

Data's Social Media: Linking the Data World Together

JCC Consulting, Inc.

Jeff Haidet

Tom Musson

Keith Hare

Jeff Jalbert

Cheryl Jalbert

Abstract

- We are all familiar with social networking (Facebook, LinkedIn, etc.). These systems allow connections to be built across cities, states, countries and cultures. They give us a medium to connect and talk to one another. Technology is requiring similar links. Our applications all speak their own languages and have their own culture. They may or may not live in the same eco-system. For example, one system might be for OLTP and run on OpenVMS, another system is a mobile system requiring android and yet another is a telephony application requiring a custom interface. The challenge moving forward is going to be linking these systems together so they can all communicate and play together while keeping agile enough so as to not rewrite entire applications if changes are required.
- This talk focuses on a solution that links the data world together. Similar to social media, we will follow data from birth to death showing all of its connections. We will illustrate how data born on OpenVMS and put into the incubator of Rdb makes trip through Oracle (spatial), SQLite (android), SQL Server (internet) and MySQL (vendor specific) and finally back home again. We will touch on the languages involved (Java, XML, PL/SQL), the role of the JCC LogMiner Loader, and the various places the data “checks in” along the way. As we go, you will begin to see how many “friends” our data has along the way.

The Good Old Days

- Do you remember the days when:
 - Visited your friends in person?
 - Actually talked on the phone?
 - Went to a video store to rent a movie?
 - Wondered if a photo you took actually looked okay?
 - Actually took the film to a store to be developed?
 - Wrote a letter and mailed it?

The Good Old Days, Cont'd

- Back in the day, things were fairly simple.
 - Limited number of ways of communication
 - Snail Mail
 - Telephone
 - Person to Person
 - (I'm sure there are others, but I'd like to think I am not that old!)
 - To get something from point A to point B, someone took it there

The Good Old Days of Development

- We can all relate to the following (typical) setup:
 - One physical computer
 - Running one operating system
 - Procedural in nature
 - No GUI
 - A common datastore
 - RMS/Flat files
 - Database

The Good Old Days of Development, Cont'd

■ Advantages

- Some would say development was easier
- Only had a single operating system
 - System service calls simplified
 - All programs had the “same access” to underlying OS components
 - Lock manager
 - File system
- A single developer could be the “master of the universe”

The Good Old Days of Development, Cont'd

■ Disadvantages

- Very closed off
- Moving systems was a royal pain
- Moving operating systems was a nightmare
- At the mercy of the vendor
- If the next best thing came along, integration was very cumbersome if not impossible
- Coordinating with a system introduced by a merger or another department, was sufficient to raise the issue of complete conversion.

Relating the Good Old Days

- The “good old days” of both life and application development have a lot of parallels
 - Closed off communication
 - Closed off cultures
- Parallels can be compared to the advances in things like social networking
 - Data = People
 - Communication = Replication/Data Availability

The Internet Game Changer

- Advances in the internet changed interpersonal relationships and software
 - Email
 - Instant Messaging
 - File Sharing
- Data and communications flowing along unknown media to end devices
 - Snail Mail versus Email

The Game Changer Cont'd

- The rise of the internet allowed for the basis of Social Networking
 - Facebook
 - Linked In
 - MySpace
 - Google+
- The common piece is using the internet as a means of connecting data

The Game Changer Cont'd

- The rise of the internet also changed software
 - Web interfaces
 - HTML
 - Portals
 - FTP
 - RSS
- Instead of having a single machine to worry about, now there are many

Life Today

- Users expect instant gratification
 - Instant Messages (Google Talk, Blackberry Messenger, Facebook, etc.)
 - Text Messages
 - Life of a user getting more convenient
- Software must respond to these demands or be left in the dust
 - Convenience versus Simplicity versus Security
 - Life of a Systems Architect getting harder

Challenge

- Software Architects and Developers charged with developing applications that are easily extendible to an unknown number of targets
 - Phones
 - Tablets
 - Internet PCs/Netbooks
 - Servers
 - Various OSs

Challenge Cont'd

- Legacy Developers charged with opening the legacy systems to integrate with the newer technology
 - Insert/Update records in legacy data store?
 - Read information in “real time”?
 - Replicate data in “real time”?
- Oh...and all of this has to be done yesterday under a limited budget

Planning for the Future

- Five years ago the future was the internet
 - Thin Clients
 - Netbooks
 - Browser based applications
- Today the future is mobile
 - Emphasis on apps, apps, apps.

Planning for the Future Cont'd

- If the target changes so fast, how do you plan for this?
 - What's a netbook? (that's a joke, but seriously, can you still purchase these?)
 - Target device with cheap memory and disk space?
 - Screen size?
 - Data format?

The Crystal Ball



Back to Social Networking

- Consider Facebook:
 - Companies leveraging Facebook as an “app”
 - “Friends”
 - “Connections”
 - “Check in”
- Our data does all of the above.
- Our data is a player in the software social network

Example Case

- Background information
 - RMS based application running on a VAX, migrated from PDP-20
 - Migrated to Alpha (and Rdb) back in mid 1990s
 - Migrated to Itanium in mid 2000s.
 - Cobol/SQLMODs used as predominant in backend
 - Old Scope / Traffic / DECforms screens replaced by Oracle Forms in late 1990s

Business Needs

- Business needs to be agile
 - Customer focused
- Costs are a major factor
- Data away from office
 - Security
- Integration needs with automated systems
 - Payment
 - Phone

Targets

- Office Target
 - End user – PC/Thin client
- Away from Office Target
 - Phone
 - Laptop
 - Tablet
- Oh...and don't forget all the custom targets for vendor specific packages

Restrictions

- Cost is a factor
 - Not a huge budget
 - Management says “make it work”
 - Magic wand
- Security
 - Don’t want name in the news for losing/revealing customer data
- Backend code not easily/quickly changed
 - 3GL code harder to test

The Daunting Task

- As a developer we “gotta make it work”
 - The magic wand
- We “gotta make it run fast”
 - The magic wand again
- We “gotta make it real time”
 - Did I mention that magic wand?
- Its “gotta work for the next great idea”
 - I think I need a bigger magic wand

Just Make It Work



The Decisions

- In this particular example, it was decided to leverage the following technologies
 - JCC LogMiner Loader
 - Java
 - Tomcat
 - Android
 - Web Services

The Targets

- Main System: Rdb (database of record)
 - Web Interface: SQL Server (real time)
 - Mapping: ESRI / Oracle (real time)
 - Telephony Applications: MySQL (real time)
 - Mobile: Android SQLite (batched)
- Data is going to be forced to “check in” at each of the above

Data Sharing



JCC LogMiner Loader

- Challenge: We have to be able to replicate data to the new targets
- The solution: JCC LogMiner Loader
 - The Loader can push data to a data source with a Type 4 JDBC driver
 - MySQL, SQL Server, etc.
 - OCI Interface
 - Oracle
 - Near real-time replication

Advantages to the Loader

- Data replication (outbound) standardized
 - Real-Time replication from database of record all handled the same way
- Operates with minimal overhead
- Does not require a bunch of fancy programming
- A great support team ☺

Challenges for the Loader

- The Loader must replicate data to a target table or tables
- Data combinations (i.e. merging with other rows)
 - Triggers in target
 - Constraints in target
 - Chicken and the Egg issues
 - Temporal Issues

Important Note About the Targets

- In some cases, vendor packages expect the data to be in a very specific format.
 - Telephony apps may want dollars and cents in different fields
 - Telephone number + area code may be stored differently
- In order to accommodate these vendor specific requirements with the Loader, database triggers may be used

Loader to Target

- Data Replication
 - Table to Table
- Trigger
 - Modify Record
- Data finally in destination format

Database of Record

- There are significant -- and possibly unsolvable issues -- in mixed architectures that do not choose a “database of record”.
- The database of record must be able to recreate the transient data
- For this example, Rdb is the database of record

Rdb and the Main Application

- 3GL code
 - Cobol
 - Basic
 - C
- Stored procedures/functions in Rdb
- User interfaces:
 - Oracle Forms and Java

Bolt-On #1

Website

- Business decision: Offer internet users access to data contained in main database
- Security concerns
 - Don't want to expose all the corporate data, just a subset
- Developers: Only know Microsoft .NET technologies using SQL Server

Website Cont'd

- Loader replicates a subset of the data to SQL Server in near real time via the Type 4 JDBC driver
- Data swallowed back from the website via two methods:
 - Legacy: “Select from/insert into” [java code]
 - Future: Web service call

Website Roundtrip

- Rdb → Loader → Sql Server
- Sql Server → Java Code → Rdb
- Sql Server → Web Service → Rdb

Note: The Java code above is executed on the trusted machine running Rdb

- Remember, any time a record is written back to Rdb, the Loader may see it
 - Look out for circular updates

Sidebar: Web Service

- Wikipedia: A **Web service** is a method of communication between two electronic devices over a network. Web Services were intended to solve three main problems, that is Firewall Traversal, Complexity, and Interoperability.
- Wikipedia: **Interoperability** is a property referring to the ability of diverse systems and organizations to work together (inter-operate).

Web Service Cont'd

- Tomcat allows for the creation of portable, generic web services
 - Why Tomcat?
 - There's a flavor of Tomcat for a vast majority of the operating systems
 - Tomcat allows for the execution of Java
 - There's a flavor of Java for the vast majority of the operating systems
 - Oracle Rdb's JDBC driver can plug into Tomcat

Tomcat Cont'd

- Not knowing what tomorrow will bring, having a solution built on a generic solution is rather handy
 - If an OS change is in the cards, code can be migrated without recompilation (etc.) to new platform
 - Load Balancing
 - Portability

Back to Website Roundtrip

- Rdb → Loader → Sql Server
- Sql Server → legacy java code → Rdb
OR/AND
- Sql Server → Web Service → Java Service →
JDBC → Rdb

Bolt-On #2

ESRI / Mapping

- Wikipedia: ESRI is a software development and services company providing Geographic Information System (GIS) software and geodatabase management applications
- ESRI was a package chosen for its superior mapping capabilities
- The software talks to Oracle (with spatial components) running on Windows

ESRI/Mapping Cont'd

- In addition to drawing maps in ESRI, an interactive mapping application for users also exists
 - Runs against Oracle on Linux
 - Built in Microsoft SilverLight
- In both mapping examples, the Loader replicates textual data from Rdb to the maps

ESRI / Mapping Back to Rdb

- Interactive mapping application must be able to return information back to the database of record
 - Location information (Geo Coordinates)
 - Asset information for inventory
 - Etc.
- We model the return trip the same as SQL Server

ESRI/Mapping Roundtrip

- Rdb → Loader → Oracle (Windows)

- Rdb → Loader → Oracle (Linux)

- Oracle → legacy Java code → Rdb

OR/AND

- Oracle → Web Service → Java Service → JDBC
→ Rdb

- AND.....

ESRI/Mapping Roundtrip Cont'd

- DBLink

- A virtual table that exists in Oracle that writes back to Rdb

- Oracle → DBLink (we are in Rdb at this point)

- Note: the insert via the DBLink sends a BLAST to a separate server side process.

Bolt-On #3

Telephony Applications

- Telephony application = IVR
- Wikipedia: **Interactive voice response (IVR)** is a technology that allows a computer to interact with humans through the use of voice and DTMF keypad inputs

IVR

- IVR requires the use of MySQL
- IVR requires the data to be in a specific format
- IVR collects information that needs to be processed immediately and in batch
 - Payment validation (immediate)
 - Telephone number change (batched)
 - Etc.

IVR Challenge

- The payment validation does not exist within the database of record
 - Located in Web Space
- Should the IVR access the Web directly?
 - Hub and Spoke model to ensure security and logging
- Therefore data may need to come into Rdb to be redirected to the Website back to Rdb and therefore back to the IVR
 - Did I mention this has to be fast?

IVR Cont'd

- Getting data to the IVR is easy
- Rdb → Loader → MySQL
- Remember, there may need to be triggers in the target to format data

IVR

Getting Data Back

- Two types of data
 - “Batch”
 - Type of data that the database of record simply swallows
 - “Broker”
 - Type of data that needs to be passed on
- IVR will provide both

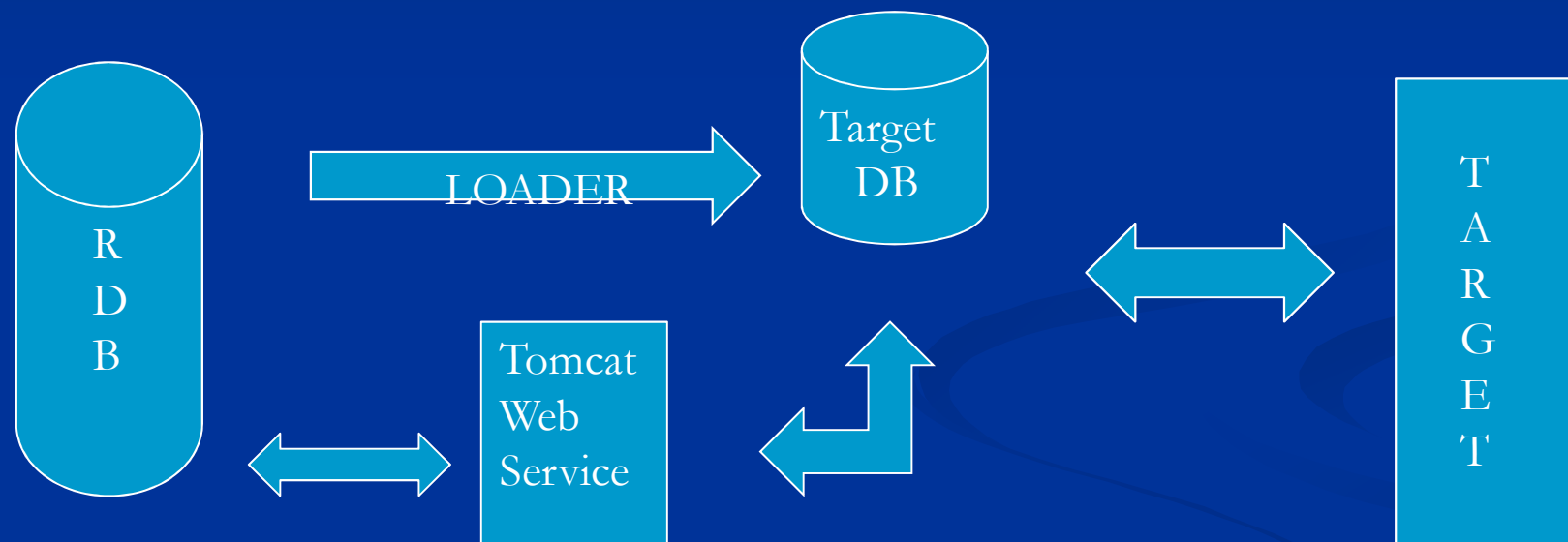
IVR “Batch” Data

- Model is exactly like that of the others
- Rdb → Loader → MySQL
- MySQL → Web Service → Java Service → JDBC → Rdb

IVR “Broker” Data

- Rdb → Loader → MySQL
- MySQL → Web Service → Java Service →
JDBC → Rdb → Loader → MySQL
| (and/or) → Java Service → Web Service →
Payment Environment (Miracle Happens) →
Web Service → Rdb → Loader → MySQL
- The database of record is acting as a broker
 - Social Networking Concept of a “friend of a friend”

A Graphical View



Bolt-On #4

Mobile Applications

- Apps, Apps everywhere
- Public perception that mobile apps are now the “in thing”
- Potential Targets:
 - iOS (Apple)
 - Android (Google)
 - WebOS (HP/Palm)
 - Blackberry OS (RIMM)
 - Others are too small to worry about

Mobile Choice

- Business decision was to support only Android to start with
 - Largest market share
 - Open development
- Potentially support iOS in the future
 - If demand is there...
- No other current plans

Android

- Android provide access to two types of devices
 - Phones
 - Tablets
- Assuming version compatibility, code may be deployed to any android device
 - Make support much easier/faster

Android Challenge

- No constant connection
 - Cell service issues
 - Data service issues
- Memory “ain’t cheap”
 - You get what you get
- Disk space is finite
 - SD cards make is a tad better, but....
- Planning mobile applications requires good development practices

SQLite

- Wikipedia: **SQLite** is an ACID-compliant embedded relational database management system contained in a relatively small (~ 275 kB) C programming library. The source code for SQLite is in the public domain and implements most of the SQL standard. In contrast to other database management systems, SQLite is not a separate process that is accessed from the client application, but an integral part of it.

SQLite Cont'd

- Single file database
- Comes buried in android
 - Given that its there, why not use it?
- Size and Performance constraints

Data to/from SQLite

- Opted to not use the Loader for data migration due to connectivity limitations
- Device will make web service calls to drop off/pick up data when connection is available
 - A poor man's data synch
 - Send encrypted XML payload back and forth
 - XML parsers built into Android

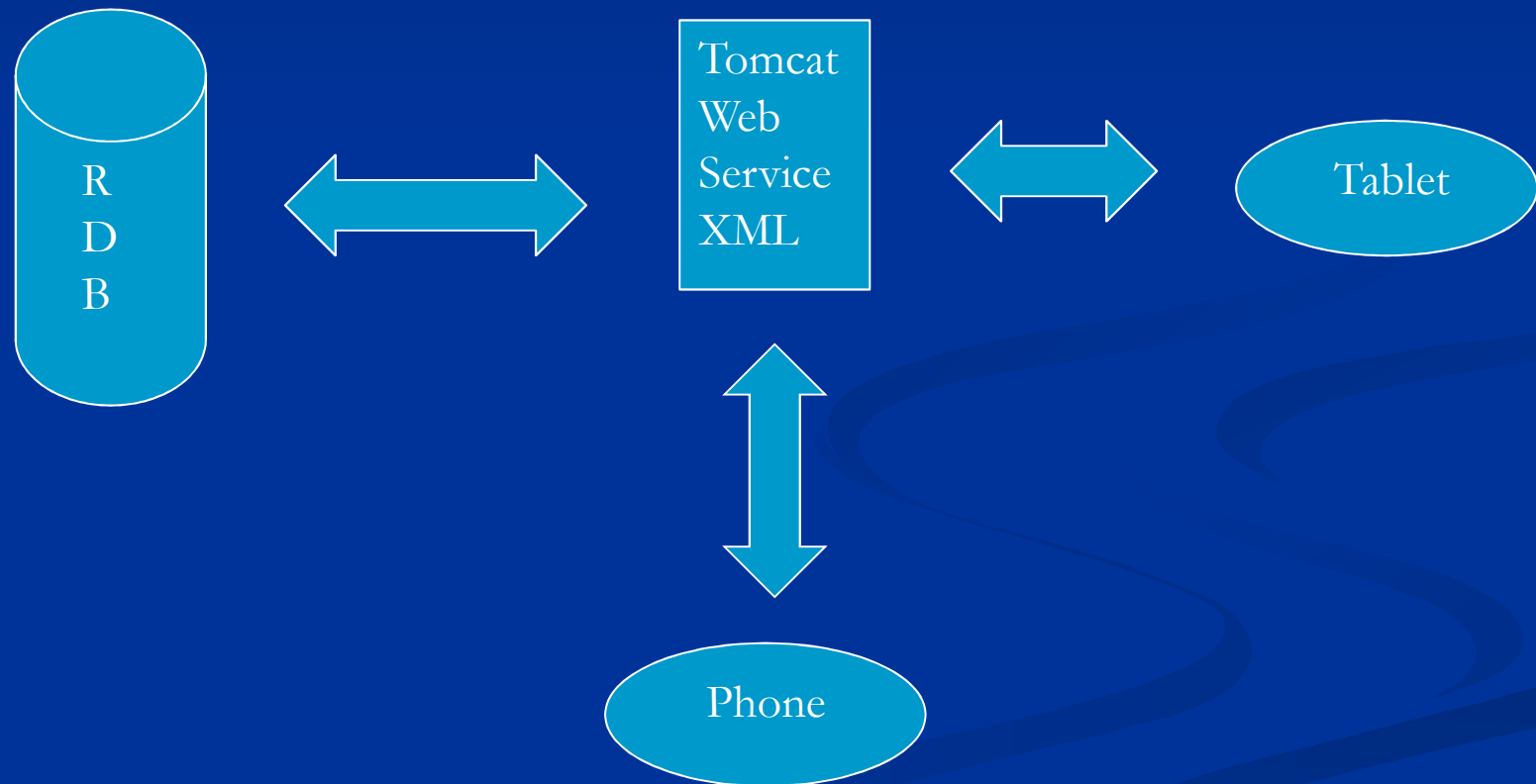
Android Data Roundtrip

- Android Device \leftrightarrow Tomcat \leftrightarrow Rdb
- Tomcat Layer
 - Handles XML parsing
 - JDBC binding

A Note on Web Services

- The same web services may be reused to get data back into Rdb
- Web Services can call other Web Services
- A web service acts like a bridge between disjoint systems

Another Graphical View



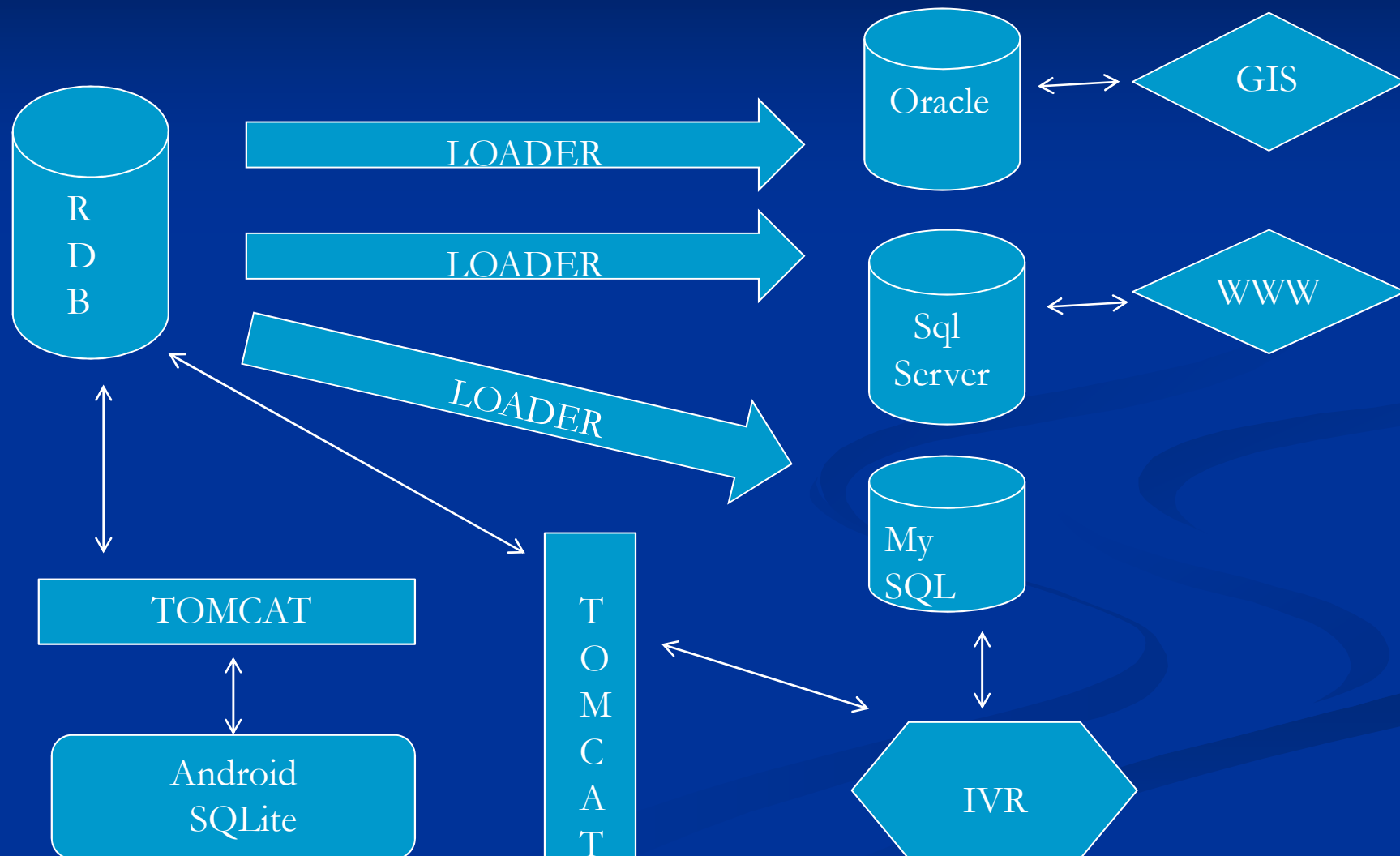
Data Lifespan

- Data “Born” and stored in database of record
 - Upon commit, the Loader(s) see it
- Data’s birth propagate via the Loader to its targets
 - Sql Server
 - Oracle (Windows and Linux)
 - MySQL
- Data’s birth available to Android apps on next synch

As the Data Grows Up...

- As the data grows (i.e. updated, additional information added/subtracted) this data is in turn collected via the various applications and written back to the database of record
- Upon write to database of record
 - (see previous slide)
- The loop has begun!!

Putting It All Together



Insights

- With the JCC LogMiner Loader, pushing data is relatively easy
 - Especially if near real time performance is a requirement
- Data manipulation is the target (i.e. row merges etc.) may require triggers
 - Syntax is the biggest hurdle
 - There might be some limitations

Insights Cont'd

- Packages that hit the “sweet spot” are only going to be more and more valuable
 - IVR
 - Mobile
 - Who knows what's next?
- Tomcat's portability a huge asset
- Java's portability a huge asset

Data As A Social Being

- Our data's friends are its systems
 - Each system is culturally different
 - Windows
 - Linux
 - OpenVMS
 - Android
- Our data cannot choose its friends, but it can play nicely with them
- Our data's connections are only going to grow over time

Summary

- Like our connections grow on Facebook, our data is linked and will be linked to an ever increasing array of applications
- The challenge is staying up to date with the technology trends to make informed decisions to increase success in the future
- Portability is key
- You always want “friends” – without them you are stuck.

Questions? Comments?

Any questions or comments?

Email: jeffh@jcc.com

Any questions regarding the Loader?

Email: info@jcc.com

A special thanks to all the members at JCC who make working on projects such as this fun!