# Migration from Proprietary to Open Source Database for eHealth Domain

Shail Dubey Kumari Nita Sujeet Kumar

## ABSTRACT
Due to the varying economic climate, organizations running Oracle are struggling to meet rising information demands with constricted budgets. Postgres significantly reduces total cost of database ownership, without sacrificing performance and scalability by eliminating up-front perpetual license fees.
Why migrate at all? Either the current system has some functional flaws or its cost/benefit ratio does not appeal anymore to its owner. Three arguments are put forward in support of migrating to Open Source database management systems: Lowering Total Cost of Ownership (TCO), special features, and Open Source code.
C-DAC's Hospital Management Information System "e-Sushrut" is a complete ERP solution for Hospitals or a chain of Hospitals. Commonly used technology in e-Sushrut is J2EE and proprietary database Oracle as backend. Organization has to pay certain amount to get license of oracle. Day by Day demand of health care services increase and it is not possible for all client to afford the cost of oracle's license. So by migration of database from Oracle to Postgres an effort is made to reduce database license cost as Postgres Community Edition is available free of cost.

## Keywords
Migration, HIMS (Hospital Management Information System), e-Sushrut, Postgres (open source community edition), Electronic Medical Record

## 1. INTRODUCTION
This paper explains what is Postgres and how it can be used to enhance functionality of our existing project i.e. "e-Sushrut" and also justify the reasons for selecting Postgresql as Backend. This paper also explain the efforts for the conversion of database and steps needed for the conversion in backend and front end level. E-Sushrut incorporates an integrated clinical information system to improve hospital administration and patient health care. It also provides an accurate, Electronic Medical Record (EMR) of the patient. This paper provides a process model for customizing application to Open Source technologies. The path from the source system (a currently used closed source database management system) to the target system (an equivalent Open Source DBMS) will be mapped. This involves decomposing the database system into separate parts ("assets") which are to be analyzed, extracted (from the old system), converted and loaded (into the new system). The proposed customization workflow aims to be as generic as possible, to be easily adapted to real life scenarios. The specifics of customizing to Open Source technologies will be subject to discussion as well. An evaluation of five enterprise-level Open Source database management systems (Firebird, Ingres, MaxDB, MySQL, and PostgreSQL) with respect to features, performance and data warehouse capabilities provides the background.

## 2. REQUIREMENT OF USING OPEN SOURCE SOFTWARE
Open Source's proponents often claim that it offers significant benefits when compared to typical commercial products such as:

- Commercial products typically favor visible features (giving marketing advantage) over harder-to measure qualities such as stability, security and similar less glamorous attributes which is served by Open source this can be known as comparison between qualities vs. features.
- Three arguments are put forward in support of customizing to Open Source technologies: Lowering Total Cost of Ownership (TCO), special features, and Open Source code and community.
- Web Commerce Group also selected PostgreSQL for three Primary reasons :Total cost of ownership ,Technology, and Maintenance and support.[8]
- Total Cost of Ownership (TCO). Because PostgreSQL is open-source software licensed under BSD, it is free. The following cost components were taken into account when evaluating TCO for PostgreSQL and other DBMS products:
  - Acquisition. How much does it cost to acquire the DBMS?
  - Deployment and setup. How much does it cost to set up and deploy applications developed in the DBMS?
  - Hardware and software working in conjunction with the DBMS. What kinds of hardware and software are required to run the DBMS and how much do they cost? What kinds of software can the DBMS work with and what is the integration cost?
  - Maintenance and upgrades. How much does it cost to maintain an application developed in the DBMS and how much does it cost to upgrade?
  - Staff training. How much does it cost to hire and train people in the DBMS?
  - Removal of the DBMS from asset. How much does it cost to remove the DBMS in order to use another replacement product?
  - Technical support. How much does it cost to provide technical support for the DBMS? PostgreSQL was the clear choice based on Web Commerce Group's TCO analysis.
- **Technology**. PostgreSQL is a sophisticated object-relational DBMS that is a hybrid of relational and object-oriented DBMS. PostgreSQL provides a number of compelling, advanced technical features, including the features that guarantee ACID (Atomicity, Consistency, Isolation, and Durability), stored procedures, triggers, replication, transaction support as well as object-oriented features such as inheritance. Through the use of Multi-Version Concurrency Control (MVCC). Postgresql can optimally support heavy transactions load in a large multi-user environment in which there are many users trying to read and update data. A good example is a large

ecommerce website. MVCC also allows PostgreSQL to make a full database backup while the database is live, without interrupting end users by shutting down the database.

- **Maintenance and Support.** PostgreSQL has the same types of advantages as other popular open-source software such as Linux. New technologies and features tend to be implemented quickly and aggressively in open-source software. Since open-source development is an ongoing process involving potentially hundreds of talented developers, PostgreSQL can offer new features, architectures, and platform support as soon as the project can implement and support it. This is often long before standard commercial products can incorporate these innovations. Because the open-source development process is highly collaborative, the resulting code sees widespread peer review, improving the probability of turning out a solid and secure product. Unlike other open source software that lack commercial support, PosgreSQL has numerous, reputable companies, such as Red Hat (www.redhat.com), that provide support and documents. There are also a fair amount of documentations and books (free and purchasable), in addition to an active, responsive developer forum and network.
- Some other reason because of which we support open source are as follow:

1. Join a worldwide community, including direct contact to the system's developers (bug reporting, wish lists, and discussion), extensive documentation and support,
2. Gain better control on its software, making the IT Infrastructure work more effectively and more tailor-made,
3. Free itself from fear of software obsolescence and dependence on vendor viability and survival as the Open Source code remains and can be taken over by anybody willing to continue (see e.g. the continuation of the Mozilla browser even after the Mozilla Foundation decided to drop the project),
4. Enjoy a high level of security based on Open Source peer review, fast patching, as well as the database system not requiring administrative system permissions, and
5. Adapt and extend the system to the company's own needs by developing missing functionality themselves.

## 3. COMPONENT OF E-SUSHRUT LITE ARCHITECTURE

This paper documents provides a basic steps which adopt for the migration of JDBC applications from Oracle to Postgresql and before illustrating that want to share configuration:

Databases -

- Oracle 10.2
- PostgreSQL 9.2

Development Environment on Windows 7 -

- PostgreSQL JDBC driver - postgresql-8.4-404.jdbc3.jar
- JDK 1.5.0
- Apache 2.0.55
- Tomcat 6.0
- Connector: Apache Tomcat JK 1.2.15 for WIN32 – works with Apache 2.0.55 and later

The major features of the application architecture of HMIS include

- N-tier J2EE Internet Architecture
- Takes full advantage of extensive markup language (XML)
- RDBMS(Oracle) for easy retrieval and better performance
- Portable across a variety of platforms

## 4. EXERCISES PERFORM FOR THE MIGRATION OF DATABASE

Basic changes perform to port application from oracle to Postgres:

### 4.1 Changes at backend level:

#### 4.1.1 Schema changes

Schema: Oracle gives every user their own schema, by default but in Postgres we create a user and schema by the same name and the first component in search_path is $user, by default.
Identifiers: Names of schema, tables, columns, functions in Oracle converts them to UPPER CASE while in Postgres converts them to lower case, unless quoted
Tables In postgres we CREATE TABLE is mostly as oracle, except,

- For Global Temporary table it use LOCAL TEMP tables
- For Partition clauses it use Inheritance, Triggers, and check Constraints
- Remove Storage Clause

Constraints Primary Key, Foreign Key, Unique, CHECK, NOT NULL are all working same as oracle. [3]

#### 4.1.2 Data Types changes

- VARCHAR, VARCHAR2, NVARCHAR, NVARCHAR2 convert to VARCHAR or TEXT.
- CHAR, NCHAR convert to CHAR
- CLOB, LONG convert to VARCHAR or TEXT
- NUMBER can be convert to into
  - BIGINT, INT, SMALLINT, REAL, REAL, DOUBLE PRECISIO which give good performance but less control on scale
  - NUMERIC which give Unlimited size (implementation specific) but low performance.
- BINARY_INTEGER, BINARY_FLOAT convert to INTEGER,FLOAT
- Date into DATE or TIMESTAMP postgres also consider timezone effects i.e. TIMESTAMP WITH TIMEZONE.[3]

#### 4.1.3 Data Migration

Data can be migrated by following:

- Use GUI tools if data type conversion was smooth and no restriction on database size
- Use ETL style such as
  - Use custom application to export in plain-text, CSV
  - Use scripting (Perl!) for transforming
  - Use COPY FROM to load
  - Avoid WAL logging by creating/truncating the table in same transaction
  - Upside: Allows parallel loads
  - Downside: Requires custom development.
- sequences in oracle we extract it using sequence_name.nextval while in Postgres use setval('sequence_name', value).[3]

#### 4.1.4 General logics changes

- RETURN becomes RETURNS
- EXECUTE IMMEDIATE becomes EXECUTE
- SELECT without INTO becomes PERFORM where PERFORM has the same syntax as a full blown SELECT
- At last of function creation we must chose a language at last of function such as given syntax

CREATE OR REPLACE FUNCTION fn (a INOUT) RETURNS
INT AS $$DECLARE . BEGIN .. END; $$ LANGUAGE Lang;

- %TYPE, %ROWTYPE: works
- cursor_name%ROWTYPE: Doesn't work; Use RECORD
- REFCURSORS: No replacement Use Set-Returning-Functions.
- Ability to COMMIT/ROLLBACK within procedures (only) Because of bounded size of ROLLBACK SEGMENTs Postgres doesn't have rollback segments
- It use EXCEPTION handling by using SAVEPOINT Not quite the same thing.[3]

### 4.1.5 Procedures

- Postgres has only functions and use RETURNS VOID for procedure. During procedure conversion we may need some application changes since calling convention in connectors (JDBC, etc.) matters.
- Oracle PL/SQL conversion is a little difficult and the obvious PostgreSQL backend language in which to (re)write stored procedures is the similar procedural language PL/pgSQL To install PL/pgSQL, the superuser DBA should run,$ createlang -d db9 plpgsql # install 'Oracle PL/SQL like' language
- Example A procedure of our project to get counter and user in oracle changes into postgres function
- In oracle procedure is as given below:

```
PROCEDURE proc_get_counter_user(modval character varying DEFAULT '1'::character varying, hosp_code character varying DEFAULT NULL::character varying, ipaddress character varying DEFAULT NULL::character varying, seatid character varying DEFAULT NULL::character varying, OUT err character varying, OUT resultset ahis_type.refcursor) IS

QUERY VARCHAR2 (2000)
 BEGIN
 IF (modVal = 1)
 THEN
 QUERY:=' SELECT BILL_MST.GETCOUNTERDTL(0 , '''||ipAddress||''' , '||hosp_code||' , 4 , 0 ) ||''^''||
PKG_RPT_FUNC.GET_USER_NAME(to_char('||hosp_code||') , to_char('||seatid||')) FROM DUAL ';
 ELSE
 QUERY: =' SELECT
PKG_RPT_FUNC.GET_USER_NAME (to_char ('||hosp_code||'), to_char ('||seatid||')) FROM DUAL ';
END IF;
 OPEN resultset FOR QUERY;
 EXCEPTION
  WHEN OTHERS
       THEN
         err := SQLERRM;
         RAISE;
   END;
```

- In Postgres same procedure is create as function such as given below:

```
CREATE OR REPLACE FUNCTION ahis.pkg_bill_view_proc_get_counter_user(IN modval character varying DEFAULT '1'::character varying, IN hosp_code character varying DEFAULT NULL::character varying, IN ipaddress character varying DEFAULT NULL::character varying, IN seatid character varying DEFAULT NULL::character varying, OUT err character varying, OUT resultset refcursor) RETURNS record AS

$BODY$
Declare
QUERY VARCHAR (2000);
 BEGIN
 IF (modVal ='1')
THEN
QUERY:=' SELECT
coalesce(ahis.bill_mst_GETCOUNTERDTL(0 , '''||ipAddress||''' , '||hosp_code||' , 4 , 0 ),"-") ||''^''||
ahis.bill_mst_GET_USER_NAME(to_char('||hosp_code||','999') , to_char('||seatid||','999999'))';
 ELSE
 QUERY: =' SELECT ahis.bill_mst_GET_USER_NAME
(to_char ('||hosp_code||','999'), to_char ('||seatid||','999999'))';
 END IF;
 OPEN resultset FOR execute QUERY;
 EXCEPTION
 WHEN OTHERS
 THEN
 err := SQLERRM;
 RAISE;
END
$BODY$
LANGUAGE plpgsql VOLATILE SECURITY DEFINER
 COST 100;
```

Where ahis is the schema to which function belong.

### 4.1.6 Functions

- RETURN becomes RETURNS
- We should provide parentheses () even for empty parameter list such as CREATE FUNCTION fn() RETURNS ...
- DEFAULT values for parameters same as oracle
- It can return pseudo type RECORD the caller needs to know the column names
- It also can return set of records; RETURNS SETOF type whereas in Oracle we have TABLE FUNCTIONs.[3]

### 4.1.7 Packages

- A package is a group of variables, functions and procedures
- Postgres dosen't support package it use schema to group functions
- Use (temporary) tables to replace variables
- No substitute for private functions, and variables
- Package Body initialization code: not very often used it Call an initializer function in every member function. [3]

### 4.1.8 Built-in functions

- In place of NVL use SQL standard COALESCE()
- DECODE become the SQL Standard CASE clause
- TO_CHAR() Postgres has this, but not very robust; requires testing of queries as it is 2-argument version whereas Orafce provides the 1-argument version
- SUBSTR() Postgres provides this with SQL standards compliant syntax
- SYSDATE becomes current_timestamp
- MONTHS_BETWEEN two dates not support in postgres but we can have it using:
  - select extract(year from age(current_date, '2012-12-09')) * 12+ extract(month from age(current_date, '2012-12-09'))
  - Or we can create a function as

```
CREATE OR REPLACE FUNCTION
months_between (date, date) RETURNS integer AS
$BODY$
Select abs (months_of (age ($1, $2)))
$BODY$
LANGUAGE sol IMMUTABLE STRICT
COST 100;
```

- PostgreSQL does not have an INSTR function, but you can work around it using a combination of other functions.[3]

### 4.1.9 Others

- INTERSECT Becomes EXCEPT
- Function Call using named notation => becomes :=
  For example: var = fn( c => 10, a => 'xyz', b => 2.5);
  Becomes var = fn( c := 10, a := 'xyz', b := 2.5);
- DUAL Oracle provides this table but postgres don't support it
- In place of ROWNUM postgres use ROW_NUMBER () windowing function and use as a wrapper around the main query, if needed.
- In Place of ROWID postgres use CTID system column which may fail when used in conjunction with partitioning
- OID column which has performance implication since it creates an implicit index
- Postgres doesn't have Optimizer Hints, and doesn't want them but we can discard, or keep for future reference; they won't bite you.[3]

## 4.2 CHANGES AT APPLICATION LEVEL:

### 4.2.1 JDBC driver

A pure Java (Type 4) JDBC driver implementation can be downloaded from http://jdbc.postgresql.org/ assuming the use of the JDK 1.5, download postgresql-8.1-404.jdbc3.jar and make the driver available to the application server classpath. For Tomcat 6.0, copy the file to TOMCAT_HOME\common\lib [1]

### 4.2.2 J2EE Application Servers-Configuring Data Sources

With Tomcat 6.0, to configure a PostgreSQL DataSource specific to an application (i.e. not defined globally), create a context.xml file containing:

```
<context>
<Resource name="AHIS"
 auth="Container"
type="javax.sql.DataSource"
driverClassName = "org.postgresql.Driver"
url="jdbc:postgresql://10.226.17.7:5432/ahis"
username = "postgres"
password = "opensource"
maxActive = "8"
maxIdle = "5"
maxWait = "5000" />
<WatchedResource>WEB-INF/web.xml</WatchedResource>
</context>
```

This application specific file context.xml (as per above) needs to be created under META INF (alongside WEB-INF) in the WAR.
The hierarchical application WAR directory tree should look something like

```
<app root>
<app root>/*.jsp files
<app root>/*.html files
<app root>/*.gif files
<app root>/*.jsp files
<app root>/WEB-INF dir
<app root>/WEB-INF/web.xml file
<app root>/WEB-INF/classes dir
<app root>/WEB-INF/lib dir
<app root>/WEB-INF/*.jar files
<app root>/META-INF dir
<app root>/META-INF/context.xml file
```

To enable the application to reference the Tomcat managed DataSource, a resource XML entry (matching the DataSource defined in context.xml) must be placed in the application web.xml file – for example:

```
<resource-ref>
<description>vAuth Datasource</description>
<res-ref-name>jdbc/vAuthDS</res-ref-name>
<res-type>javax.sql.DataSource</res-type>
<res-auth>Container</res-auth>
</resource-ref>
```

### 4.2.3 Direct JDBC Connections

If non-DataSource derived Connection objects are used, then the URL used to connect to the PostgreSQL server should be of the form *jdbc:postgresql://host:port/database* as seen in an earlier section, this URL should also be used within DataSource definitions. Replace the line (used to load the JDBC driver)
Class.forName ("oracle.jdbc.driver.OracleDriver");
with
Class.forName("org.postgresql.Driver");
and remove any Oracle specific imports, such as
import oracle.jdbc.driver.*; [1]

### 4.2.4 JDBC Connection Setup

Not really PostgreSQL specific issues, but at the Connection level, it is also advisable to switch off the autocommit feature
Connection con;
…
con.setAutoCommit (false);
and set the default isolation level to "read committed"
con.setTransactionIsolation
(Connection.TRANSACTION_READ_COMMITTED); this setup provides a default TX behavior that mirrors that of Oracle. [1]

### 4.2.5 JDBC Extensions

Remove any Oracle JDBC extensions, such as
((OracleConnection) con2).setDefaultRowPrefetch (50);
Instead, the row pre-fetch must be specified at an individual Statement level =>
Eg. PreparedStatement pi = con1.prepareStatement ("select…");
If not set, the default fetch size will default. [1]

## 5. ADVANTAGES OF OPEN SOURCE

There are lower software development costs. There are no licensing fees, and minimal maintenance fees. Open technology is a cost effective and time saving trick we use to make our clients' businesses run better while keeping it in budget. Most of the open-source applications available today are free to download, use and customize and are available under the General Public License (GPL) which define its use. The cost aspect is what makes open-source technologies stand out in comparison to proprietary software. Open-source software has many great benefits, both for a software developer and for a business. Developers have the advantage

of world-wide collaboration with other developers to help audit their software and ensure that the best ideas make it into future versions. Businesses may leverage open-source software a cost-efficient, license free and reliable way to solve business needs whether it be on the desktop or a server.

## 6. CONCLUSION

Major clinical modules of e-Sushrut like Registration& Emergency have been successfully migrated from Oracle database to Postgres Database. Complete system migration will take place in due course of time. Migrated system is working fine and it has opened platform for use of open source software to gain momentum. Open source eSushrut will gradually establish itself as credible alternatives to proprietary software also higher up in the stack.

It is clear that open source software is providing true alternatives for many HIS opportunities, next to the commercial or custom built solutions.

- Open source software is much more accessible. While online demos of proprietary software might give you a first impression of the product, it is normally difficult to get access to evaluation versions. Downloading and installing open source software involves a minimum of bureaucracy and provides real insights into the products.[9]
- Open source software is capturing the essence of what the market is looking for, rather than attempting to provide an overkill of functionality that quickly becomes unmanageable.

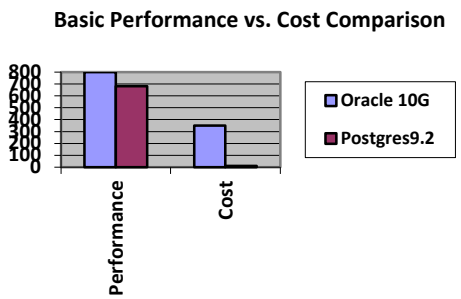


Fig 1: Operation/Second vs. $1000's

## 7. ACKNOWLEDGEMENT

We express our deep thanks to Mr. P. K. Srivastava, Dr. B.K. Murthy and Dr.J.S. Bhatia with the help of whose vast experience, efforts, valuable suggestion, support and motivation has helped us to great extent for the executing the task to achieve desired output.

## 8. . REFERENCES

[1] Drawater 2006, PostgreSQL 8.1 for J2EE/JDBC applications v1.0-http://www.holindis.co.uk/

[2] Oracle 9i documentation- http://download-east.oracle.com/otndoc/oracle9i/901_doc/nav/docindex.htm

[3] Gurjeet Singh 2011, Oracle to Postgres Migration-http://www.pgcon.org/2011/schedule/attachments/205_Oracle_to_Postgres_Migration.pdf

[4] PostgreSQL documentation-http://www.postgresql.org/idocs/

[5] Oracle to Postgre Conversion-http://openacs.org/doc/openacs/html/oracle-to-pg-porting.html

[6] PostgreSQL JDBC 2.0 compliance-http://lab.applinet.nl/postgresql-jdbc/

[7] An important source of information is the PostgreSQL mailing lists-http://archives.postgresql.org/

[8] Web Commerce Group 2002. A Web Commerce Group Case Study on PostgreSQL.-http://www.postgresql.org/files/about/casestudies/wcgcasestudyonpostgresqlv1.2.pdf

[9] Understanding open source CRM-http://www.allindoc.com/story.php?title=crm-open-source-ID949