Practical experience with moving applications to PostgreSQL

Radovan Jablonovsky
Database Architect/DBA
Streamline Technologies Inc
PostgreSQL Conference, 2007
Overview

- System architecture
- Communication
- Databases
- Servers
- Client web app.
- Custom client/server app.
Applications

• Truck on board Orion data collection and communication system

• Servers applications
  • Map server
  • Data transformation server
  • Web application/services, Web servers: .NET
  • Web reporting server: java

• Client/server applications

• Batch data transformation applications
Orion data collection and communication system
Servers applications - Data transformation server

- Receive data from cell network (currently PCS)
- Decode data
- Recalculate and compare information from Trucks to stored on PostgreSQL
- Store data on PostgreSQL
- 24/7 service, windows application
Servers applications - Web application/services
Client/server applications

![Image of EZTicket 2 - Ticket Editor interface]

- **Start Date:** 08-Apr-2007
- **End Date:** 08-May-2007
- **Version:** 1.0.0.11

**Paperwork Type:** 09 - CCrk Reload (Chip) to Silverdale

- **Arrow ProBill #**
- **CURL Ticket #**
- **SilvMill Ticket #**
- **Trailer A**
- **Axle Type**
- **Product:** Chips
- **Load Gross (KG)**
- **Load Tare (KG)**
- **Load Location:** CC Reload
- **Load Date/Time:** May 08 2007 00:00
- **Unload Location:** Silverdale
- **Unload Date/Time:** May 08 2007 00:00
- **Unload Driver**
- **Unload Vehicle**
- **Unload Gross (KG)**
- **Unload Tare (KG)**
- **Truck Type:** Arrow
- **Haul Type:** Back Haul - Leg 2
- **Redirected:** No
- **Route**

**Buttons:**
- Export/Order Import
- Delete Record
- Clear Fields
- Save/Add Load
Batch data transformation applications

- Daily synchronization data from MSSQL company resource repository system with PostgreSQL DB – DTS, psqlODBC
- Daily import data from external text file sources (customers, etc) – DTS, psqlODBC
- SQL scripts to transform and calculate data when moving from one schema to other – stat calc., datawarehouse transformation, PHP
Propagation of PostgreSQL inside company

- Spread informations
- Compare RDBMS features, stability, price, scalability
- Show PostgreSQL support from other companies (Sun, EnterpriseDB, Novell)
- Advantage of following standards (SQL 99)
- Port simple application to PostgreSQL and show performance
- Be prepare for not logical questions and arguments.
# Compare RDBMS features, stability, price, scalability

<table>
<thead>
<tr>
<th>Purpose</th>
<th>PostgreSQL 8.1</th>
<th>MySQL 5.0</th>
<th>FireBird</th>
<th>Oracle 9.i</th>
<th>MSSQL2000</th>
<th>Sybase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with mainstream business application</td>
<td>4 - Breadth of features support mainstream workloads.</td>
<td>3 - very new major version – it still include many bugs.</td>
<td>3 - Support mainstream workloads. Different internal and external SQL</td>
<td>5 - All standard and extended features support mainstream workloads. Excellent scalability and tuning capability</td>
<td>4 - All standard and extended features support mainstream workloads.</td>
<td>4 - All standard and extended features support mainstream workloads.</td>
</tr>
<tr>
<td>Target for migration from Oracle, DB2, MS SQL, or other server class RDBMS</td>
<td>5 - very good 8.1 SQL 99 compliance and database features ease migration</td>
<td>5 - SQL 99 compliance and database features ease migration</td>
<td>4 - SQL 92 entry level compliance</td>
<td>5 - SQL 99 compliance and database features ease migration</td>
<td>5 - SQL 99 compliance and database features ease migration</td>
<td>5 - SQL 99 compliance and database features ease migration</td>
</tr>
<tr>
<td>Distribute with a closed source application</td>
<td>5 - BSD license *, PostGIS GPL license **</td>
<td>3 - Requires commercial license.</td>
<td>4 - and ***</td>
<td>1 – Commercial license required</td>
<td>1 – Commercial license required</td>
<td>1 – Commercial license required</td>
</tr>
<tr>
<td>Use for production quality J2EE/ASP/PHP web site</td>
<td>4 - Leverages scalability and database features.</td>
<td>3 - new major version has few bugs and few missing features, missing GIS</td>
<td>4 - Leverages scalability and database features.</td>
<td>5 - Leverages scalability and database features.</td>
<td>4 - Leverages scalability and database features.</td>
<td>4 - Leverages scalability and database features.</td>
</tr>
<tr>
<td>Use for datawarehouse, data mining, GIS app.</td>
<td>5 – All features for complex queries, GIS module, extended data types</td>
<td>4 – Extremely fast load process, very fast response to simple queries. Complex queries have to be manually tuned, lack of data types</td>
<td>3 – Queries have to be manually tuned, lack of data types</td>
<td>5 – All features for complex queries, GIS module, extended data types, standard interface to many 3-th party tools</td>
<td>4 – All features for complex queries, missing internal GIS module – external module SQS, lack of provided data types –user-defined data types</td>
<td>4 – All features for complex queries, missing internal GIS module – external module SQS, lack of provided data types –user-defined data types</td>
</tr>
<tr>
<td>Cost for version to utilize 2 dualcore CPUs and 8GB RAM</td>
<td>Total: 0 USD</td>
<td>Total: 0 USD</td>
<td>595 USD/server on Linux Total: 595 USD</td>
<td>Enterprise version 40,000 USD/CPU(*0.75/core) on Linux Total: 120000 USD</td>
<td>Enterprise version 19,999 USD/CPU + Ent. Win Server - 3,999 USD Total: 43997 USD</td>
<td>31,679 CAN/CPU on Linux Total: 63358 CAN</td>
</tr>
</tbody>
</table>

* BSD license allows free use within closed source application

** GPL license allows free use but custom changes to source code have to be open too.

### Port simple application to PostgreSQL, performance

Test between MySQL 4.1 and PostgreSQL 8.1. Lower value is the better. Indexes are created on the same fields.

<table>
<thead>
<tr>
<th>Query type/task type</th>
<th>MySQL 4.1 result in sec</th>
<th>PostgreSQL 8.1 result in sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>FleatView level 3 – 10 users x 20 queries; Used table testvwFulCycle in db test, no indices</td>
<td>1176.47</td>
<td>344.83</td>
</tr>
<tr>
<td>FleatView level 3 – 2 users x 5 queries; Used table testvwFulCycle in db test, no indices</td>
<td>38.46</td>
<td>8.93</td>
</tr>
<tr>
<td>FleatView level 3 – 10 users x 20 queries; Used table testvwFulCycle in db test, indices</td>
<td>747.74</td>
<td></td>
</tr>
<tr>
<td>FleatView level 3 – 2 users x 5 queries; Used table testvwFulCycle in db test, indices</td>
<td>27.03</td>
<td>17.86</td>
</tr>
<tr>
<td>Load 393635 rows of vwFullCycle table data into db – MySQL MyISAM engine</td>
<td>7.76</td>
<td>27.1 – use only 1 storage engine</td>
</tr>
<tr>
<td>Load 393635 rows of data from txt file into db – MySQL InnoDB engine</td>
<td>28.28</td>
<td>28.5 – use only 1 storage engine</td>
</tr>
</tbody>
</table>
Introduction PostgreSQL to developers

• Adjust to new or different datatypes
  • Timestamp – MSSQL datetime
  • Serial – MSSQL bigint identity
  • Bigserial – MSSQL integer identity
  • Timestamp with time zone – MSSQL has no such data type

• Functions – MSSQL stored procedures

• Triggers – new, old row approach versus spec. table inserted, deleted
Introduction PostgreSQL to developers

• Problems with data connection components
  • Windows OS
    • ODBC - psqlODBC
      • .NET – slower performance
      • Borland C++ Builder – slow performance, big overhead
  • Native PostgreSQL data components
    • .NET
    • Borland C++ Builder 5-6
      • microOLAP - PostgresqlDAC problems with bigint, timestamp
      • pgExpress – more for Delphi then for C++ Builder
Introduction PostgreSQL to developers

• Problems with data connection components
  • *NIX OS
    • Native libraries in C, C++ - very good and stable
    • PHP – use native connection to PostgreSQL
      • PEAR
        • DB – old but STABLE, almost no bugs, db abstraction layer
        • MDB2 – current, still some bugs, db abstraction layer
        • ADOdb – MS ADODB like interface, db abstraction layer
        • PECL – in development, bugs, sister to PEAR, compiled packages, db abstraction layer PDO
Database Design. What is correct approach?

- **Schema: rdata** – for real time data. Data are pushed from 24/7 days server.
- **Schema: data** – for production online, editable data.
- **Schema: stat** – for online reports, precalculated statistics data updated immediately from schema data. (query response time 1-5 sec)
- **Schema: dataware** – for more complex queries. Schema is updated periodically every 10 min from schemas data and stat. (query response time 5-30 sec)
- **Schema: catalog** – for keeping all changed and deleted information from schema data.
- **Schema: udata** – for user, groups, companies hierarchies, privileges. Information are used to allow and restrict access to app. It can be maintained via manual web app and ActiveDirectory.
- **Schema: rdata** – for resource information. Initially loaded from ReourceMaster.
- **Schema: export** – for data transformation and export to other system(s). Example: financial, client resource management.
- **Schema: import** – for data transformation and import from other system(s). Example: financial, repair shop, resource.
- **Schema: customer** – for data from customers system. Used for reconciliation and other related purposes.
- **Schema: finance** – for internal financial data. Used for reconciliation and other related related purposes. Schema finance is used with schema customer.
Database Design

• Real time schema – eliminate overhead
  • No custom triggers
  • Constraints
  • Foreign Key
  • use indexes wisely. Less is more

• Datawarehouse schema
  • Star and snowflake schemata
  • Transformations data
  • Functions versus SQL batch
  • Indexes
Data migration from MSSQL 2000 to PostgreSQL

• MSSQL DTS
  • Advantage
    • simple development
    • wide variety of drivers – connections
    • flexibility
  • Disadvantage:
    • Performance
    • Resources
Database GUI tools

- CASE – Toad data modeler formally know as CASE Studio 2, ErWin
- SQL Editor – pgAdmin III 1.6.3, Navicat PostgreSQL, Maestro PostgreSQL: comparison
- Graphical Query Editor – important for windows developer
- Administration tools – pgAdmin III for daily maintenance, command line for setup security, performance tuning, server restart
Database GUI tools

- CASE – Toad data modeler formally know as CASE Studio 2, ErWin
Database GUI tools

• SQL Editor – pgAdmin III 1.6.3, Navicat PostgreSQL, Maestro PostgreSQL: comparison
  • Similar features: database object browsing
  • Difference: export, import data
  • Stability

• Graphical Query Editor – important for windows developer
  • Help with building/starting building queries
  • Creating complex queries with joins
  • Have visual grasp of problem/query
  • Improve performance of programmer

• Administration tools – pgAdmin III for daily maintenance, command line for setup security, performance tuning, server restart
  • Excellent for remote administration tasks
  • Problems with using on Novell linux10 (formerly SLES10)
  • command line - most powerful
Difference between MSSQL and PostgreSQL

• Case sensitivity - difference between MSSQL, MySQL and PostgreSQL
• Index usage – Which index use in PostgreSQL versus MSSQL. Clustered indexes
• Scalability. Advantage of *NIX over Windows, 64bit OS
• TSQL – programmatic approach to solve performance issues
• PL/PGSQL – problem with performance in some situations
Difference between MSSQL and PostgreSQL

- TSQL – programmatic approach to solve performance issues
- Example: Calculating Running Totals

<table>
<thead>
<tr>
<th>Solution</th>
<th>No Index</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temp Table/Cursor</td>
<td>2 sec</td>
<td>2 sec</td>
</tr>
<tr>
<td>The &quot;Celko&quot; Solution</td>
<td>20 sec</td>
<td>20 sec</td>
</tr>
<tr>
<td>The &quot;Guru's Guide&quot; Solution</td>
<td>38 sec</td>
<td>17 sec</td>
</tr>
</tbody>
</table>