DB2 Connect: History of Supporting Application Infrastructure

Curt Cotner
IBM Fellow, CTO for IBM Database Servers
cotner@us.ibm.com
DB2 Connect

- Make DB2 for z/OS into an enterprise database server
- Greatly improve programmer and DBA productivity
- Deliver on DB2 for z/OS strengths:
  - Continuous application availability
  - Manage and balance the workload (SYSPLEX)
- Improve mainframe resource utilization
- Provide transparent access via SQL and standard APIs to CICS, IMS, MQ, VSAM and other data sources
- Extend applications and data to mobile devices
- Make DB2 for z/OS a full participant in SOA
- Simplify application deployment
Desktop PCs

16-bit machines

DOS

DB2RA Protocol

Netbios TCP/IP

OS/2

Database Manager

Data

Basic
PowerBuilder
Lotus 1-2-3
Lotus Approach

Windows

OS/2

Database Manager
DB2 Connect – first introduced in 1993

Major technology challenges of the time:
- setup and resource usage of SNA
- only 300 threads in DB2
  (one PC could need 5-10 threads)
- how to administer 100’s of desktops?

Desktop PCs

16-bit machines

Basic
PowerBuilder
Lotus 1-2-3
Lotus Approach

DOS

OS/2

Windows

S/370 running MVS and DB2 2.3

Data

Many competitors:
MDI Gateway
Oracle Gateway
Sybase Gateway
Attachmate
Rumba
CrossAccess

StarQuest
Microsoft Host Int. Server
Neon Shadow Direct
HiT Software
DataDirect
EDA
DB2 Connect – Emergence of the “gateway”

Advantages in the 90’s:
- eliminates need for SNA on the PC
- protocol conversion from DB2RA to DRDA
- pool connections to minimize need for DB2 threads
- centralizes DB2 Connect admin within IT team
Advantages in the 90’s:
- eliminates need for SNA on the PC
- protocol conversion from DB2RA to DRDA
- pool connections to minimize need for DB2 threads
- centralizes DB2 Connect admin within IT team
DB2 Connect
Current day

- Desktop PCs
- Application servers
- Web application servers

APIs

Communication

Data

System z
or System i

- ADO.NET, ADO
- ODBC, OLE DB
- JDBC, SQLJ
- pureQuery
- DB2 CLI
- PHP
- Ruby
- Python
- Embedded SQL
DB2 Connect questions I hear all the time

Is DB2 Connect required if I use JDBC Type 4 driver?

Is DB2 Connect going away?
DB2 Connect PRODUCT is always required!

DB2 Connect PRODUCT is NOT going away!
DB2 Connect PRODUCT:

- A part number you purchase from IBM.

- Five different Editions are available:
  - Personal Edition,
  - Enterprise Edition,
  - Application Server Edition,
  - Unlimited Edition,
  - Unlimited Edition for System i.

DB2 Connect SERVER:

- Component of the DB2 Connect PRODUCT

- You install it on a
  - Linux: Intel/AMD, POWER (System p&i), System z
  - UNIX or
  - Windows server.

- Deploying DB2 Connect SERVER component is optional.

- Can instead deploy one or more of the other components (e.g., JDBC Type 4 driver) that are a part of the DB2 Connect PRODUCT.
DB2 Connect Deployment Models

- **Direct connectivity from each desktop: DB2 Connect Personal Edition**
  - Install DB2 Connect Personal Edition on each desktop for direct connectivity to the mainframe

- **Utility: DB2 Connect server farm:**
  - A cluster of DB2 Connect servers to be used by multiple applications
  - Drivers (ODBC, OLE DB, .NET, JDBC, SQLJ, Embedded SQL etc.) deployed to all desktops and application servers

- **Application server co-location:**
  - DB2 Connect installed on each application server
DB2 Connect
Recommended Deployment Options

Several options:
- Personal Edition or
- DB2 Connect Server or
- file server

Recommendation:
- Personal Edition is best for small numbers of end users
- DB2 Connect server or file server deployment is best for lots of desktops

Rationale:
- drivers now include the key DB2 Connect gateway features (sysplex workload balancing, connection concentrator, XA support, automatic reconnect, etc.)
- fewer potential points of failure
- less hardware cost
- less system administration cost
- fewer network hops (up to 40% better elapsed time)
- simplified failover strategy
- less complex problem determination and monitoring
DB2 for z/OS Sysplex Balancer

- High Availability feature added for Type 4 connectivity in DB2 LUW v8.1 FP10
- The driver maintains physical connections, known as transports, to the various members of a DB2 for z/OS parallel sysplex
- Using feedback received from WLM, the driver balances work for new transactions across the data sharing members by associating connections with various transports
- Rebalancing to another member will occur in case of a member failure and is transparent if the failure occurs on a transaction boundary
DB2 for z/OS Connection Concentrator

- High Availability feature added for Type 4 connectivity in DB2 LUW v8.1 FP10
- Allows a given JVM to map multiple connections onto a smaller number of physical connections known as transports
- Helps reduce total number of physical connections to a database server
- Automatically associates connection with a new transport in case of a connectivity failure
- Failure is transparent if it occurs on a transaction boundary
DB2 Application Development
What is about?

- SQL
- APIs and drivers that implement them
- Application Development Tools and Utilities
- Information
- Application deployment
SQL-based DB2 APIs

- **Microsoft environments:**
  - ADO (via ODBC or OLE DB)
  - ADO.NET

- **Java programmers:**
  - JDBC
  - SQLJ

- **UNIX, Windows C programmers**
  - DB2 Call Level Interface (DB2 CLI)
  - ODBC
  - Embedded SQL
Non-SQL based DB2 APIs

- **Microsoft environments:**
  - Web Services: application programmer accessing DB2 as a set of Web Services created by an application DBA

- **Java programmers:**
  - **pureQuery**: you will want your Java programmers to use this
  - Java Beans: J2EE programmer accessing DB2 as a set of Java Beans created by an application DBA by wrapping DB2 stored procedures.
  - Web Services: application programmer accessing DB2 as a set of Web Services created by an application DBA
Scripting Languages
The great shift in programming

- The shift from Desktop to Web applications led to an increase in the adoption of dynamic languages like Ruby, Python and PHP.
- This trend is coupled with the emerging popularity of Web frameworks for these languages, like Ruby on Rails, Zend Framework and Django.
- DB2 Connect objective: make DB2 for z/OS premiere data server for applications built using scripting languages
Ruby

- From 27\textsuperscript{th} in 2004, to 9\textsuperscript{th} most popular language in the world in 2007 (source: TIOBE Index)
- Ruby on Rails is the fastest growing Web framework in the industry

![TIOBE Index History for Language Ruby](image)
Ruby and DB2 Connect

- DB2 Connect provides IBM developed and supported Ruby driver and a Rails adapter.
- 8 releases of the IBM_DB gem in the past year alone. Reached version 1.05 of the IBM_DB gem with major coverage in the Rails community. Shipped 4 releases of Starter Toolkit for DB2 on Rails.
- Support for the latest Rails v2.3
- The same driver also supports other IBM databases: DB2 LUW, DB2 for i5/OS, Informix Dynamic Server
- Major improvements in the support of Ruby on Rails compatibility with DB2, including acceptance of patches submitted by IBM into the Rails core.
- Complete support for the Migration framework.
- Several applications deployed in production and thousands downloads.
- Active community with more than 500 messages in the support forum.
- Mac OS X support.
- Used Ruby on Rails in DB2 to deploy several applications in the Cloud (Amazon EC2)
- From 10th in 2001, to 5th most popular language in the world today (source: TIOBE Index).
- Over 2 million Zend Framework downloads.
PHP and DB2 Connect

- Choice of two PHP extensions:
  - IBM_DB2: This is the common API for all the dynamic languages
  - PDO_IBM: best for object oriented PHP style of coding

- Current release of IBM_DB2 is 1.7.1, with 25 releases in between this and the 1.0 release.

- Drivers developed and supported by IBM. This is unique in the PHP world

- Published scalability case study with 10,000 concurrent database connections.

- Zend Core for IBM now ships DB2 Express – C. Great for development and then re-hosting on DB2 for z/OS

- Integration with popular PHP build tools (e.g. Zend Studio)
Python

- From 13<sup>th</sup> in 2003, to 6<sup>th</sup> most popular language in the world today (source: TIOBE Index).
- Several frameworks exist, with Django increasingly popular amongst Web developers.
- SQL Alchemy is the most popular ORM in the Python community.
Python and DB2 Connect

- **Three modules released:**
  - `ibm_db`: an extension driver for DB2 and IDS
  - `ibm_db_dbi`: a wrapper compliant with the Python DB-API 2.0 specification
  - `ibm_db_sa`: a Python adapter implementing the SQLAlchemy 0.4 specification APIs

- **Current version for the two drivers is 0.4.1, while the SQLAlchemy adapter is 0.1.6. For a total of 21 releases in 2008.**

- **Support for SQLAlchemy allows DB2 to be used with several Python Web frameworks like Pylons, TurboGears**

- **Django support is available today!**
Perl and DB2 Connect

- In slight regression, but still the 9th most popular language in the world today (source: TIOBE Index).
- Released a DBD::DB2 driver module that implements the DBI interface for DB2 on all the supported OS platforms.
- 3 Levels of support provided by IBM.
- Rich API and stable v1.2.
DB2 Connect and Dynamic Programming Languages

- DB2 Connect enables you to develop and deploy DB2 for z/OS and DB2 for i5/OS applications using popular dynamic languages:
  - PHP
  - Python
  - Ruby on Rails
  - PERL

- Provides the only vendor developed and supported drivers for DB2 for z/OS and DB2 for i5/OS
Java access to relational – pureQuery on ramp for all cases

- I don't want to have any SQL in my program.
- I need to run on J2SE.
- I want a full persistence layer with transaction management.
- I just need a light-weight interface that makes SQL coding easy.
- I need OR mapping between relational and my object model.
- I need an interface that allows efficient SQL access and detailed monitoring for problem diagnosis.
- I need an interface that makes SQL coding easy.
- I need business objects returned to my program – not individual columns.

pureQuery Technologies

© 2009 IBM Corporation
Java Data Access – many forms

Spectrum of choices

POJO with inline SQL
JDBC, SQLJ

POJO  iBatis  Hibernate  EJB 3

EJB application
EJB query lang
OR mapping layer
Persistence Layer

cfg file
for named queries

Pro’s:
Simplicity
Easy to control SQL
Good performance
Good monitoring (SQLJ)

Con’s:
Not tied to object model
More work for app pgmr

Pro’s:
Less work for app pgmr
Access via OO business objects

Con’s:
Complexity
Less control over SQL issued
Performance can suffer
Very difficult to monitor or diagnose problems
What performance/diagnosis challenges?

EJB Query Language:
SELECT object(e) FROM Employee e
WHERE e.dept=?1 AND e.salary>=?2

Query language is a subset of SQL. Doesn't have all the SQL features you want.

App query syntax is different from SQL query. How do you track problem SQL queries back to the app that issued the original query???

In most cases, queries map to JDBC. No ability to lock in access path at program deployment. No ability to search catalog to see which queries are issued by a given program.

Often, app query is intercepted by persistence layer, and the resulting SQL query looks nothing like the app query.
-- Resulting query might perform badly.
-- Changing app query might not result in a similar change in the SQL query…
What is pureQuery

pureQuery is a high-performance, data access platform to simplify developing, managing, securing, and optimizing Java data access for new and existing applications.

pureQuery Components:

- **Development tools**
  Integrated development environment with Java and SQL support delivered with Optim Development Studio

- **Simple and intuitive API**
  Enables SQL access to databases or in-memory Java objects

- **pureQuery Runtime**
  Flexible deployment options with static SQL support with Optim pureQuery Runtime for z/OS or for Linux, UNIX, and Windows
pureQuery throughout the lifecycle

Model Integration
• Generate Object Model and code from Data Model

Jump Start Application Design
• Generate SQL and Code from Database Objects
• Setup basic DAO Pattern

Existing JDBC and .NET to Static
• Reroute Dynamic Queries to Static

Code Development Productivity
• Code Generation, Content Assist Database aware, Java SQL Editor

SQL Replacement
• Replace Query w/o changing source

SQL Performance Metrics
• Find and sort query elapsed time from Java

Problem Determination
• Monitor WebSphere Connection Pool, JDBC Driver, Network
• Track back to SQL and line of code in the application

Java to SQL Integration
• Categorize by Java, SQL, Database, Packages, track back to line of code

Static SQL
• Lock in Access plans, Improve Security, Consistent Performance

SQL Injection Prevention
• Lock down SQL for Dynamic

New!

IBM Data Management
© 2009 IBM Corporation
What does pureQuery offer?

- **Take back control of the database**
  - See inside applications that are driving your database
    - Understand where SQL comes from
    - Understand which developers you need to talk to and when
    - Understand when frameworks and ORM’s are getting in the way
  - Optimally organize SQL
    - Better manage database security
    - Improve use of database resources
  - Control performance
    - The DBA decides how the SQL is deployed
    - Understand and lock down the access plan for SQL
  - Easier problem determination and troubleshooting
    - Correlate problem SQL with applications, ORM’s and frameworks
  - Avoids SQL injection attacks
    - Control which SQL can be issued
Simpler Development

**pureQuery API’s:**

Employee my_emp = db.queryFirst("SELECT Name, HomeAddress, HomePhone FROM Employee WHERE Name=?", Employee.class, my_emp);

-or-

Employee my_emp = getEmployee(name);

**SQLJ:**

```java
#sql [con] {
SELECT NAME, ADDRESS, PHONE_NUM INTO :name, :addr, :phone FROM EMP
WHERE NAME=:name 
}
new Employee my_emp;
my_emp.setName(name);
my_emp.setHomeAddress(addr);
my_emp.setHomePhone(phone);
```

**JDBC:**

```java
java.sql.PreparedStatement ps = con.prepareStatement("SELECT NAME, ADDRESS, PHONE_NUM FROM EMP WHERE NAME=?");
ps.setString(1, name);
java.sql.ResultSet names = ps.executeQuery();
names.next();
new Employee my_emp;
my_emp.setName(names.getString(1));
my_emp.setHomeAddress(names.getString(2));
my_emp.setHomePhone(names.getString(3));
names.close();
```

**Table**

<table>
<thead>
<tr>
<th>Table</th>
<th>Column</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMP</td>
<td>NAME</td>
<td>CHAR(64)</td>
</tr>
<tr>
<td>EMP</td>
<td>ADDRESS</td>
<td>CHAR(128)</td>
</tr>
<tr>
<td>EMP</td>
<td>PHONE_NUM</td>
<td>CHAR(10)</td>
</tr>
</tbody>
</table>

**class Employee**

```java
{   public String Name;
    public String HomeAddress;
    public String HomePhone;
...
}
```
What you will want to tell your Java programmers

- EJBQL and runtime SQL generation based on object manipulation make the value of pureQuery even more important in the JPA setting.

- IBM is enhancing our JPA implementation with both pureQuery APIs and pureQuery runtime lifecycle benefits.

- JPA w/pureQuery enables problem determination, optimization, and governance connecting the EJBQL and business logic to the actual SQL and database operation.

- JPA / EJB3 is a J2EE5 standard.

- WebSphere is delivering JPA.

- Apache openJPA is the only JPA implementation supported by more than one major vendor: BEA and IBM.

- Hibernate users should use the JPA standard APIs and migrate to the openJPA implementation.
pureQuery Improves Java Data Access Performance

- Choose between dynamic or static execution at deployment time, rather than development time
- Deliver static SQL execution value to existing DB2 applications
  - *Make response time predictable and stable* by locking in the SQL access path pre-execution, rather than re-computing at access time
  - *Limit user access* by granting execute privileges on the query packages rather than access privileges on the table
  - *Aid forecasting accuracy and capacity planning* by capturing additional workload information based on package statistics
  - *Increase system capacity* by driving down DB cycles
Optim pureQuery Runtime
Better Java performance (DB2 for z/OS)

- In-house testing shows double-digit reduction in CPU costs over dynamic JDBC

- IRWW – an OLTP workload, Type 2 driver (local call)
- Cache hit ratio between 70 and 85%
- 42% reduction in CPU per transaction over dynamic JDBC
pureQuery for existing JDBC work flow
Without changing the application

- **Capture**
  - Determine SQL coming from application

- **Review**
  - Review SQL to identify issues
    - **Optimize**
      - Replace SQL with enhanced to SQL
    - **Restrict**
      - SQL approved list

- **Configure**
  - Flag SQL to be in a package and configure driver properties
    - **Bind**
      - Create packages according to configure

- **Execute**
  - Switch to run with new configuration
pureQuery for existing applications

*Improve Java data access performance—* **without changing a line of code!**

pureQuery enables static execution for existing **JDBC** applications (custom-developed, framework-based, or packaged)
Dynamic Execution

- Computing query execution access paths on-the-fly affects performance.
- Prevents administrators from granting security at finer levels.
- Unpredictable access path.
- Difficult to pin-point source of the problem.
- Difficult to summarize performance data at program level.
- No record of which objects are referenced by a compiled SQL statement.

Dynamic Execution is Problematic.
SQL Execution - Dynamic vs. Static

Dynamic SQL:
1. Check auth for package / plan
2. Parse SQL Statement
3. Check Table / View Auth
4. Calculate access path
5. Store access path in a temporary package
6. Execute SQL statement

Static SQL:
1. Check auth for package / plan
2. Extract access path from catalog or package cache
3. Execute SQL statements
DB2 Connect Support for .NET Development

- Industry leading support for .NET
- High performance and quality .NET drivers
- Deep integration into Microsoft Visual Studio
- Member of the VSIP program since 2002
- Now pureQuery is available for .NET!
Optim pureQuery Runtime
Better .NET performance (DB2 for z/OS)

• 21% higher throughput compared with dynamic SQL at 100% cache hit rate
• 159% higher throughput compared with dynamic SQL for more typical 79% cache hit rates
Optim pureQuery Runtime
Lower resource use for .NET apps (DB2 for z/OS)

- 17% reduction of System z CPU utilization compared with dynamic SQL at 100% cache hit rate
- 61% reduction in System z CPU utilization with dynamic SQL for more typical 79% cache hit rates
What’s So Great About DB2 Accounting for CICS?

- What’s great about 30 yr old CICS apps?
  - See end user ID (CICs thread reuse)
    - Take a user identify and push it down to the database and compare users to each other
  - See address space (AOR name)
  - See program (package level accounting)
    - See package data between programs or how a package has changes over time
  - See TX (PLAN name)
    - Measure correlated performance

- CICS give you performance monitoring and tuning which is really a giant Pivot table
  - Rotate and analyze the program
  - How much of my system does this application take?
  - What called this poorly running query
  - +++
What if we took away the key monitoring tokens you use for CICS and you only knew 3 things about the CICS workload?
- Only one plan name: COBOL
- CICS AOR started task name
- No end user ID - just user ID of the AOR that started task

How would you know?
- Which app is running?
- Which user is running?
- Which developer wrote the app?
- What other SQL does this app issue?
- When was the app last changed?
- How has CPU changed over time?
- +++
You’d Have “Life Before Optim pureQuery”

- **What is visible to the DBA?**
  - IP address of WAS application server
  - Connection pooling user ID for WAS
  - Application is running JDBC

- **What is not known by the DBA?**
  - Which application is running
  - What application developer wrote it?
  - What other SQL does this application issue?
  - When was the application last changed?
  - How has CPU changed over time?

- **We seemed to have forgotten a lot in this modern era:**

  Enterprise Amnesia
Enterprise Amnesia is a Well Known Problem

- This is a time-consuming and staff intensive processes
  - No holistic view
  - No resource consumption profile
  - No correlation of SQL to source application
  - Problem isolation requires cross-role collaboration
What’s so Great About pureQuery Accounting?

- Data Studio and pureQuery provide the same granularity for reporting WebSphere’s DB2 resources that we have with CICS:
  - By transaction
    - Set Client Application name
  - By class name
    - Program/Package level accounting
  - By address space
    - IP address
  - By end user ID
    - DB2 trusted context and DB2 role

- Makes it very easy to:
  - Isolate performance problems
  - Perform capacity planning exercises
  - Analyze program changes for performance regression
  - Compare user’s resource usage
  - +++
SQL Insight
Correlate SQL to JAVA

- Correlate the SQL statement back to the originating line of JAVA code
  - Custom-developed, framework-based, or packaged applications

- Impact analysis identifies application code impacted due to database changes
  - Answered “where used” questions like “where is this column used within the applications”
  - Use with modern JAVA frameworks e.g. Hibernate, Spring, iBatis, OpenJPA
pureQuery Outline
Speed up problem isolation for developers – even when using frameworks

Captured SQL

Statement correlation

Table correlation
All SQL with Execution Time
The request loses details about "who" issued it on its way through the layers. Request flows through many layers of the application stack. Performance bottlenecks can be caused by any layer! The request loses details about "who" issued it on its way through the layers. Today's monitoring is done here.
Extended Insight (EI) Overview dashboard

![Extended Insight Analysis Dashboard](image)

Use this dashboard to monitor and analyze the workloads executed by application servers and client applications.

### Workload Cluster Details

<table>
<thead>
<tr>
<th>Workload Cluster Group/Workload Cluster</th>
<th>Average End-to-End Response Time (s)</th>
<th>Maximum Inflight Elapsed Time (s)</th>
<th>Average Data Server Time (s)</th>
<th>Average Network Time (s)</th>
<th>Average Client Time (s)</th>
<th>Warning (%)</th>
<th>Critical (%)</th>
<th>Transactions (min)</th>
<th>Row Read Rate</th>
<th>Row Modified Rate</th>
<th>Row Returned Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clientserver-IDs</td>
<td>0.884</td>
<td>38.734</td>
<td>52.984</td>
<td>0.619</td>
<td>0.002</td>
<td>0.958</td>
<td>0.02</td>
<td>4.996</td>
<td>27.32</td>
<td>274,540.453</td>
<td>8,116.01</td>
</tr>
<tr>
<td>deployy_admin</td>
<td>2.104</td>
<td>2.453</td>
<td>11.484</td>
<td>1.458</td>
<td>0.153</td>
<td>0.302</td>
<td>N/P</td>
<td>N/P</td>
<td>N/P</td>
<td>11.022</td>
<td>31.917</td>
</tr>
<tr>
<td>mery</td>
<td>2.651</td>
<td>36.734</td>
<td>52.984</td>
<td>1.643</td>
<td>0.003</td>
<td>0.120</td>
<td>0.082</td>
<td>15.361</td>
<td>8.684</td>
<td>274,197.055</td>
<td>23,963.204</td>
</tr>
<tr>
<td>paul</td>
<td>0.484</td>
<td>0</td>
<td>1.126</td>
<td>0.175</td>
<td>0</td>
<td>N/P</td>
<td>N/P</td>
<td>N/P</td>
<td>N/P</td>
<td>5.017</td>
<td>143.254</td>
</tr>
<tr>
<td>kevin</td>
<td>0.104</td>
<td>0</td>
<td>1.480</td>
<td>0.006</td>
<td>0</td>
<td>0.030</td>
<td>0</td>
<td>0</td>
<td>6.088</td>
<td>6.081</td>
<td>1</td>
</tr>
</tbody>
</table>

Charts for selected workload cluster groups
WebSphere – a first class OPM citizen

WebSphere support has a built-in support for OPM (starting with V6.0.21), allowing to:

- identify problems with WAS connection pool
- identify differences in the configuration of nodes in a WAS cluster
- check if a node in a WAS cluster has a system or network problem

### Client Comparison

<table>
<thead>
<tr>
<th>Client Host Name or IP Address</th>
<th>Time of First Connection</th>
<th>Network Time</th>
<th>Client Time</th>
<th>Currently Used Connections</th>
<th>Connection Pool Size</th>
<th>Maximum Connection Wait Time</th>
<th>JRE Version</th>
<th>Operating System</th>
<th>Database Driver Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>GoSales2.ibm.com</td>
<td>03/31 13:50.8...</td>
<td>11:04.0...</td>
<td>47</td>
<td>50</td>
<td>57.67</td>
<td></td>
<td></td>
<td></td>
<td>3.58.82</td>
</tr>
<tr>
<td>GoSales1.ibm.com</td>
<td>04/07 17:17:39</td>
<td>25.336</td>
<td>52</td>
<td>100</td>
<td>0.34</td>
<td></td>
<td></td>
<td></td>
<td>3.58.82</td>
</tr>
</tbody>
</table>
The Future of DB2 Connect: Application Infrastructure

Curt Cotner

IBM Silicon Valley Lab
cotner@us.ibm.com