution will make use of a desktop personal computer (PC) running Windows/Linux and can be broken down into various parts.
The first stage uses Extract, transform and load (ETL) tool to retrieve data from stand alone iDART databases to the iDART data mart. Second stage is accessing data in the data mart, analyzing it, creating reports, graphs, and charts using a dashboard.

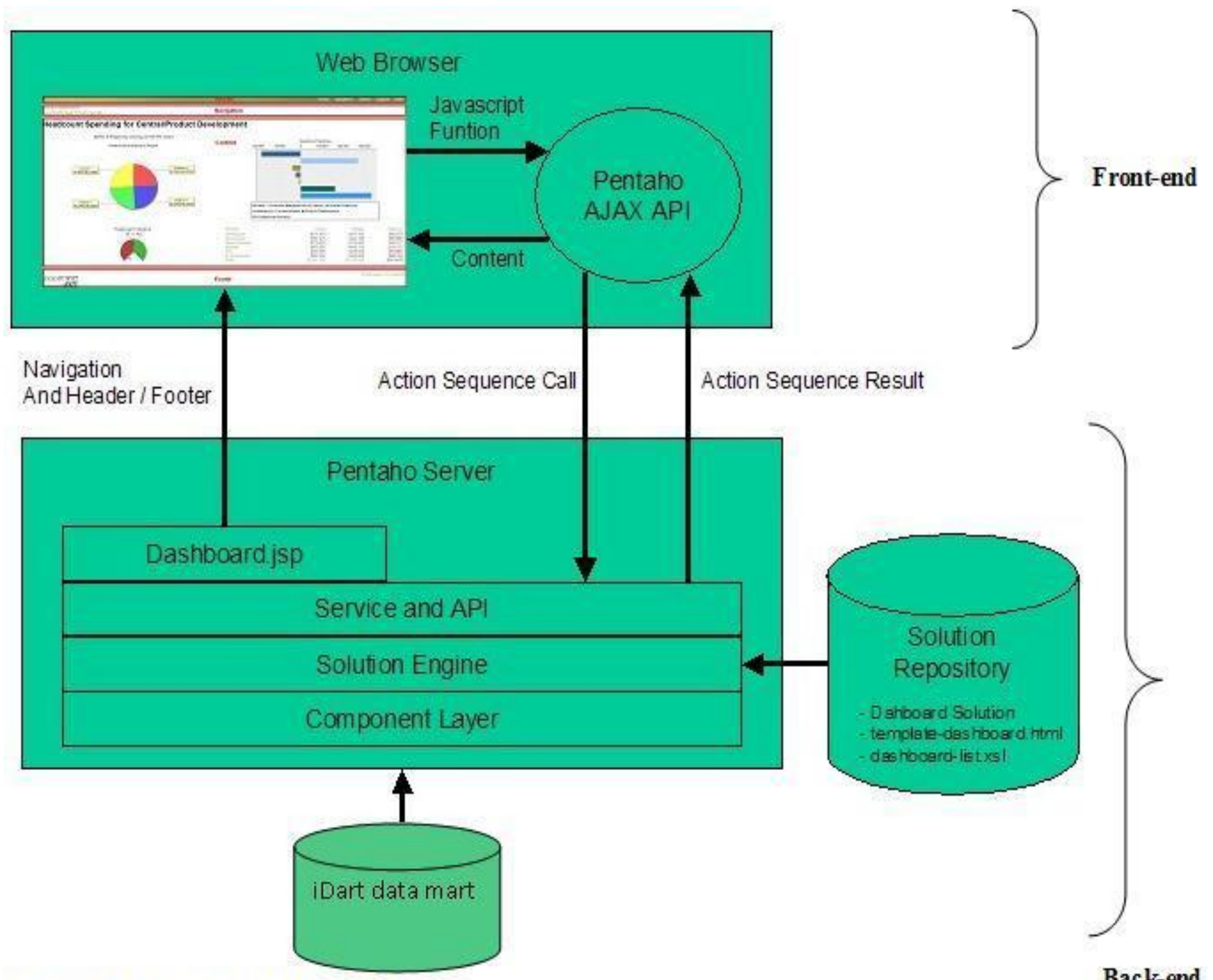


Figure 2: An overview of the system.

3.3 Testing the suggested solution

There are many different approaches to test software. For this project, functional and usability testing will be performed.

1. Functional Testing:

This is a new system and critical, so I must ensure its functional quality. All the features will be tested to ensure all functions provide the expected output.

2. Usability Testing:

Usability testing of this system will evaluate the potential for errors and difficulties involved in using the system for Cell-Life related activities.

C h a p t e r   4

USER INTERFACE SPECIFICATION

The purpose of this chapter is to provide a detailed specification of the iDART DATA MART user interface. These requirements will detail the outwardly observable behavior of the program. The user interface provides the means for the user, to interact with the program. This User Interface Specification is intended to convey the general idea for the user interface design and the operational concept for the software. Many details have been omitted for both clarity and because they have not been addressed yet. This document will be updated with additional detail as our analysis and design activities progress.

Section 4.1 gives a description of the complete user interface, Section 4.2 shows what the user interface looks like to the user, Section 4.3 tells how the interface behaves and Section 4.4 tells how the user interacts with the system.

4.1 Description of the complete user interface

The User Interface Specification (UIS) consists of one main graphical user interface (GUI), which consists with different operations enlisted in the options.

4.2 What the user interface looks like to the user

The Login page consists of two text boxes, namely Username and Password, and a Login command button allowing the users to log into the system. The login page helps the users to login as a user who visualizes and analyze data contained in the database, and as an Administrator (someone from the IT department) whose duty is to update, edit and modify the dashboard.

Once logged on, the user is presented with the dashboard.
Figure 3 shows the complete User Interface Specification (UIS). This is what a simple typical dashboard for any organization would look like.

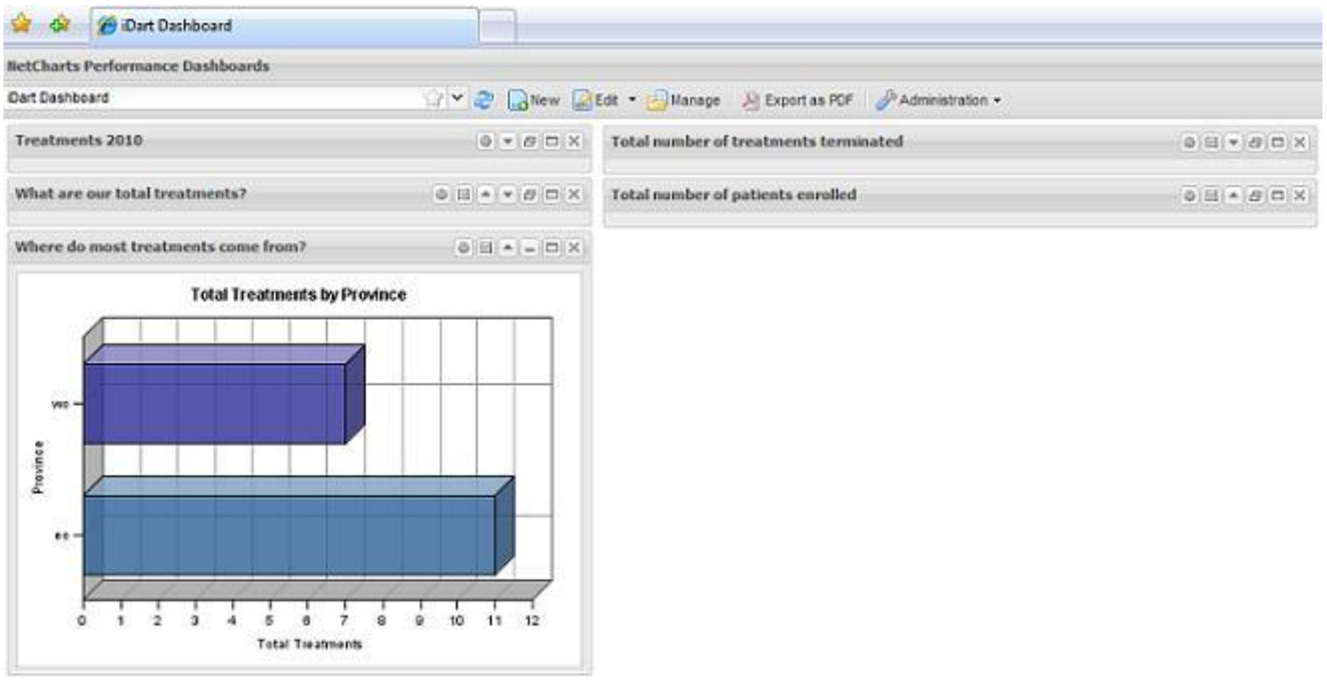

iDart Dashboard
NetCharts Performance Dashboards
iDart Dashboard
New
Edit
Manage
Export as PDF
Administration
Treatments 2010
Total number of treatments terminated
What are our total treatments?
Total number of patients enrolled
Where do most treatments come from?
Total Treatments by Province
Province
wc
ec
0
1
2
3
4
5
6
7
8
9
10
11
12
Total Treatments

Figure 3: User Interface Specification (UIS).

4.3 How the user interface behaves

How the dashboard interface behaves during manipulation is interesting. Each Key Performance Indicator (KPI) on a page is contained with a portlet featuring up to 7 controls in the upper right corner (Figure 4) used telling the object how to move, resize or do anything else according to a certain user input.

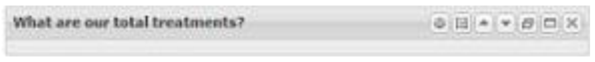


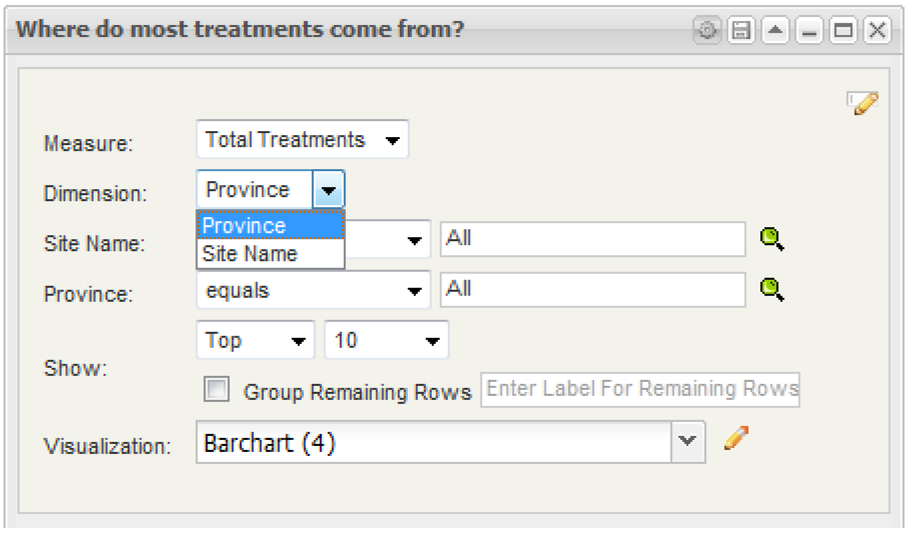
Figure 4: Example of KPI

With these controls, the KPI can be deleted from the page, enlarged, repositioned over the one above it and so on. Such behavior provides the user will full control of how data represented appears in the dashboard.

4.4How the user interacts with the system

A dashboard report is an important tool for any C-level executive and other business manager. While keeping them on top of vital statistics and KPIs, dashboard reports help them visualize and track trends on every level of the business and to align activities with key goals. The user interface enables users to visualize and analyze data stored in the data mart database. The interface will enable users to choose what data they want to view (*measures*) and how they want to view it (*dimensions*). Figure 5: illustrates how this is achieved:

Consider a scenario where a user wants to see the total number of patients treated province/site name. By clicking the Open Preference menu icon on the **Where do most treatments come from?** KPI, third in the left column Figure 4 will be shown.



Where do most treatments come from?

Measure: Total Treatments

Dimension: Province

Site Name: All

Province: equals All

Show: Top 10

☐ Group Remaining Rows Enter Label For Remaining Rows

Visualization: Barchart (4)

Figure 5:

If the user chooses to view number of treatments by province the output would be as shown in Figure 6.

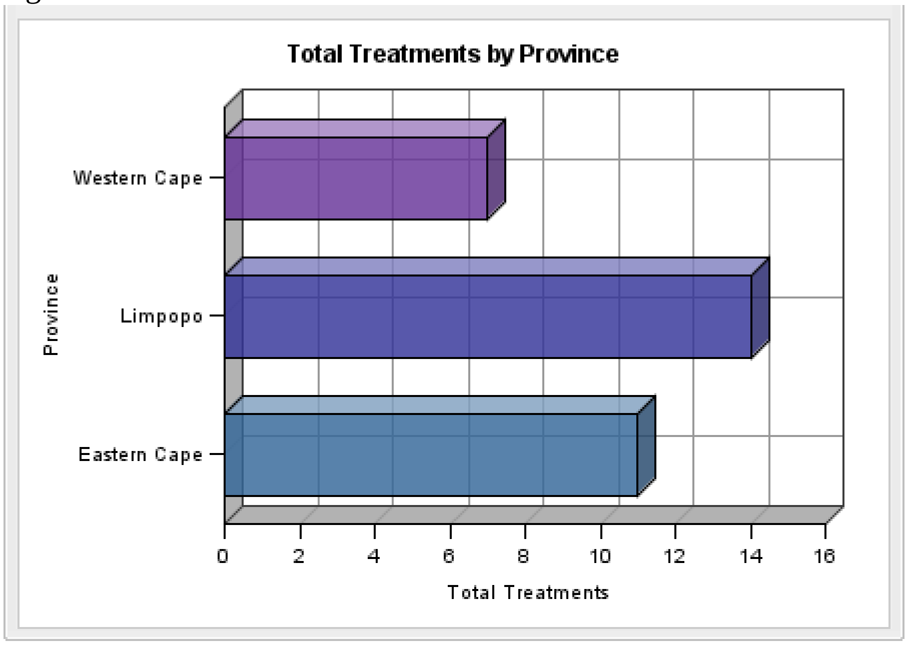


Figure 6:KPI example 1.

Where as if the user chooses to view number of treatments by site name the output would be as shown in Figure 7.

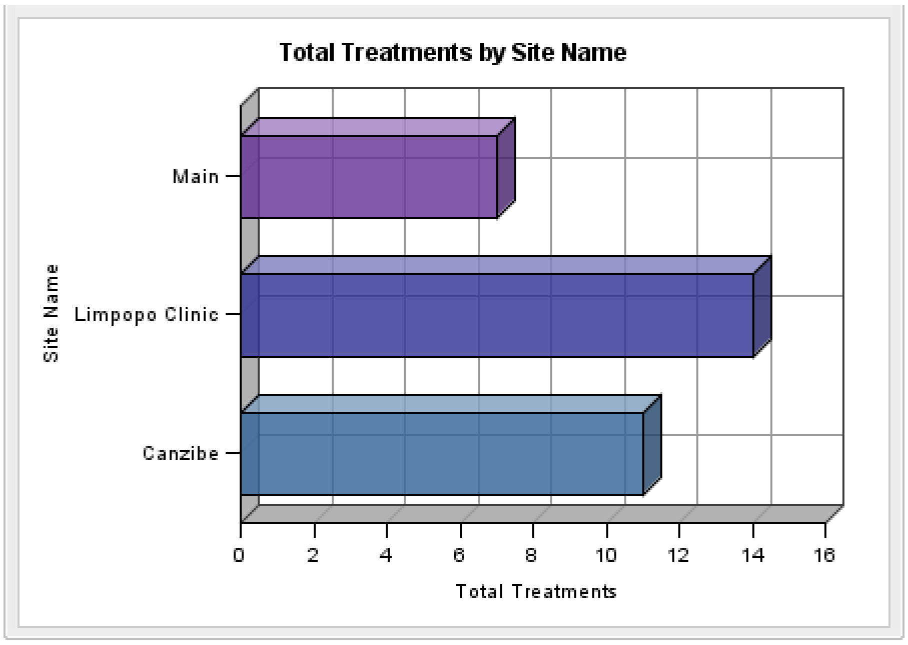


Figure 7: KPI example 2.

Chapter 5

HIGH LEVEL DESIGN

This chapter presents the object oriented view of the system, analysis of the high level design and describes the objects needed to implement the system. Each one of these objects is described and documented, and a data dictionary providing details of each object is provided.

5.1 Components

Component name	**Component description**
Talend Open Studio	Talend Open Studio is an open source data integration product designed to combine, convert and update data in various locations across a business.
Pentaho BI Server	The BI Server is an enterprise-class Business Intelligence (BI) platform that supports Pentaho's end-user reporting, analysis, and dashboard capabilities.
Pentaho Dashboard Designer	Pentaho Dashboard Designer is within the Pentaho User Console. Self-service dashboard designer that lets business users easily create personalized dashboards with little to zero training

Table 1: Objects required.

5.2 User interface design (Use Case Diagram)

Optimized User Interface Design requires a systematic approach to the design process. The importance of good User Interface Design can be the difference between system acceptance and rejection in the marketplace. If end-users feel it is not easy to learn, not easy to use, an otherwise excellent product could fail. Good User Interface Design can make a product easy to understand and use, which results in greater user acceptance. The use case diagram below shows some functional activities of the system that a user can perform.

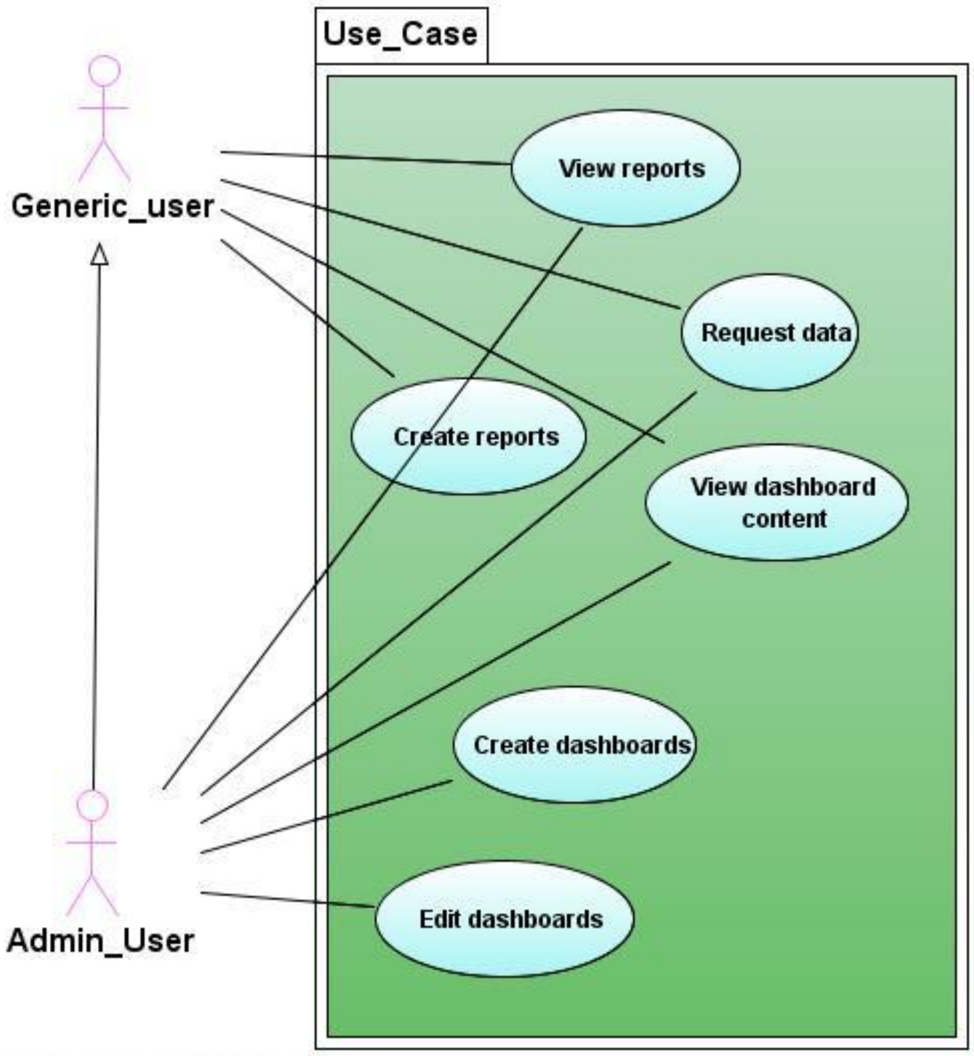


Figure 6: Use case diagram.

The above use case diagram illustrates that a generic user requests data from the data mart by dimension, creates and view reports and can view dashboards and that an administrator has its own behavior but also have the behavior of the generic user. The benefits of generalization eliminates duplicate behavior and attributes that will ultimately make the system more understandable and flexible.

5.3 Use case index

Use Case	Use Case Name	Primary Actor	Scope	Complexity
1	Request Data	Generic User	In	Low
2	Get Status	Generic User	In	Low
3	Create Reports	Generic User	In	Mid
4	Edit Dashboard	Administrator	In	High
5	Edit Dashboard Content	Administrator	In	High

Table 2: Use case index table.

5.4 Class Diagram

Structure diagrams are useful throughout the software lifecycle. Here we've used class diagrams to design and document the system's soon-to-be-coded classes. The purpose of the class diagrams is to show the types being modeled within the system. These types include:

- a class
- an interface
- a data type
- a component

Due to the nature of this project, we have a few number of classes and the reason for this is the fact that java script is mainly used. Figure 7 shows a more detailed description of the class diagrams.

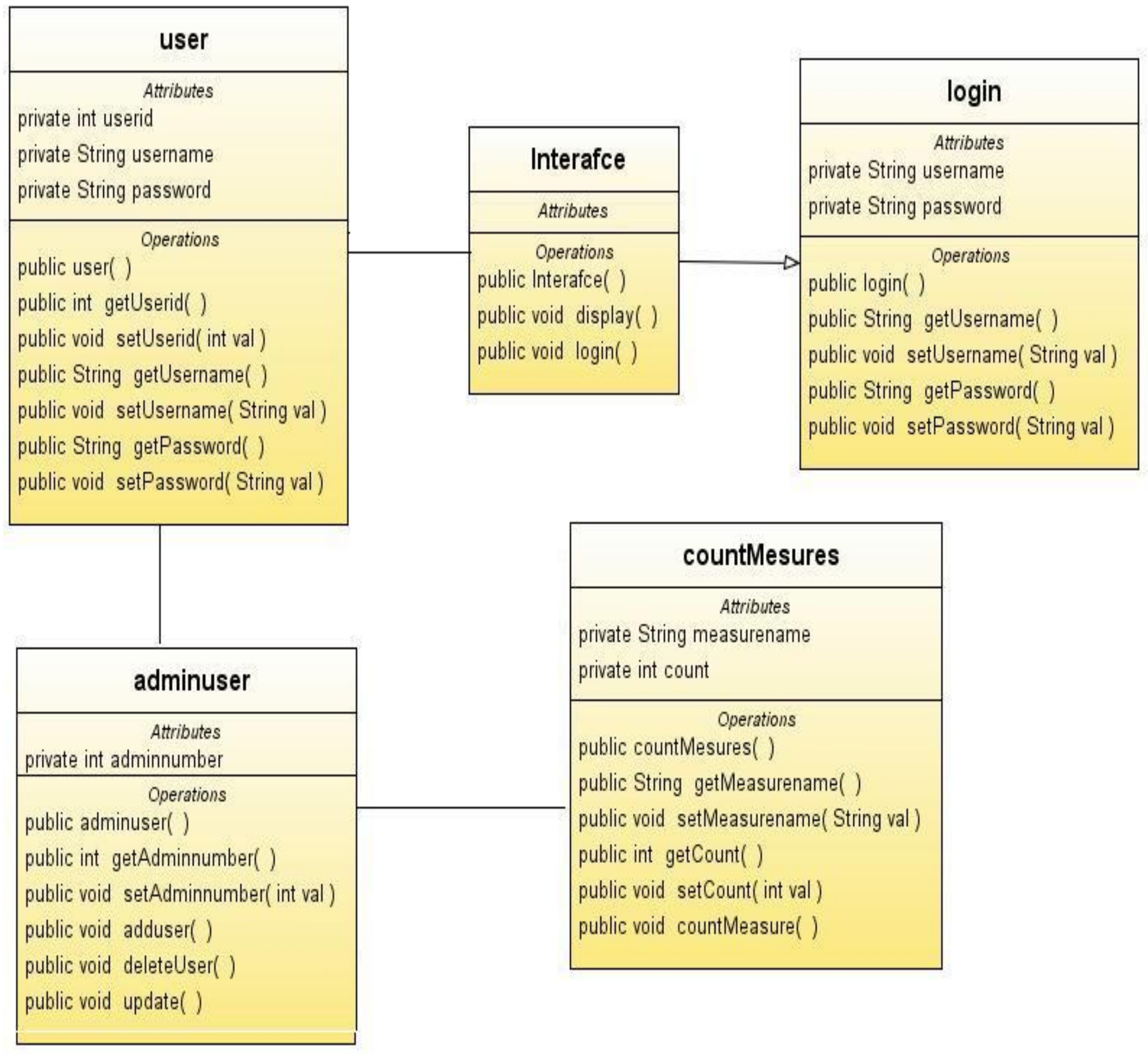
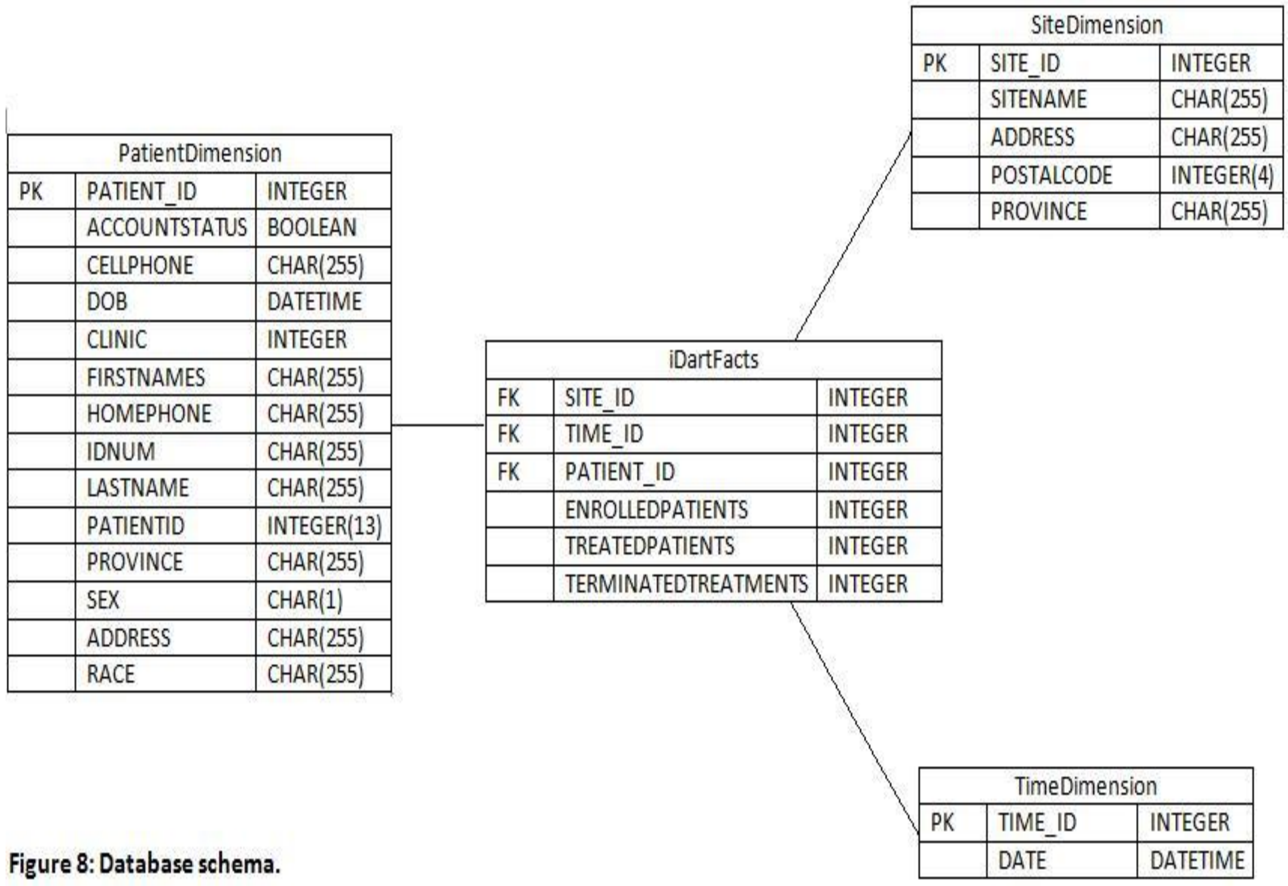

user
Attributes
private int userid
private String username
private String password
Operations
public user( )
public int getUserid( )
public void setUserid(int val)
public String getUsername( )
public void setUsername(String val)
public String getPassword( )
public void setPassword(String val)
Interafce
Attributes
Operations
public Interafce( )
public void display( )
public void login( )
login
Attributes
private String username
private String password
Operations
public login( )
public String getUsername( )
public void setUsername(String val)
public String getPassword( )
public void setPassword(String val)
adminuser
Attributes
private int adminnumber
Operations
public adminuser( )
public int getAdminnumber( )
public void setAdminnumber(int val)
public void adduser( )
public void deleteUser( )
public void update( )
countMesures
Attributes
private String measurename
private int count
Operations
public countMesures( )
public String getMeasurename( )
public void setMeasurename(String val)
public int getCount( )
public void setCount(int val)
public void countMeasure( )

Figure 7. Class diagrams.

5.5 Data mart schema

The figure below (Figure 8) shows the data mart schema for the proposed system. It consist of the three dimension tables and one fact table. The three dimension tables are PatientDimension, SiteDimension and TimeDimension. Each of these tables contains a number of fields and a description of data types.



PatientDimension

PK	PATIENT_ID	INTEGER
	ACCOUNTSTATUS	BOOLEAN
	CELLPHONE	CHAR(255)
	DOB	DATETIME
	CLINIC	INTEGER
	FIRSTNAMES	CHAR(255)
	HOMEPHONE	CHAR(255)
	IDNUM	CHAR(255)
	LASTNAME	CHAR(255)
	PATIENTID	INTEGER(13)
	PROVINCE	CHAR(255)
	SEX	CHAR(1)
	ADDRESS	CHAR(255)
	RACE	CHAR(255)

iDartFacts

FK	SITE_ID	INTEGER
FK	TIME_ID	INTEGER
FK	PATIENT_ID	INTEGER
	ENROLLEDPATIENTS	INTEGER
	TREATEDPATIENTS	INTEGER
	TERMINATEDTREATMENTS	INTEGER

SiteDimension

PK	SITE_ID	INTEGER
	SITENAME	CHAR(255)
	ADDRESS	CHAR(255)
	POSTALCODE	INTEGER(4)
	PROVINCE	CHAR(255)

TimeDimension

PK	TIME_ID	INTEGER
	DATE	DATETIME

Figure 8: Database schema.

C h a p t e r   6

LOW LEVEL DESIGN

This chapter presents the object oriented design of the system, analysis of the low level design and provides details for the object oriented analysis of the system.

6.1 Details of class attributes

Class	Attributes
User	**Int Userid**- uniquely identifies the user **String Username**- stores the username of the user **String password-** stores the user password
adminuser	**Int adminnumber**- uniquely identifies the admin user
countMeasures	**String measurename-** stores the name of the mesure. **Int count**- stores the number of measures
login	**String Username**- stores the username of the user **String password-** stores the user password

Table 3. A description of attributes of each class.

6.2 Details of class methods

Class	Function
User	**Public int setUserid()-** sets the userid **Public void setUsername()-** sets the username of the user. **Public void setPassword()-** sets the user password **Public int getUserid()-** returns the user id when invoked. **Public int getUsername()-** returns the user name when invoked. **Public int getPassword()-** returns the user password when invoked.

adminuser	**Public int setAdminnumber()-** sets the admin user number **Public int getAdminnumber ()-** returns the admin user number when invoked. **Public void adduser ()-** adds a new user when invoked. **Public deleteUser ()-** deletes a specified user when invoked.
countMeasure	**Public void setMesurename()-** sets the measure name. **Public void setCount()-** sets the count **Public int getMesurename ()-** returns the measure name. **Public int getCount()-** returns the measure count **Public void countMeasure ()-** returns the actual value of the specified when invoked.
login	**Public void setUsername()-** sets the username of the user. **Public void setPassword()-** sets the user password **Public int getUsername()-** returns the user name when invoked. **Public int getPassword()-** returns the user

Table 5. A description of methods/functions of each class.

6.3 Pseudo code

```
public class User {

      public int Userid;
      public String password;
      public String password;


      /**
       * Constructor for User.
       * @param Userid int
       * @param Username String
       * @param password String
       */

      /**
       * Method getUserid.
```

```
         * @return int
         */
        public int getUserid() {
              return Userid;
        }

        /**
         * Method setUserid.
         * @param Userid int
         */
        public void setUserid(String Userid) {
              this.Userid = Userid;
        }

        /**
         * Method getUsername.
         * @return String
         */
        public String getUsername() {
              return Username;
        }

        /**
         * Method setUsername.
         * @param Username String
         */
        public void setUsername (String Username) {
              this. Username = Username;
        }


        /**
         * Method getPassword.
         * @return String
         */
        public String getPassword() {
              return Password;
        }

        /**
         * Method setPassword.
         * @param Password String
         */
        public void setPassword (String password) {
              this.password = password;
        }

  }


  public class admin_user extends User {

        public int adminnumber;



        /**
         * Constructor for admin_user.
         * @param adminnumber int
         */
```

```
      /**
       * Method getAdminnumber.
       * @return int
       */
      public int getAdminnumber () {
            return adminnumber;
      }

      /**
       * Method setUserid.
       * @param adminnumber int
       */
      public void setAdminnumber (String adminnumber) {
            this.adminnumber = adminnumber;
      }

public void adduser()         // delete & update are similar
{
  String name, password;
  Int userid;
  Connection       db;        // A connection to the database
  Statement        sql;       // Our statement to run queries with
  DatabaseMetaData dbmd;      // This is basically info the driver delivers
                              // about the DB it just connected to.


    Class.forName("org.postgresql.Driver"); //load the driver
    db = DriverManager.getConnection("jdbc:postgresql:"+database,
                                     username,
                                     password); //connect to the db
    dbmd = db.getMetaData(); //get MetaData to confirm connection
    System.out.println("Connection to "+dbmd.getDatabaseProductName()+" "+
                       dbmd.getDatabaseProductVersion()+" successful.\n");
    sql = db.createStatement(); //create a statement that we can use later


    String sqlText = "insert into usertable values (name,userid,password
etc)";

    sql.executeUpdate(sqlText);

      . . . . .
      . . . . . . //some exception handling code for invalid password, etc.

}


}

Public class countMeasure{

import java.sql.*;   // Everything we need for JDBC
import java.text.*;
import java.io.*;

public void countMeasure()
{
  Int measure;
  Connection       db;        // A connection to the database
  Statement        sql;       // Our statement to run queries with
  DatabaseMetaData dbmd;      // This is basically info the driver delivers
```

```
                             // about the DB it just connected to.


    Class.forName("org.postgresql.Driver"); //load the driver
    db = DriverManager.getConnection("jdbc:postgresql:"+database,
                                     username,
                                     password); //connect to the db
    dbmd = db.getMetaData(); //get MetaData to confirm connection
    System.out.println("Connection to "+dbmd.getDatabaseProductName()+" "+
                       dbmd.getDatabaseProductVersion()+" successful.\n");
    sql = db.createStatement(); //create a statement that we can use later

    // Here will be a code that will actually count each of the measures
       This is tricky since on our data sources these measures aren't
       Counted.
    String sqlText = "";

    sql.executeUpdate(sqlText);
    measure = sql.getUpdateCount();
}
```

C h a p t e r   7

IMPLEMENTATION

This chapter provides the major steps involved in implementing a data mart. These steps are to design the schema, construct the physical storage, populate the data mart with data from source databases and accessing data from data mart.
Section 3.1 is the design step, Section 3.2 describes the construction step, Section 3.3 describes the populating step and Section 3.4 describes the access step.

For IDART DATA MART implementation the following Business Intelligent (BI) technologies will be used:

- PostgreSQL 8.3
- Pentaho Business Intelligent suite Community Edirion 3.6.0
- Talend Open Studio 4.0.1

There are no restrictions, any BI technologies can be used.

3.1 Design step

Design step is the first step of the data mart process. Design step covers all of the tasks from initiating the request of for a data mart through gathering user requirements (Chapter 2), analyzing user requirements (Chapter 3) and developing the logical and physical design of the data mart.
This step consists of the following tasks:

- Getting business and technical requirements
- Identification of data sources
- Choosing an appropriate data subset
- Designing logical and physical structure of the data mart

Chapters 2 & 3 covered all these tasks.

3.2 Construction step

This step includes creating the physical database and the logical structures associated with the data mart to provide fast and efficient access to the data. This step consists of the following tasks:

- Creating the physical database and storage structures like tablespaces associated with the data mart.
- Creating schema objects
- Determining how best to set up the tables and access structures

An SQL script to create the physical database is included in the source code pack that contains all the source files for IDART DATA MART.
Here’s a partial content of this file.

```
-- PostgreSQL database dump
--

drop database if exists sampledata;
```

```
CREATE DATABASE sampledata  WITH OWNER = postgres  ENCODING = 'UTF8'
TABLESPACE = pg_default;

\connect sampledata postgres

SET statement_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = off;
SET check_function_bodies = false;
SET client_min_messages = warning;
SET escape_string_warning = off;

SET search_path = public, pg_catalog;

SET default_tablespace = '';

SET default_with_oids = false;

--
-- Name: clinic; Type: TABLE; Schema: public; Owner: postgres; Tablespace:
--

CREATE TABLE clinic
(
  id integer NOT NULL,
  address1 character varying(255),
  address2 character varying(255),
  notes character varying(255),
  postalcode character varying(255),
  province character varying(255),
  telephone character varying(255),
  mainclinic boolean,
  clinicname character varying(255),
  city character varying(255),
  modified character(1),
  CONSTRAINT clinic_pkey PRIMARY KEY (id),
  CONSTRAINT unique_clinicname UNIQUE (clinicname)
);

ALTER TABLE public.clinic OWNER TO postgres;
.
.
.
.
```

This script creates all a database named sampledata and grants access to users that will use the data mart. This database has six tables named clinic, patient, episode, package, idartfact and time. A step-by-step guide on how to run an SQL script is included on the user guide which is on the last chapter.

Next a star schema to be used for analysis view is created. This schema has for dimensions namely:

- Site/Location
- Time
- Gender
- Age group

Here's an xml schema file.

```
<?xml version="1"?>
<Schema name="iDARTSchema">
<!-- Shared dimensions -->

  <Dimension name="Site">
    <Hierarchy hasAll="true" allMemberName="All Sites">
      <Table name="CLINIC"/>
      <Level name="Site" column="CLINICNAME" uniqueMembers="true"/>
    </Hierarchy>
  </Dimension>

  <Dimension name="Time" foreignKey="TIME_ID" >
    <Hierarchy hasAll="true" allMemberName="All Years" primaryKey="TIME_ID">
    <Table name="DIM_TIME"></Table>
      <Level name="Years" column="YEAR_ID" type="String" uniqueMembers="true"/>
      <Level name="Quarters" column="QTR_NAME" type="String" uniqueMembers="true"/>
      <Level name="Months" column="MONTH_NAME" type="String"
         uniqueMembers="true"/>
    </Hierarchy>
  </Dimension>

  <Dimension name="Gender">
    <Hierarchy hasAll="true" allMemberName="All Genders">
      <Table name="PATIENT"/>
      <Level name="Gender" column="SEX" uniqueMembers="true"/>
    </Hierarchy>
  </Dimension>

  <Cube name="Treatment Analysis">
    <Table name="CLINIC"/>
    <DimensionUsage name="Site" source="Site"/>
    <DimensionUsage name="Time" source="Time" />
    <DimensionUsage name="Gender" source="Gender" />
    <Measure name="Treated patients" column="TREATED" aggregator="sum"
formatString="#,###"/>
    <Measure name="Enrolled patients" column="ENROLLED" aggregator="sum"
formatString="#,###"/>
```

```
  </Cube>

</Schema>
```

3.3 Populating step

This step includes all the tasks related to getting the data from the sources, modifying it to the right format and level of detail and moving it into the data mart. This step consists of the following tasks:

- Mapping data sources to target data structures
- Extracting data
- Loading extracted data into the data mart

On this step Talend Open Studio 4.0.1 is used. From this tool a job named iDART_ETL is created, then three database connections to the IDART instances from where data will be extracted are created. Another database connection is created to the data mart. After running the job the data will be extracted from the data sources through the database connections created to the data mart. A step-by-step guide on how to create an ETL job, how to run it and how to create database connections using Talend Open Studio 4.0.1 is included in the user guide which is on the last chapter.

3.4 Accessing step

This step involves putting the data in the data mart into use: query the data, analyzing it, creating reports, graphs, charts and publishing these. The end user uses a graphical front-end tool to submit queries to the database and display the results of the queries. This step consists of the following tasks:

- Setting up an intermediate layer for the front-end tool to use. This layer translates database structures and object names into business terms, this helps end users interact with the data mart using terms that are related to the business function.
- Manage and maintain business interfaces.
- To help queries submitted through the front-end tool execute quickly and efficiently, set up and manage database structures.

Pentaho Business Intelligent suite Community Edirion 3.6.0 will provide common functions of business intelligence technologies such as reporting, online analytical processing, analytics, etc. All the components (xaction and xml files) that will be used to provide and view data are included is included in the user guide which is on the last chapter.
The jsp script file named SampleDashboard is created.
The script in this jsp file controls the layout and content generation of the dashboard.

The above steps provide a roadmap to data mart design and implementation.

BIBLIOGRAPHY

[1] BATIN, C., SERI, S., AND NAVATE, S.B, (1994) *Conceptual Database Design: An Entity Relational Approach*, Redwood City, California

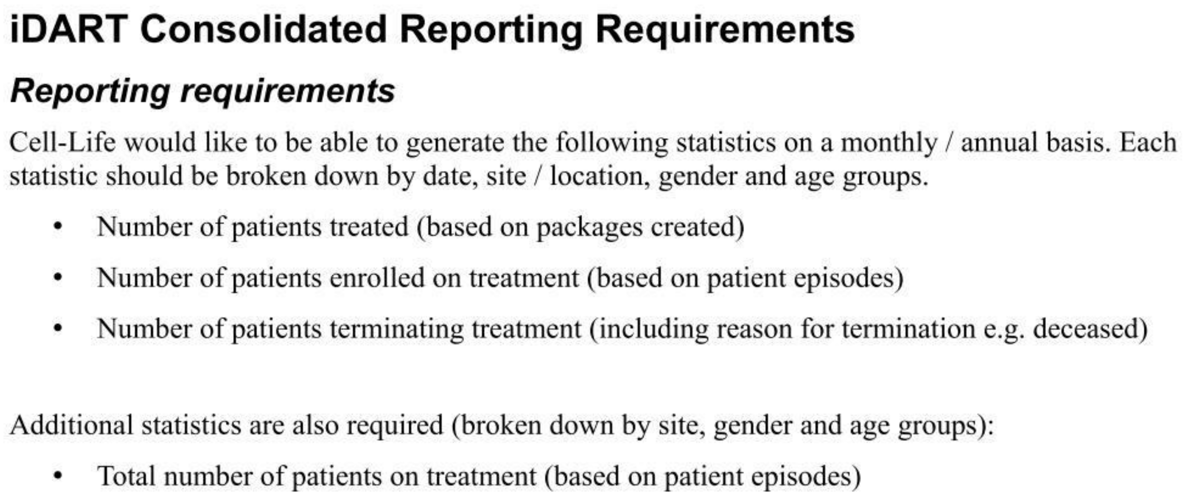
[2] Executive editors: Alain Abran, James W. Moore; editors Pierre Bourque, Robert Dupuis, ed (March 2005).

[3] InfoManagement Direct, November 1999. *Data Mart Does Not Equal Data Warehouse* [online]. Available http://www.information-management.com/infodirect/19991120/1675-1.html [accessed 7 March 2010]

[4] KIMBALL, R.,(1996): *The Data Warehouse Toolkit*, New York: J. Wiley & Sons.

[5] KIMBALL, R.,(1997): *DBMS Online*,A Dimensional Manifesto August, 1997.

APPENDIX A



iDART Consolidated Reporting Requirements

Reporting requirements

Cell-Life would like to be able to generate the following statistics on a monthly / annual basis. Each statistic should be broken down by date, site / location, gender and age groups.

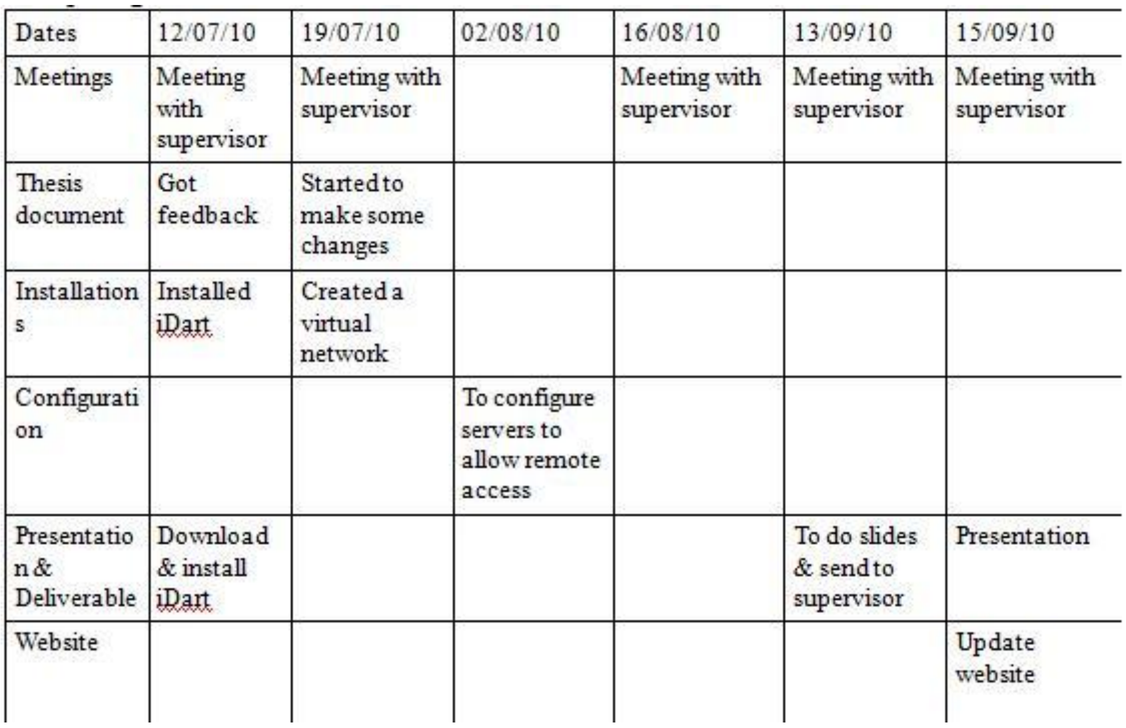
- Number of patients treated (based on packages created)
- Number of patients enrolled on treatment (based on patient episodes)
- Number of patients terminating treatment (including reason for termination e.g. deceased)

Additional statistics are also required (broken down by site, gender and age groups):

- Total number of patients on treatment (based on patient episodes)

APPENDIX B

TERM 3



Dates	12/07/10	19/07/10	02/08/10	16/08/10	13/09/10	15/09/10
Meetings	Meeting with supervisor	Meeting with supervisor		Meeting with supervisor	Meeting with supervisor	Meeting with supervisor
Thesis document	Got feedback	Started to make some changes				
Installations	Installed iDart	Created a virtual network				
Configuration			To configure servers to allow remote access			
Presentation & Deliverable	Download & install iDart				To do slides & send to supervisor	Presentation
Website						Update website