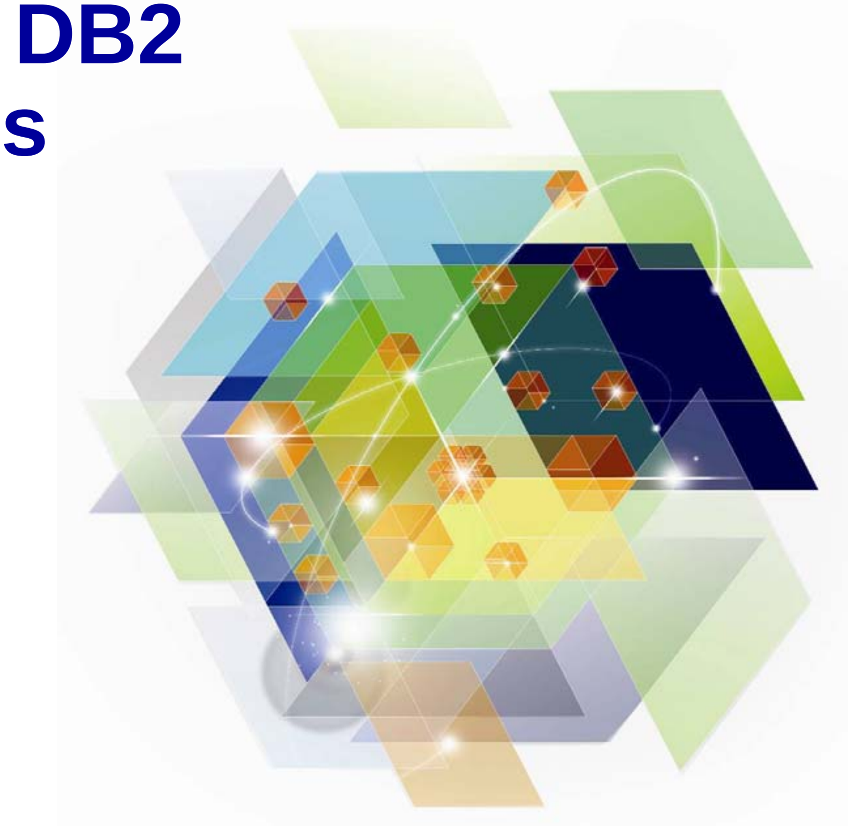
Getting The Most Out of DB2 in Your New Applications



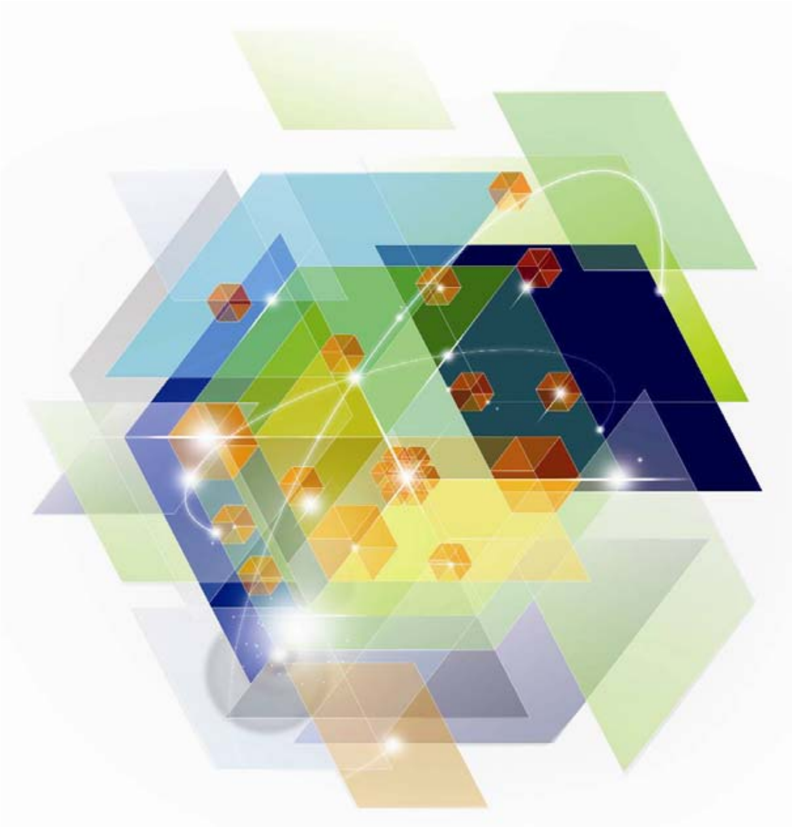
Curt Cotner

DB2 Development,
IBM Fellow and VP,
IBM Software Group

Disclaimer

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion. Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision. The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

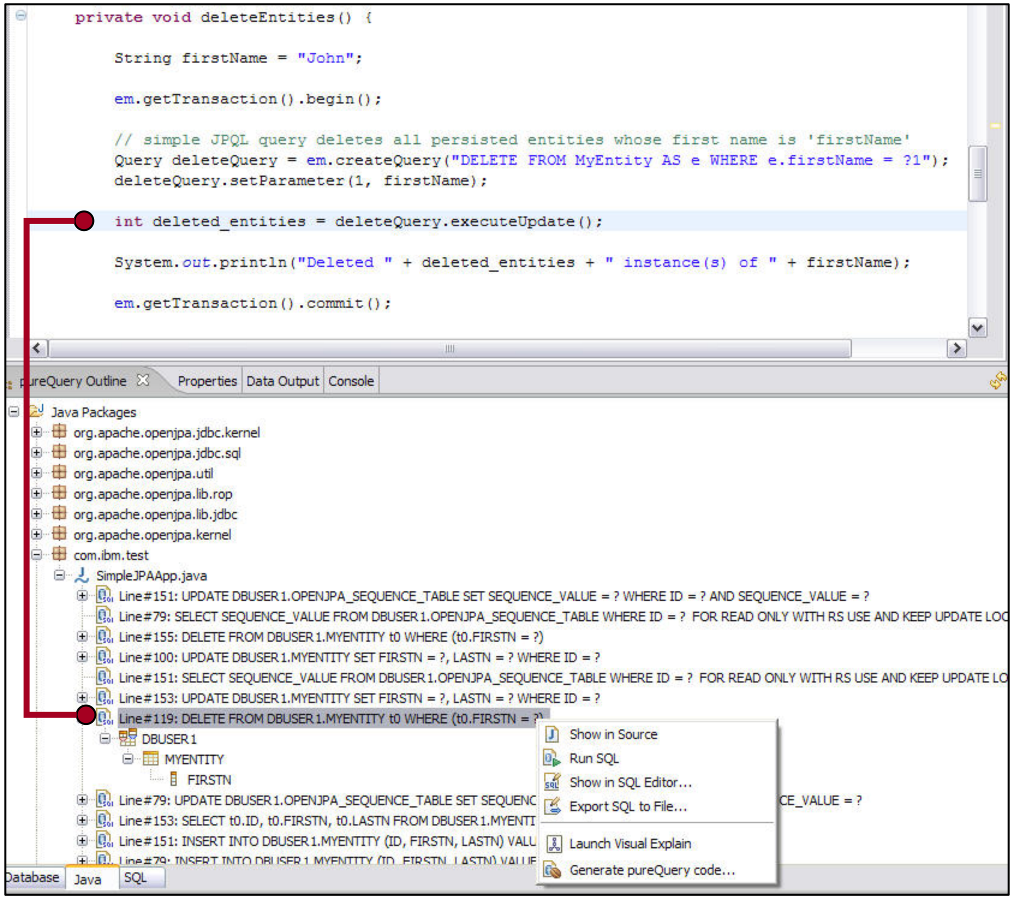
Optimizing Your Java and WebSphere Applications with Data Studio and Optim Tools



Data Studio -- pureQuery tooling is in the box!

Speed up problem isolation for developers – even when using frameworks

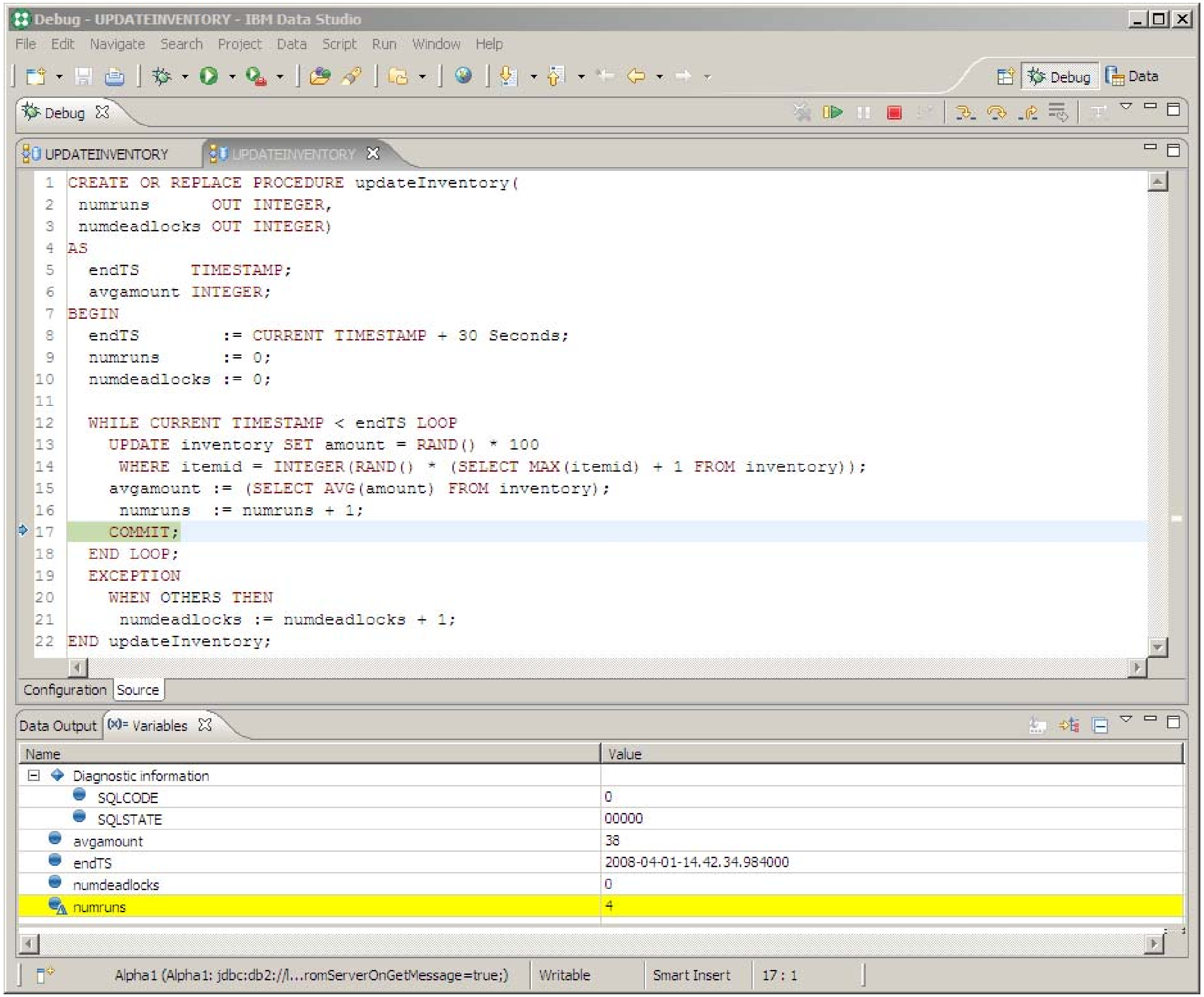
- **Capture application-SQL-data object correlation (with or without the source code)**
- **Trace SQL statements to using code for faster problem isolation**
- **Enhance impact analysis identifying application code impacted due to database changes**
- **Answer “Where used” questions like “Where is this column used within the application?”**
- **Use with modern Java frameworks e.g. Hibernate, Spring, iBatis, OpenJPA**



```
    private void deleteEntities() {

        String firstName = "John";

        em.getTransaction().begin();

        // simple JPQL query deletes all persisted entities whose first name is 'firstName'
        Query deleteQuery = em.createQuery("DELETE FROM MyEntity AS e WHERE e.firstName = ?1");
        deleteQuery.setParameter(1, firstName);

        int deleted_entities = deleteQuery.executeUpdate();

        System.out.println("Deleted " + deleted_entities + " instance(s) of " + firstName);

        em.getTransaction().commit();
```

```
Java Packages
  org.apache.openjpa.jdbc.kernel
  org.apache.openjpa.jdbc.sql
  org.apache.openjpa.util
  org.apache.openjpa.lib.rop
  org.apache.openjpa.lib.jdbc
  org.apache.openjpa.kernel
  com.ibm.test
    SimpleJPAApp.java
      Line#151: UPDATE DBUSER1.OPENJPA_SEQUENCE_TABLE SET SEQUENCE_VALUE = ? WHERE ID = ? AND SEQUENCE_VALUE = ?
      Line#79: SELECT SEQUENCE_VALUE FROM DBUSER1.OPENJPA_SEQUENCE_TABLE WHERE ID = ?  FOR READ ONLY WITH RS USE AND KEEP UPDATE LOC
      Line#155: DELETE FROM DBUSER1.MYENTITY t0 WHERE (t0.FIRSTN = ?)
      Line#100: UPDATE DBUSER1.MYENTITY SET FIRSTN = ?, LASTN = ? WHERE ID = ?
      Line#151: SELECT SEQUENCE_VALUE FROM DBUSER1.OPENJPA_SEQUENCE_TABLE WHERE ID = ?  FOR READ ONLY WITH RS USE AND KEEP UPDATE LO
      Line#153: UPDATE DBUSER1.MYENTITY SET FIRSTN = ?, LASTN = ? WHERE ID = ?
      Line#119: DELETE FROM DBUSER1.MYENTITY t0 WHERE (t0.FIRSTN = ?)
        DBUSER1
          MYENTITY
            FIRSTN
      Line#79: UPDATE DBUSER1.OPENJPA_SEQUENCE_TABLE SET SEQUENC                                  CE_VALUE = ?
      Line#153: SELECT t0.ID, t0.FIRSTN, t0.LASTN FROM DBUSER1.MYENTI
      Line#151: INSERT INTO DBUSER1.MYENTITY (ID, FIRSTN, LASTN) VALU
      Line#79: INSERT INTO DBUSER1.MYENTITY (ID, FIRSTN, LASTN) VALUE
```

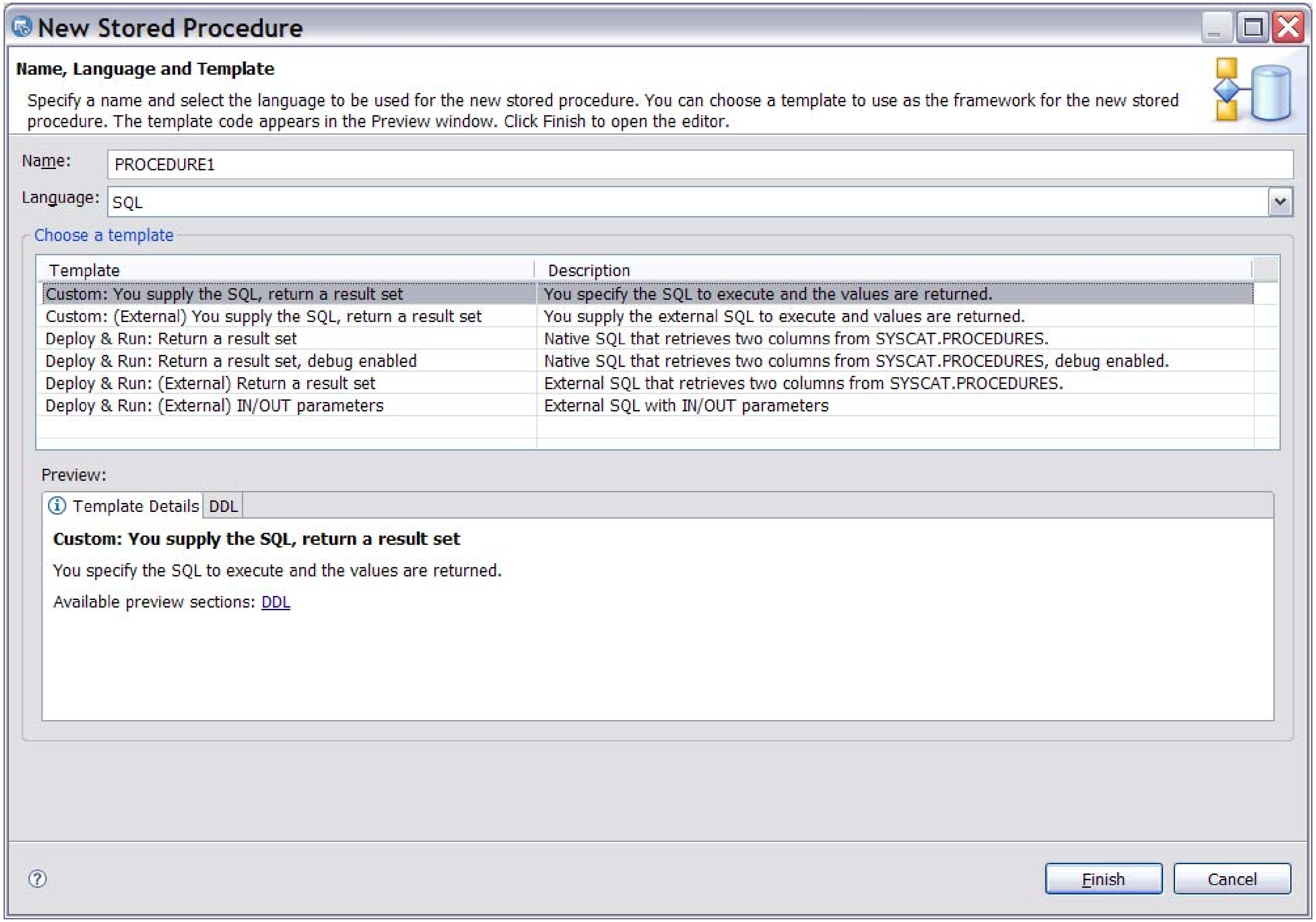
Data Studio -- Code/Debug Oracle PL/SQL or Sybase T-SQL


Debug - UPDATEINVENTORY - IBM Data Studio
File Edit Navigate Search Project Data Script Run Window Help
Debug
Data
Debug
UPDATEINVENTORY
UPDATEINVENTORY
Configuration
Source
Data Output
Variables
Alpha1 (Alpha1: jdbc:db2://l...romServerOnGetMessage=true;)
Writable
Smart Insert
17 : 1

```
CREATE OR REPLACE PROCEDURE updateInventory(
 numruns      OUT INTEGER,
 numdeadlocks OUT INTEGER)
AS
  endTS     TIMESTAMP;
  avgamount INTEGER;
BEGIN
  endTS        := CURRENT TIMESTAMP + 30 Seconds;
  numruns      := 0;
  numdeadlocks := 0;

  WHILE CURRENT TIMESTAMP < endTS LOOP
    UPDATE inventory SET amount = RAND() * 100
     WHERE itemid = INTEGER(RAND() * (SELECT MAX(itemid) + 1 FROM inventory));
    avgamount := (SELECT AVG(amount) FROM inventory);
     numruns  := numruns + 1;
    COMMIT;
  END LOOP;
  EXCEPTION
    WHEN OTHERS THEN
     numdeadlocks := numdeadlocks + 1;
END updateInventory;
```

Name	Value
⊟ Diagnostic information	
SQLCODE	0
SQLSTATE	00000
avgamount	38
endTS	2008-04-01-14.42.34.984000
numdeadlocks	0
numruns	4

Data Studio -- New Routine Creation Wizard



New Stored Procedure

Name, Language and Template

Specify a name and select the language to be used for the new stored procedure. You can choose a template to use as the framework for the new stored procedure. The template code appears in the Preview window. Click Finish to open the editor.

Name: PROCEDURE1

Language: SQL

Choose a template

Template	Description
Custom: You supply the SQL, return a result set	You specify the SQL to execute and the values are returned.
Custom: (External) You supply the SQL, return a result set	You supply the external SQL to execute and values are returned.
Deploy & Run: Return a result set	Native SQL that retrieves two columns from SYSCAT.PROCEDURES.
Deploy & Run: Return a result set, debug enabled	Native SQL that retrieves two columns from SYSCAT.PROCEDURES, debug enabled.
Deploy & Run: (External) Return a result set	External SQL that retrieves two columns from SYSCAT.PROCEDURES.
Deploy & Run: (External) IN/OUT parameters	External SQL with IN/OUT parameters

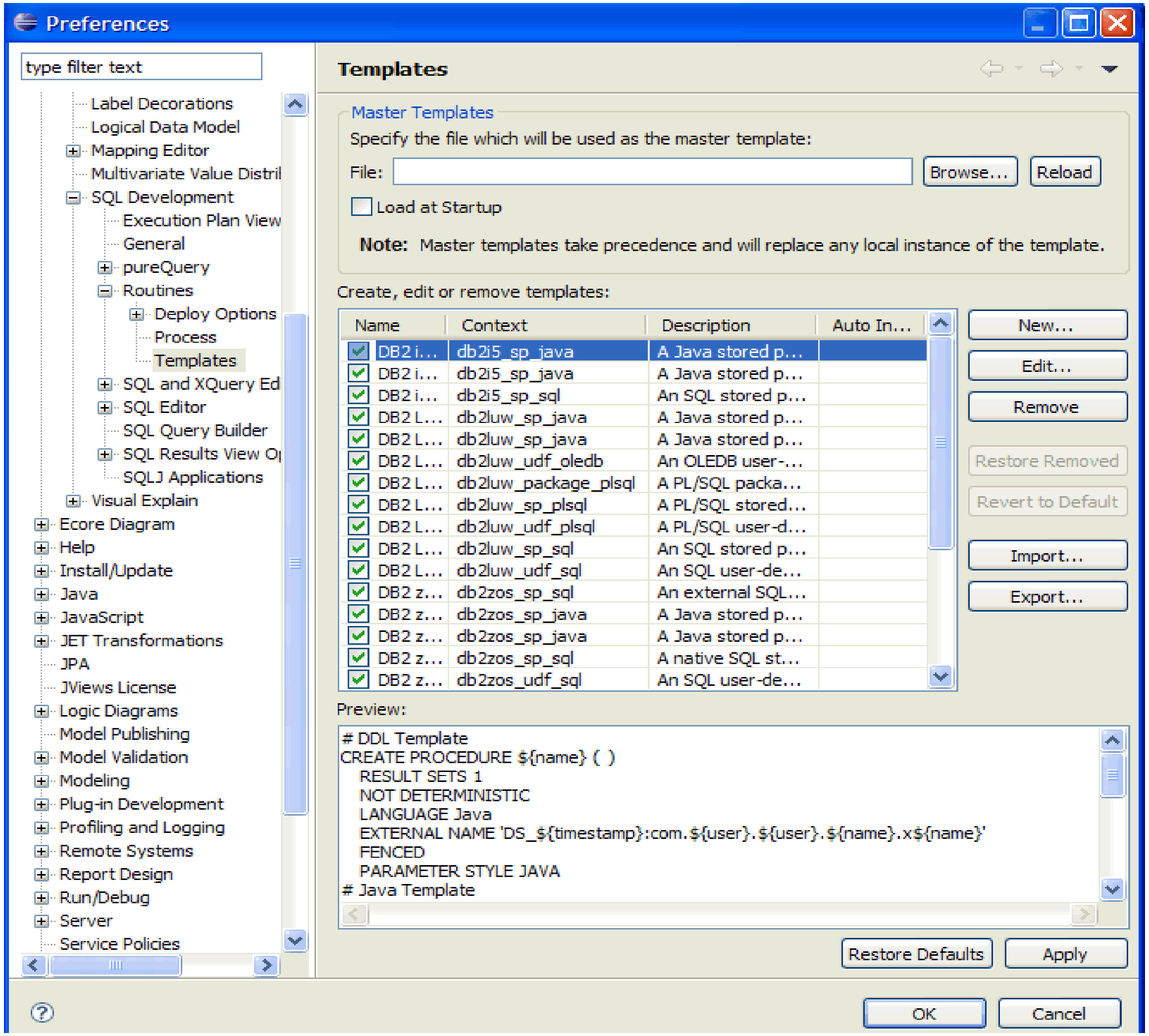
Preview:

Custom: You supply the SQL, return a result set

You specify the SQL to execute and the values are returned.

Available preview sections: DDL

Data Studio -- Templates Management



Specify the file which will be used as the master template:

☐ Load at Startup

Note: Master templates take precedence and will replace any local instance of the template.

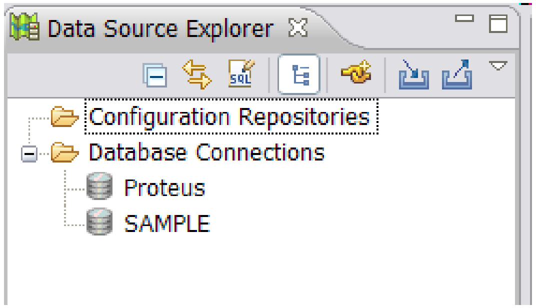
Create, edit or remove templates:

Name	Context	Description	Auto In...
☑ DB2 i...	db2i5_sp_java	A Java stored p...	
☑ DB2 i...	db2i5_sp_java	A Java stored p...	
☑ DB2 i...	db2i5_sp_sql	An SQL stored p...	
☑ DB2 L...	db2luw_sp_java	A Java stored p...	
☑ DB2 L...	db2luw_sp_java	A Java stored p...	
☑ DB2 L...	db2luw_udf_oledb	An OLEDB user-...	
☑ DB2 L...	db2luw_package_plsql	A PL/SQL packa...	
☑ DB2 L...	db2luw_sp_plsql	A PL/SQL stored...	
☑ DB2 L...	db2luw_udf_plsql	A PL/SQL user-d...	
☑ DB2 L...	db2luw_sp_sql	An SQL stored p...	
☑ DB2 L...	db2luw_udf_sql	An SQL user-de...	
☑ DB2 z...	db2zos_sp_sql	An external SQL...	
☑ DB2 z...	db2zos_sp_java	A Java stored p...	
☑ DB2 z...	db2zos_sp_java	A Java stored p...	
☑ DB2 z...	db2zos_sp_sql	A native SQL st...	
☑ DB2 z...	db2zos_udf_sql	An SQL user-de...	

Preview:

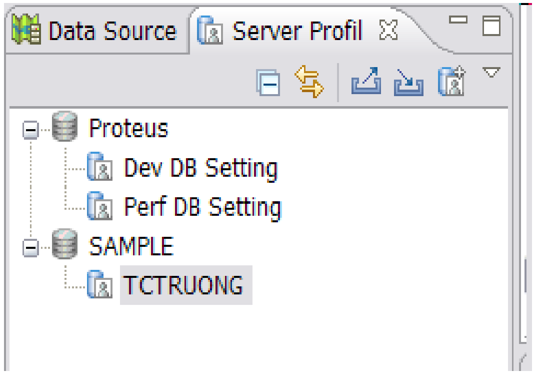
```
# DDL Template
CREATE PROCEDURE ${name} ( )
   RESULT SETS 1
   NOT DETERMINISTIC
   LANGUAGE Java
   EXTERNAL NAME 'DS_${timestamp}:com.${user}.${user}.${name}.x${name}'
   FENCED
   PARAMETER STYLE JAVA
# Java Template
```

Data Studio -- Server Profile Management



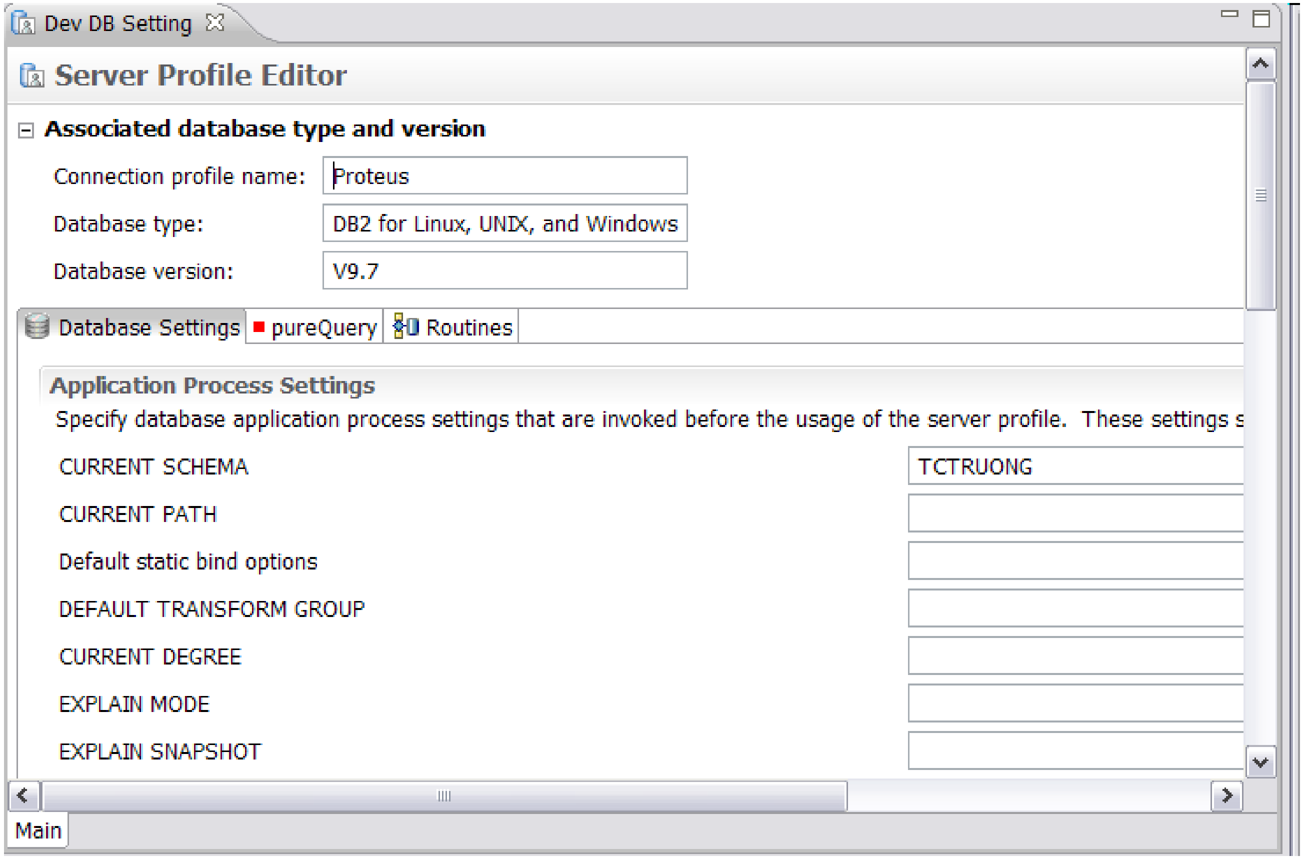
Data Source Explorer

- Configuration Repositories
- Database Connections
  - Proteus
  - SAMPLE



Data Source | Server Profil

- Proteus
  - Dev DB Setting
  - Perf DB Setting
- SAMPLE
  - TCTRUONG



Dev DB Setting

Server Profile Editor

Associated database type and version

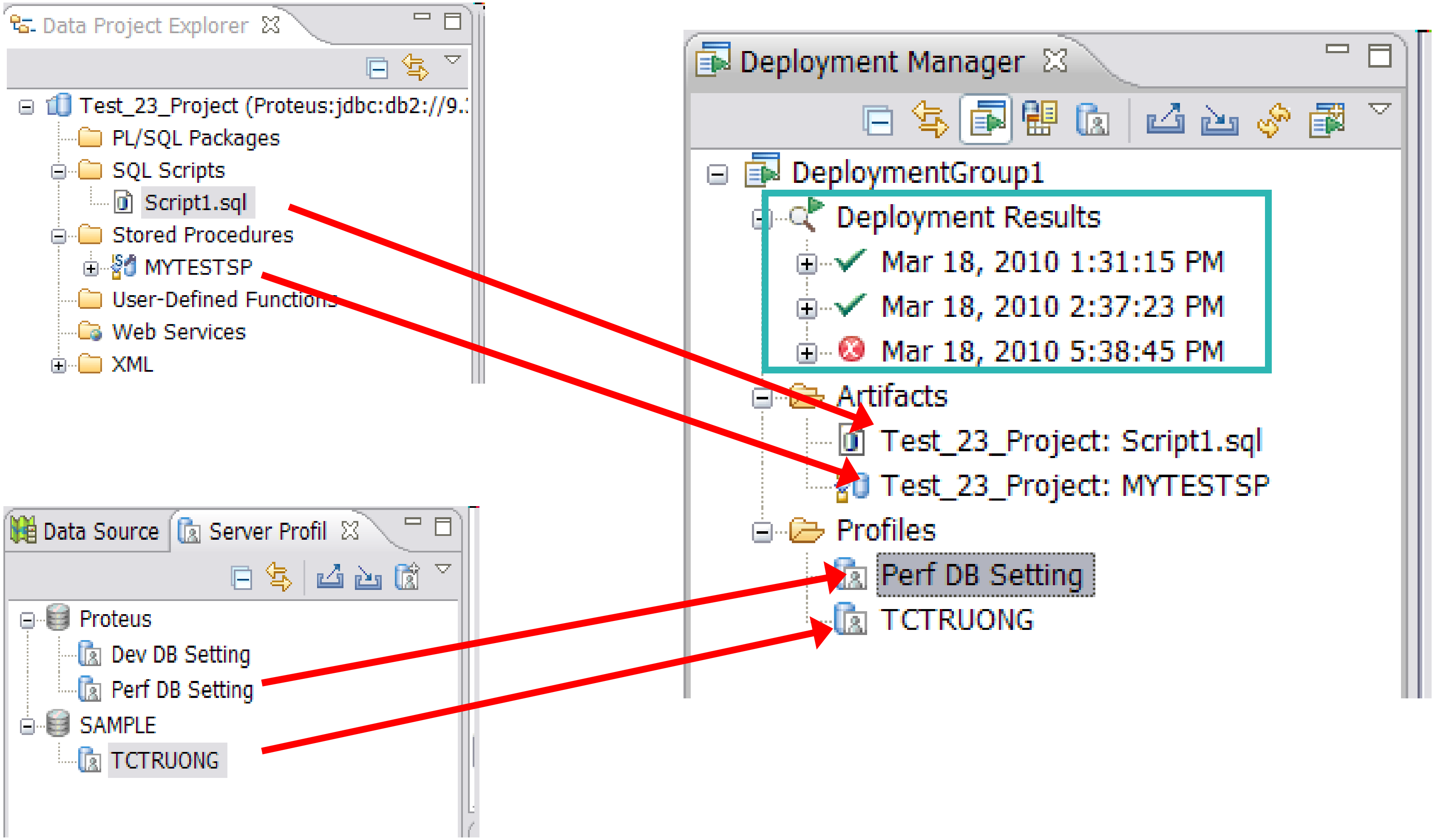
Connection profile name:	Proteus
Database type:	DB2 for Linux, UNIX, and Windows
Database version:	V9.7

Application Process Settings

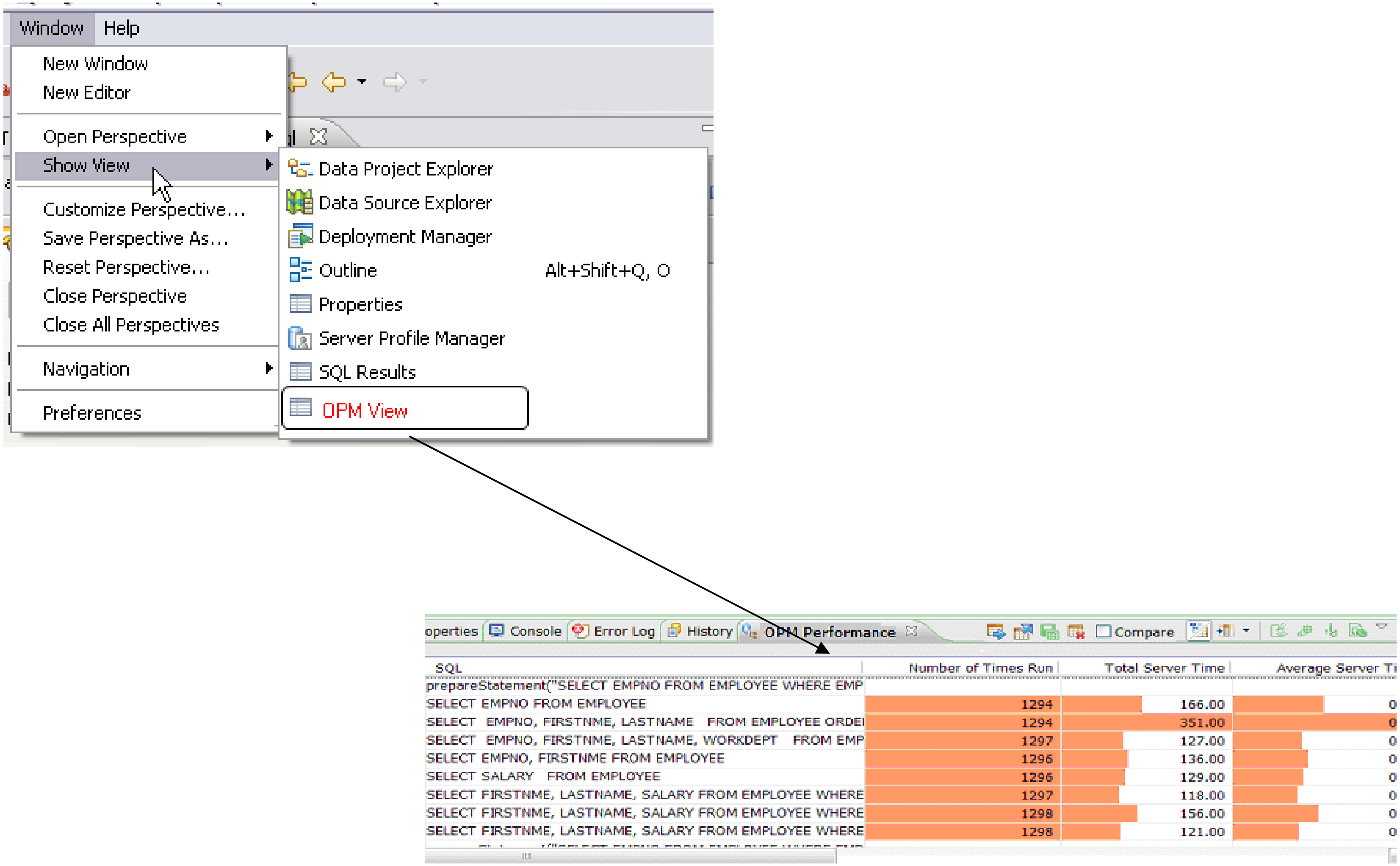
Specify database application process settings that are invoked before the usage of the server profile. These settings s

CURRENT SCHEMA	TCTRUONG
CURRENT PATH	
Default static bind options	
DEFAULT TRANSFORM GROUP	
CURRENT DEGREE	
EXPLAIN MODE	
EXPLAIN SNAPSHOT	

Data Studio -- Deployment Management



Data Studio -- OPM Performance View



SQL	Number of Times Run	Total Server Time	Average Server Ti
prepareStatement("SELECT EMPNO FROM EMPLOYEE WHERE EMP			
SELECT EMPNO FROM EMPLOYEE	1294	166.00	0
SELECT EMPNO, FIRSTNME, LASTNAME FROM EMPLOYEE ORDEI	1294	351.00	0
SELECT EMPNO, FIRSTNME, LASTNAME, WORKDEPT FROM EMP	1297	127.00	0
SELECT EMPNO, FIRSTNME FROM EMPLOYEE	1296	136.00	0
SELECT SALARY FROM EMPLOYEE	1296	129.00	0
SELECT FIRSTNME, LASTNAME, SALARY FROM EMPLOYEE WHERE	1297	118.00	0
SELECT FIRSTNME, LASTNAME, SALARY FROM EMPLOYEE WHERE	1298	156.00	0
SELECT FIRSTNME, LASTNAME, SALARY FROM EMPLOYEE WHERE	1298	121.00	0

Data Studio -- OPM Performance View

Table Columns

Golf Score
SQL statement
Annotation
Inputs for host variables
Total Server Time
Average Server Time
Number of Rows
Number of Rows Examined
Average Number of Row Returned
CPU time
Number of Sorts
Number of RSCANs
Number of ISCANs
Number of physical IOs
Number of logocial IOs

Table Actions

Export – Exports the data to file
Remove All – clears the table of all rows

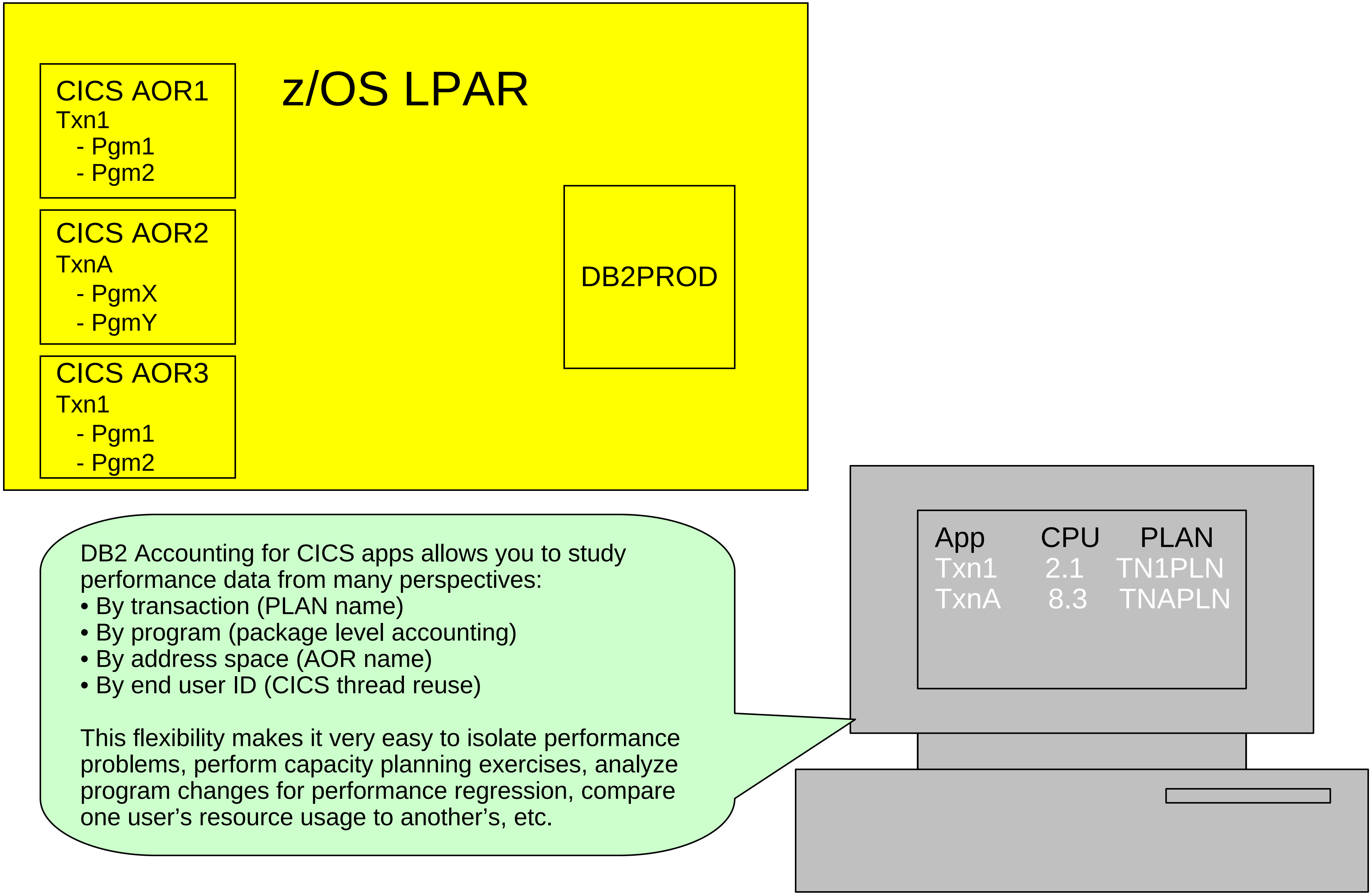
Row Actions

Open in SQL Editor – opens SQL editor with selected SQL statements
Filter – Hides all but the selected rows
Remove – removes selected row(s)

pureQuery Runtime – every Java application benefits!

- **JDBC – acceleration for any JDBC application**
  - Convert dynamic SQL to static SQL
  - Replace problem queries without changing the source
  - Remove literals from SQL to get better statement cache hit ratios
- **Hibernate/OpenJPA/iBatis – acceleration for persistence layers**
  - Improved SQL “batch” peformance
  - Auto-tuning of Hibernate and OpenJPA peristence options
- **SQL-friendly APIs for OO access to relational**
  - Object to relational mapping
  - APIs that can be tailored to return XML, JSON, arrays, etc.
- **Improved management, monitoring, problem determination**
  - Tracks SQL back to the Java class file and line number
  - Enables performance monitors to report by application name
- **Provides the foundation for improved developer tooling**
  - Syntax assist, code generation, performance reporting, etc.

What’s so Great About DB2 Accounting for CICS Apps?



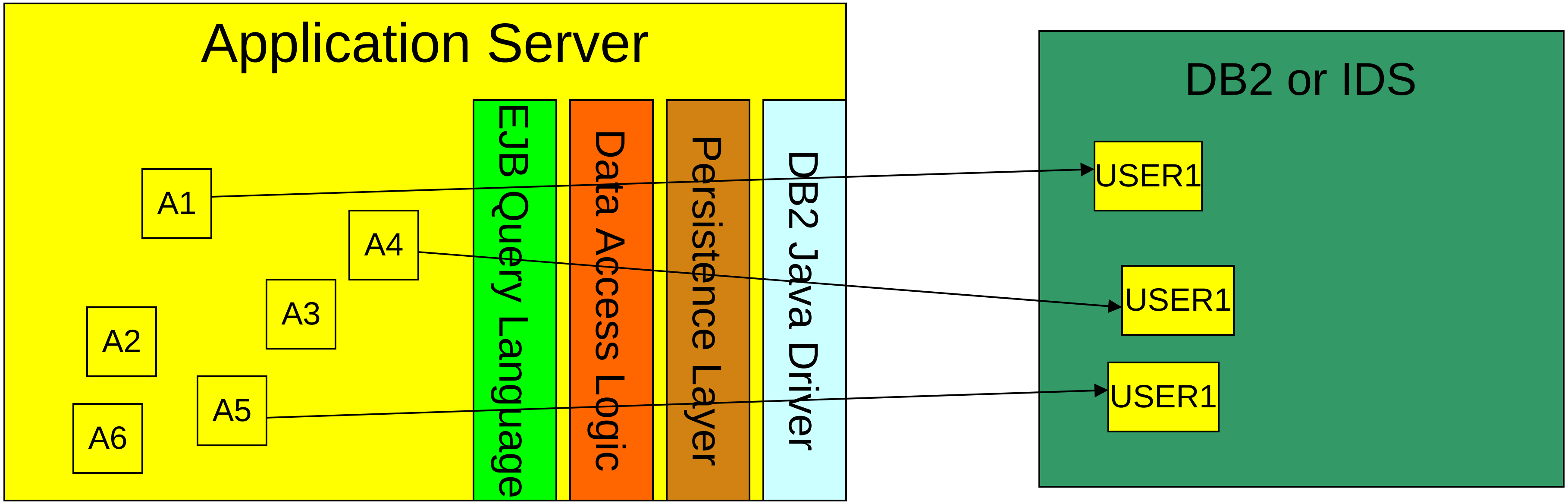
App	CPU	PLAN
Txn1	2.1	TN1PLN
TxnA	8.3	TNAPLN

DB2 Accounting for CICS apps allows you to study performance data from many perspectives:

- By transaction (PLAN name)
- By program (package level accounting)
- By address space (AOR name)
- By end user ID (CICS thread reuse)

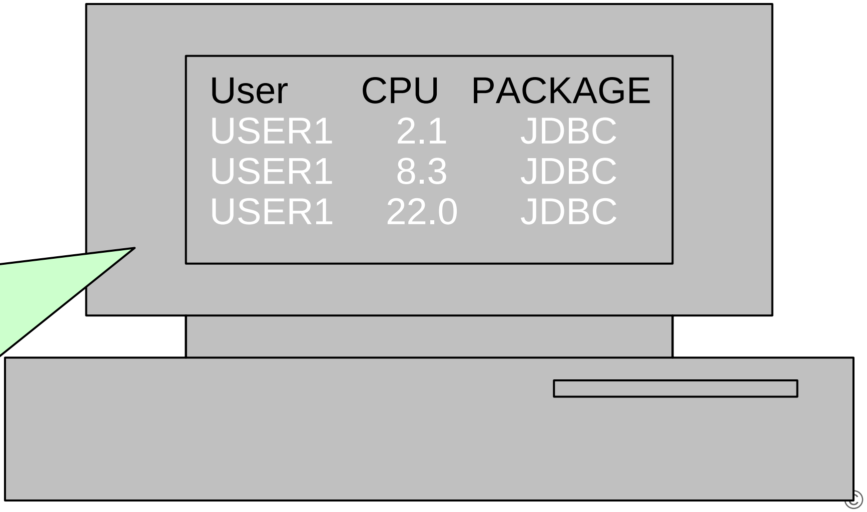
This flexibility makes it very easy to isolate performance problems, perform capacity planning exercises, analyze program changes for performance regression, compare one user’s resource usage to another’s, etc.

JDBC Performance Reporting and Problem Determination – Before pureQuery



What is visible to the DBA?
- IP address of WAS app server
- Connection pooling userid for WAS
- app is running JDBC or CLI

What is not known by the DBA?
- which app 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?
- etc.



User	CPU	PACKAGE
USER1	2.1	JDBC
USER1	8.3	JDBC
USER1	22.0	JDBC

What’s so Great About Optim pureQuery Accounting for WebSphere Applications?

z/OS LPAR

CICS AOR2
TxnA (PLANA)
- PgmX
- PgmY

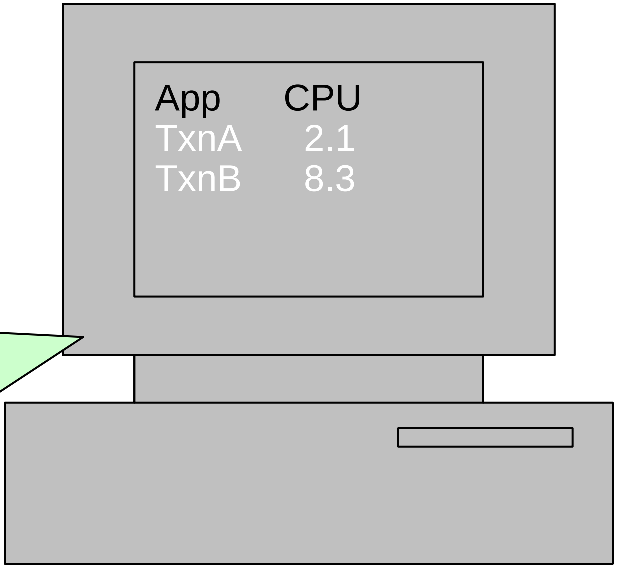
Unix or Windows

WAS 21.22.3.4
TxnA (Set Client App=TxnA)
- ClassX
- ClassY

Data Studio and pureQuery provide the same granularity for reporting WebSphere’s DB2 resources that we have with CICS:

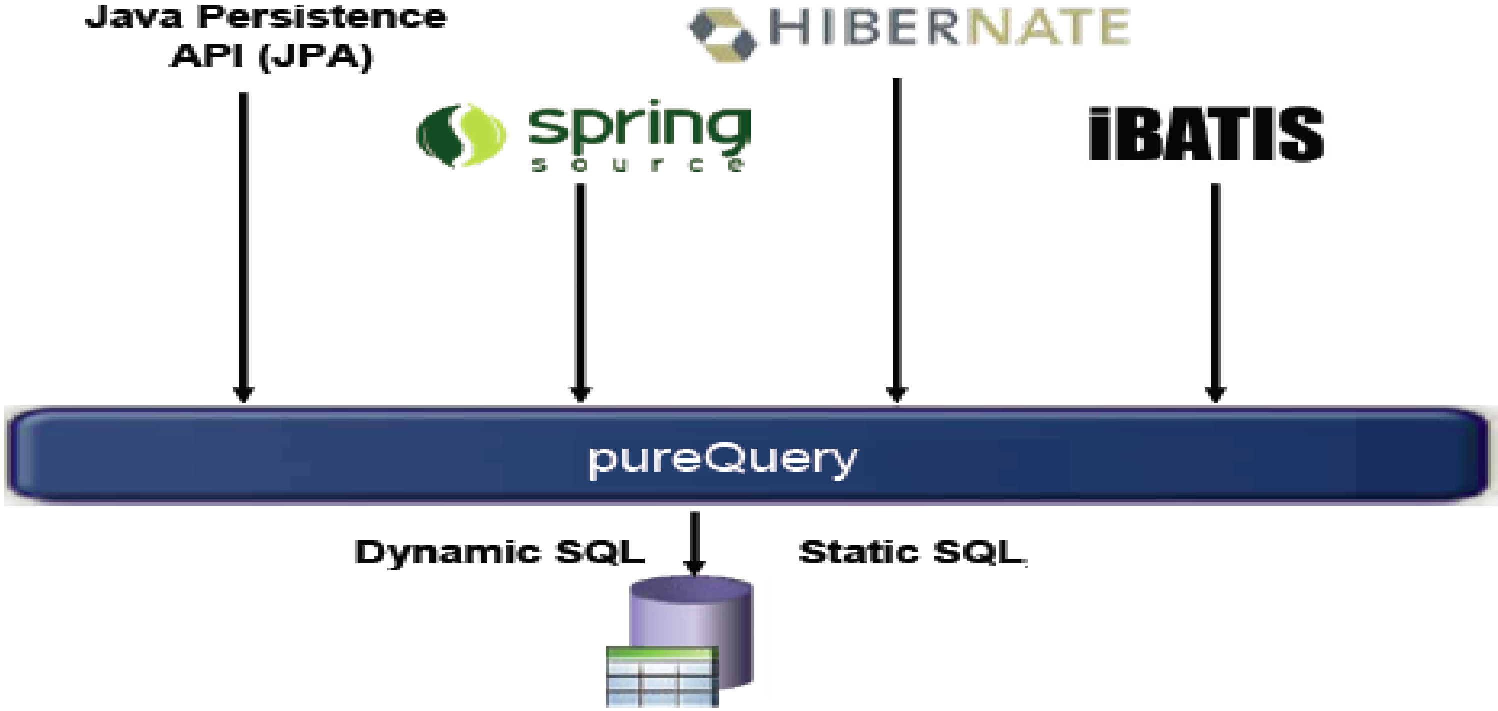
- By transaction (Set Client Application name)
- By class name (program - package level accounting)
- By address space (IP address)
- By end user ID (DB2 trusted context and DB2 Roles)

This flexibility makes it very easy to isolate performance problems, perform capacity planning exercises, analyze program changes for performance regression, compare one user’s resource usage to another’s, etc.



App	CPU
TxnA	2.1
TxnB	8.3

DB2 Java Data Access Frameworks Acceleration

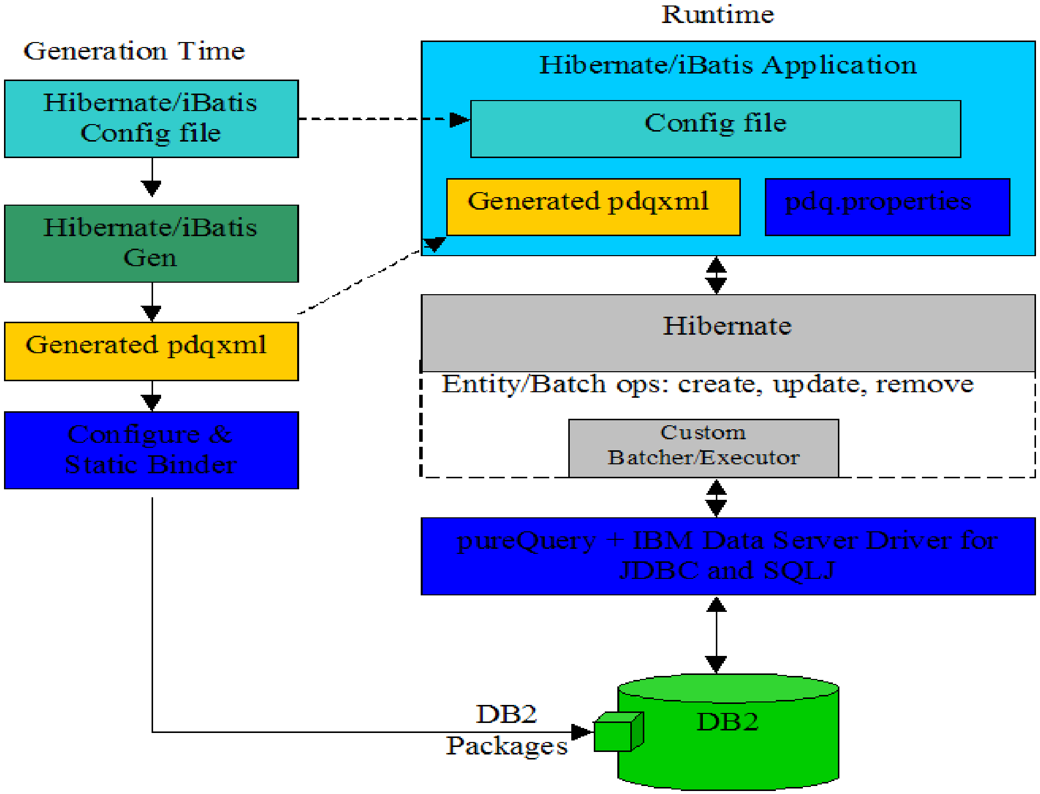


http://www.ibm.com/developerworks/data/library/techarticle/dm-1008hibernateibatispurequery1/index.html?ca=dnw-1133&ca=dth-i

http://www.ibm.com/developerworks/data/library/techarticle/dm-1009hibernateibatispurequery2/index.html

http://www.ibm.com/developerworks/data/tutorials/dm0806hsing/index.html

Accelerate Java frameworks: Hibernate & iBatis


Runtime
Generation Time
Hibernate/iBatis Config file
Hibernate/iBatis Application
Config file
Generated pdqxml
pdq.properties
Hibernate/iBatis Gen
Generated pdqxml
Hibernate
Entity/Batch ops: create, update, remove
Custom Batcher/Executor
Configure & Static Binder
pureQuery + IBM Data Server Driver for JDBC and SQLJ
DB2 Packages
DB2

Improve performance with heterogeneous batching & Static Execution

rack SQL requests back to the framework query, including java source file/line #

Object/Relational Mapping

pureQuery can monitor your Java application’s object access patterns and automatically select the optimal eager/lazy fetch setting for each SQL statement!!!

```
class Customer
{   public String Name;
    public String mailingAddress;
    public String daytimePhone;
    public Order[] recentOrders;
    public Complaint[] complaintHistory
  …
}
```

Table	Column	Type
CUST	NAME	CHAR(64)
CUST	ADDRESS	CHAR(128)
CUST	PHONE_NUM	CHAR(10)

Table	Column	Type
COMPLAINTS	CUST_NAME	CHAR(64)
COMPLAINTS	COMP_ID	CHAR(18)
COMPLAINTS	DESC	VARCHAR(32K)

Table	Column	Type
ORDERS	CUST_NAME	CHAR(64)
ORDERS	ORDER_NUM	CHAR(12)
ORDERS	DATE_ORD	DATE

Table	Column	Type
CREDIT_DATA	CUST_NAME	CHAR(64)
CREDIT_DATA	CARD_NUM	CHAR(18)
CREDIT_DATA	VALID_UNTIL	DATE

Table	Column	Type
ORDER_ITEMS	ORDER_NUM	CHAR(12)
ORDER_ITEMS	ITEM	CHAR(128)
ORDER_ITEMS	QUANTITY	SMALLINT

Eager vs. Lazy Fetch

“Select object(customer) WHERE…”

```
class Customer
{   public String Name;
    public String mailingAddress;
    public String daytimePhone;
    public Order[] recentOrders;
    public Complaint[] complaintHistory
  …
}
```

“SELECT CUST.NAME, CUST.ADDRESS … FROM CUST WHERE…”

“SELECT ORDERS.ORDER_NUM …   WHERE …”

“SELECTCOMPLAINTS.COMP_ID … WHERE …”

O

O

O

Column	Type
NAME	CHAR(64)
ADDRESS	CHAR(128)
PHONE_NUM	CHAR(10)

	Column	Type
	CUST_NAME	CHAR(64)
ORDERS	ORDER_NUM	CHAR(12)
ORDERS	DATE_ORD	DATE

COMPLAINTS	DESC	VARCHAR(32K)

Table	Column	Type
CREDIT_DATA	CUST_NAME	CHAR(64)
CREDIT_DATA	CARD_NUM	CHAR(18)
CREDIT_DATA	VALID_UNTIL	DATE

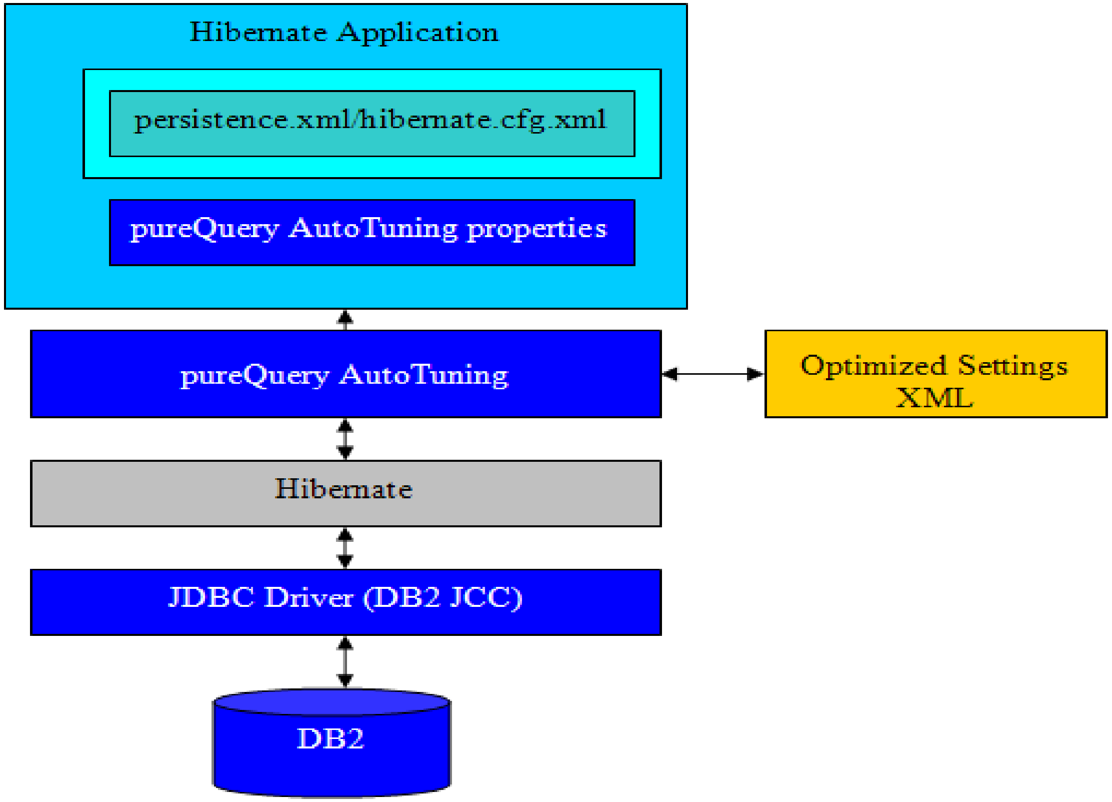
Table	Column	Type
ORDER_ITEMS	ORDER_NUM	CHAR(12)
ORDER_ITEMS	ITEM	CHAR(128)
ORDER_ITEMS	QUANTITY	SMALLINT

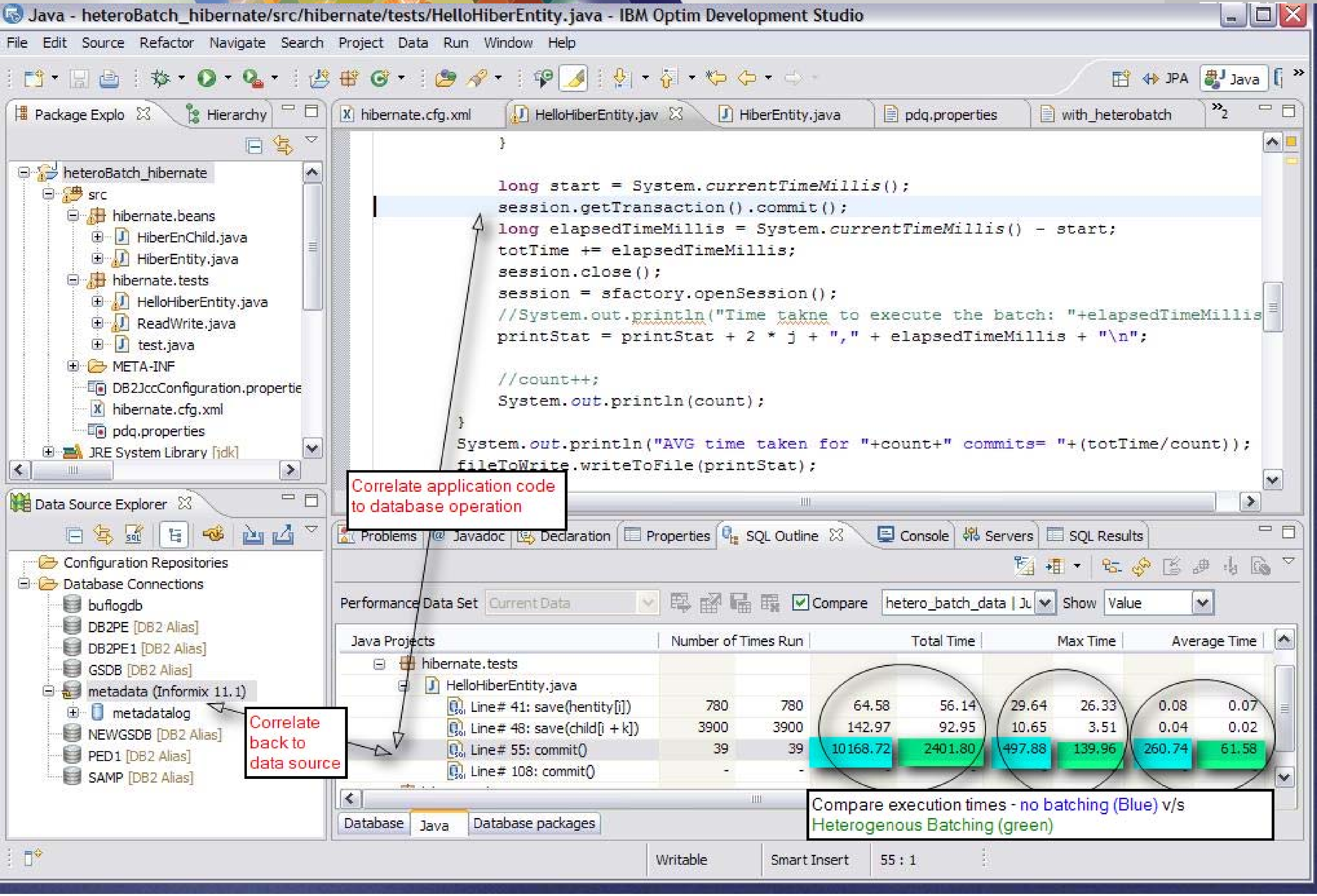
Hibernate AutoTuning

Automatically identify and fix common problems with Java Persistence applications

- hundred’s of SQL per transaction
- tens of unwanted joins per SQL

https://www.ibm.com/services/forms/preLogin.do?source=swg-iopahb





```
        }

        long start = System.currentTimeMillis();
        session.getTransaction().commit();
        long elapsedTimeMillis = System.currentTimeMillis() - start;
        totTime += elapsedTimeMillis;
        session.close();
        session = sfactory.openSession();
        //System.out.println("Time takne to execute the batch: "+elapsedTimeMillis
        printStat = printStat + 2 * j + "," + elapsedTimeMillis + "\n";

        //count++;
        System.out.println(count);
    }
    System.out.println("AVG time taken for "+count+" commits= "+(totTime/count));
    fileToWrite.writeToFile(printStat);
```

Performance Data Set: Current Data ☑ Compare: hetero_batch_data | Ju Show: Value

Java Projects	Number of Times Run		Total Time		Max Time		Average Time	
hibernate.tests								
HelloHiberEntity.java								
Line# 41: save(hentity[i])	780	780	64.58	56.14	29.64	26.33	0.08	0.07
Line# 48: save(child[i + k])	3900	3900	142.97	92.95	10.65	3.51	0.04	0.02
Line# 55: commit()	39	39	10168.72	2401.80	497.88	139.96	260.74	61.58
Line# 108: commit()	-	-	-	-	-	-	-	-

Correlate application code to database operation

Correlate back to data source

Compare execution times - no batching (Blue) v/s Heterogenous Batching (green)

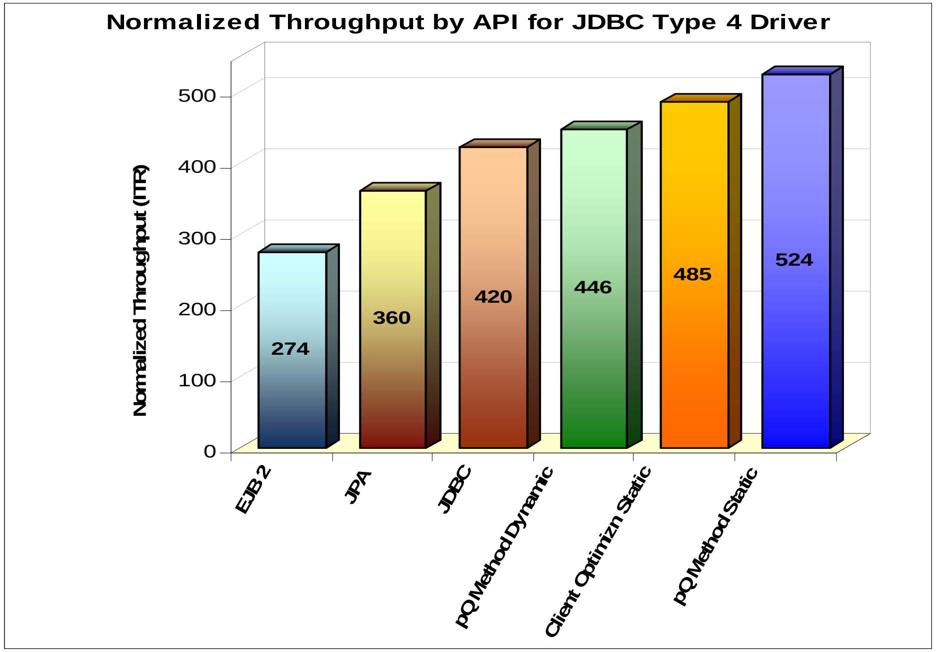
Client Optimization

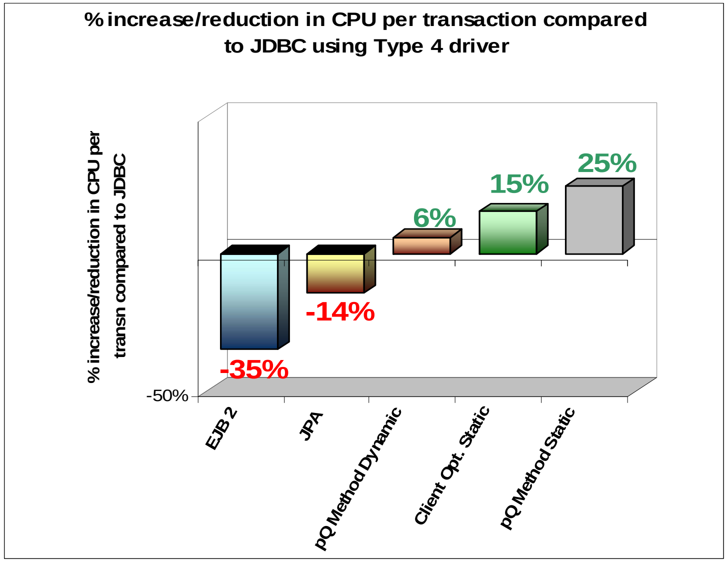
Improve Java data access performance for DB2 – without changing a line of code

- **Captures SQL for Java applications**
  - Custom-developed, framework-based, or packaged applications
- **Bind the SQL for static execution without changing a line of code**
  - New bind tooling included
- **Delivers static SQL execution value to existing DB2 applications**
  - Making response time predictable and stable by locking in the SQL access path pre-execution, rather than re-computing at access time
  - Limiting user access to tables by granting execute privileges on the query packages rather than access privileges on the table
  - Aiding forecasting accuracy and capacity planning by capturing additional workload information based on package statistics
  - Drive down CPU cycles to increase overall capability
- **Choose between dynamic or static execution at deployment time, rather than development time**

Optim pureQuery Runtime for z/OS

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

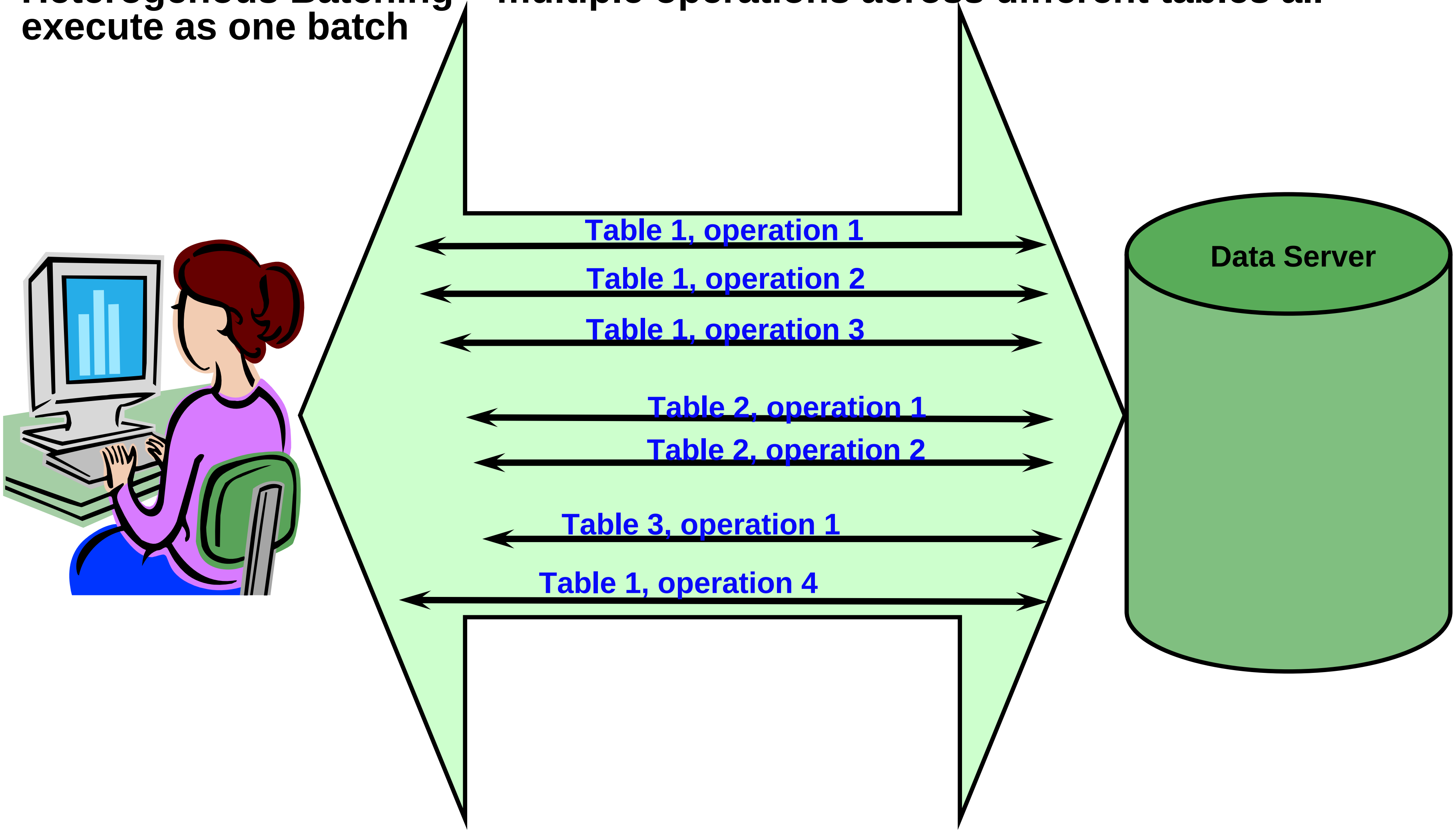




- **IRWW – an OLTP workload,** Type 4 driver
- **Cache hit ratio between 70 and 85%**
- 15% - 25% reduction on CPU per txn **over dynamic JDBC**

What Is Heterogeneous Batching?

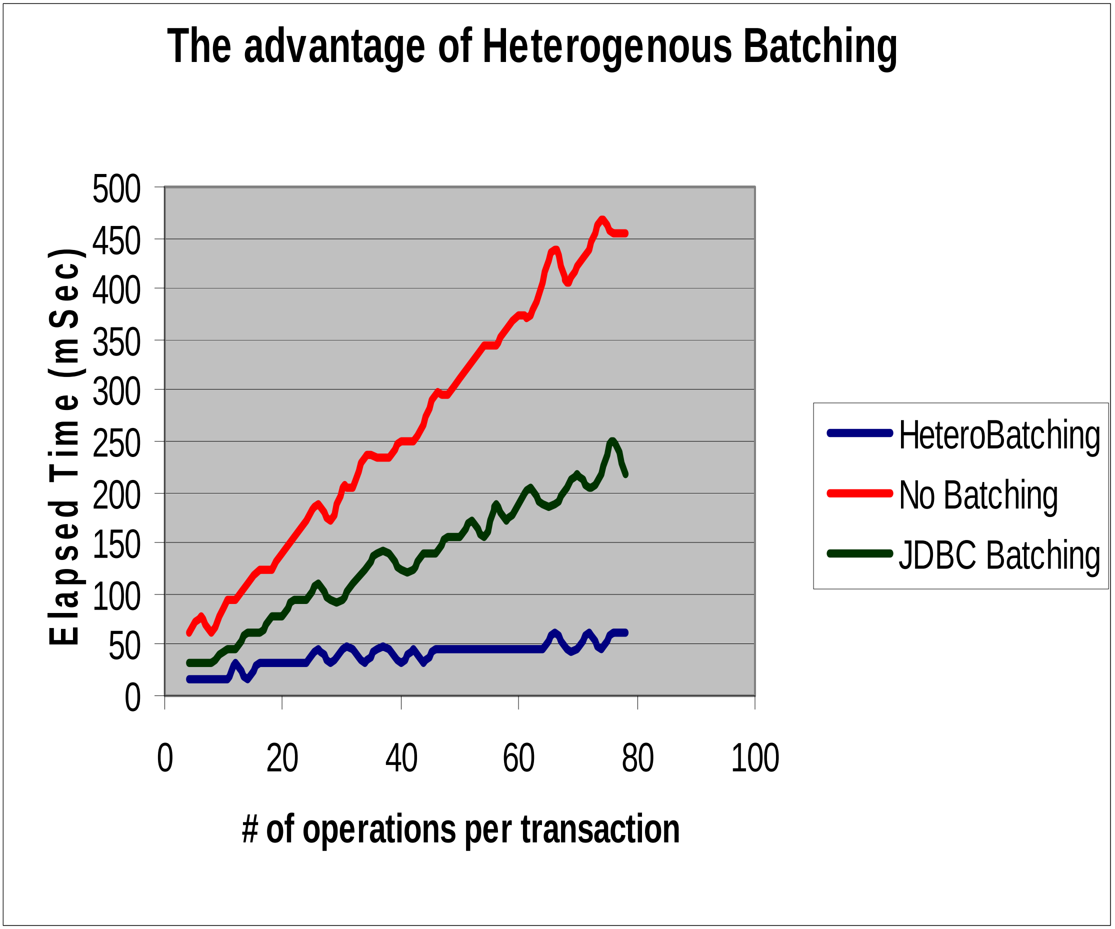
Heterogenous Batching – multiple operations across different tables all execute as one batch



JDBC Batching v/s pureQuery Heterogeneous Batching

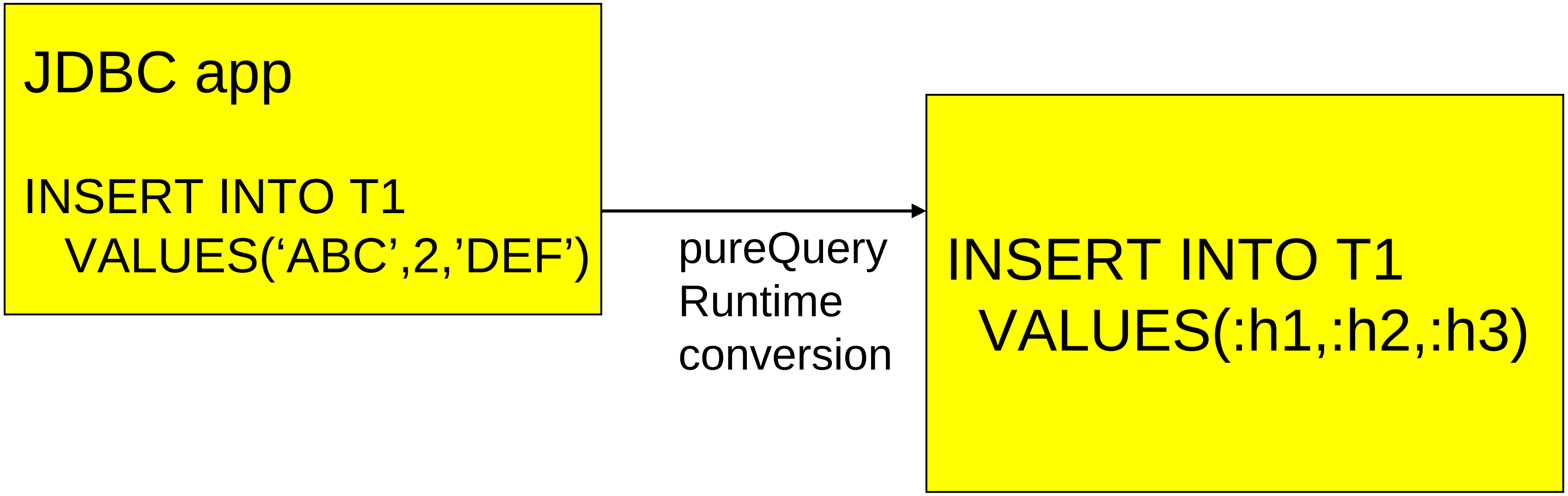
- **JDBC batching used by Hibernate Batcher is currently limited**
  - Cannot batch entities that map to multiple tables
    - Primary and Secondary tables.
    - Inheritance Join and Table per class strategies
  - Cannot batch different operations against same table
    - Field level updates
    - Insert, update
  - Cannot batch different entities
  - Each batch is a message to the database

- **pureQuery heterogeneous batching plug-in for Hibernate on the other hand**
  - Can batch entities that map to multiple tables
  - Can batch different operations against the same table
  - Can batch different entities into a single batch
  - Combines insert, deletes, updates into single batch



*** Preliminary findings based on validation with a test designed to demonstrate heterogeneous batching differences. This is not intended to be a formal benchmark.**

pureQuery – Stripping Literals from SQL



JDBC app

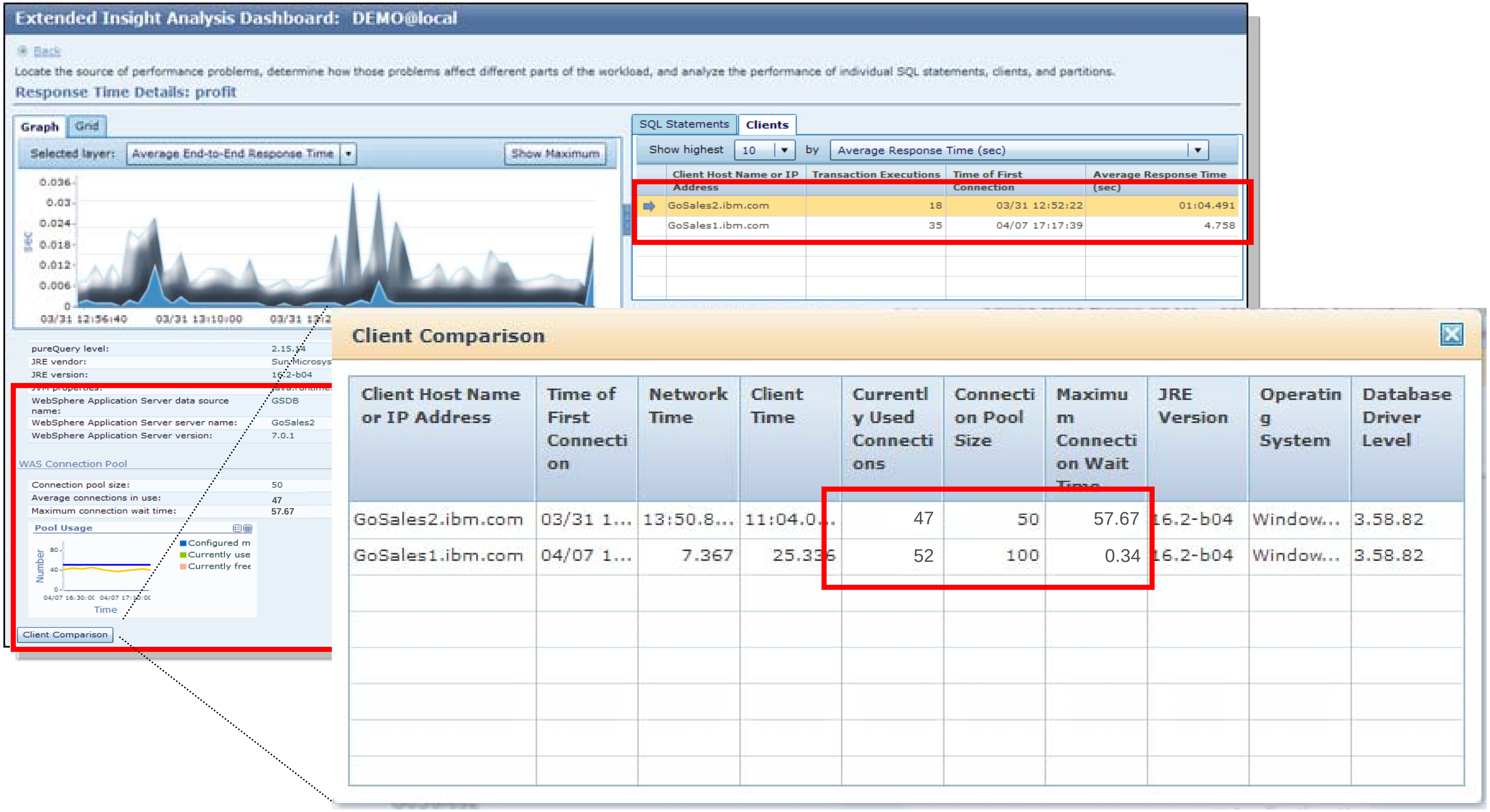
INSERT INTO T1
  VALUES(‘ABC’,2,’DEF’)

pureQuery
Runtime
conversion

INSERT INTO T1
  VALUES(:h1,:h2,:h3)

- pureQuery can identify statements that use no parameter markers, and strip the literals out at runtime
- significant performance gains:
  - less CPU cost at PREPARE
  - better use of dynamic statement cache

WebSphere – a first class OPM citizen



Extended Insight Analysis Dashboard: DEMO@local

Back

Locate the source of performance problems, determine how those problems affect different parts of the workload, and analyze the performance of individual SQL statements, clients, and partitions.

Response Time Details: profit

Show highest 10 by Average Response Time (sec)

	Client Host Name or IP Address	Transaction Executions	Time of First Connection	Average Response Time (sec)
➡	GoSales2.ibm.com	18	03/31 12:52:22	01:04.491
	GoSales1.ibm.com	35	04/07 17:17:39	4.758

pureQuery level:	2.15.14
JRE vendor:	Sun Microsys
JRE version:	16.2-b04
JVM properties:	java.runtime.
WebSphere Application Server data source name:	GSDB
WebSphere Application Server server name:	GoSales2
WebSphere Application Server version:	7.0.1

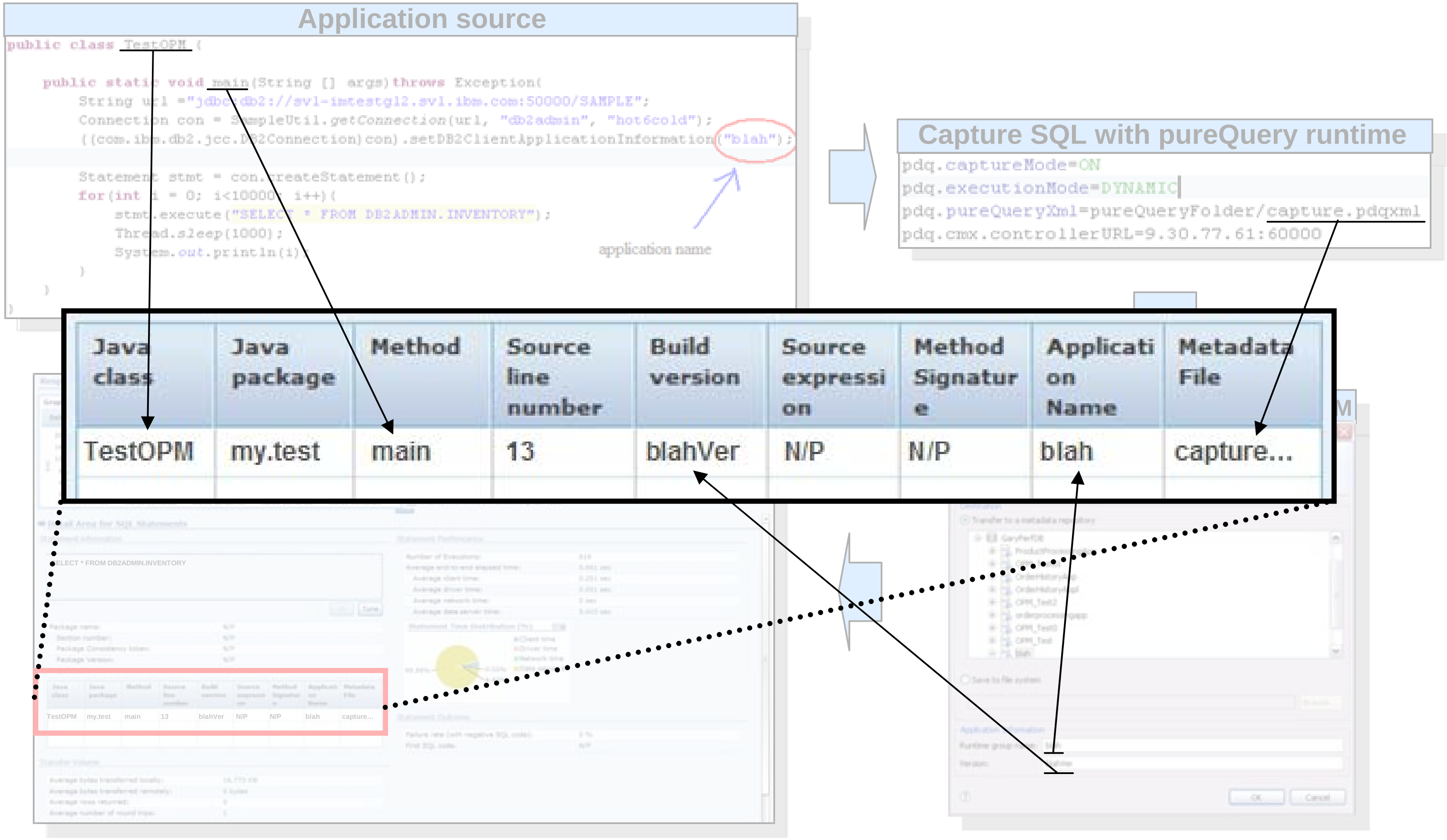
WAS Connection Pool

Connection pool size:	50
Average connections in use:	47
Maximum connection wait time:	57.67

Client Comparison

Client Host Name or IP Address	Time of First Connection	Network Time	Client Time	Currently Used Connections	Connection Pool Size	Maximum Connection Wait Time	JRE Version	Operating System	Database Driver Level
GoSales2.ibm.com	03/31 1...	13:50.8...	11:04.0...	47	50	57.67	16.2-b04	Window...	3.58.82
GoSales1.ibm.com	04/07 1...	7.367	25.336	52	100	0.34	16.2-b04	Window...	3.58.82

OPM can tell you where the query came from ...



Application source

```
public class TestOPM {

    public static void main(String [] args)throws Exception{
        String url ="jdbc:db2://svl-imtestg12.svl.ibm.com:50000/SAMPLE";
        Connection con = SampleUtil.getConnection(url, "db2admin", "hot6cold");
        ((com.ibm.db2.jcc.DB2Connection)con).setDB2ClientApplicationInformation("blah");

        Statement stmt = con.createStatement();
        for(int i = 0; i<10000; i++){
            stmt.execute("SELECT * FROM DB2ADMIN.INVENTORY");
            Thread.sleep(1000);
            System.out.println(i);
        }
    }
}
```

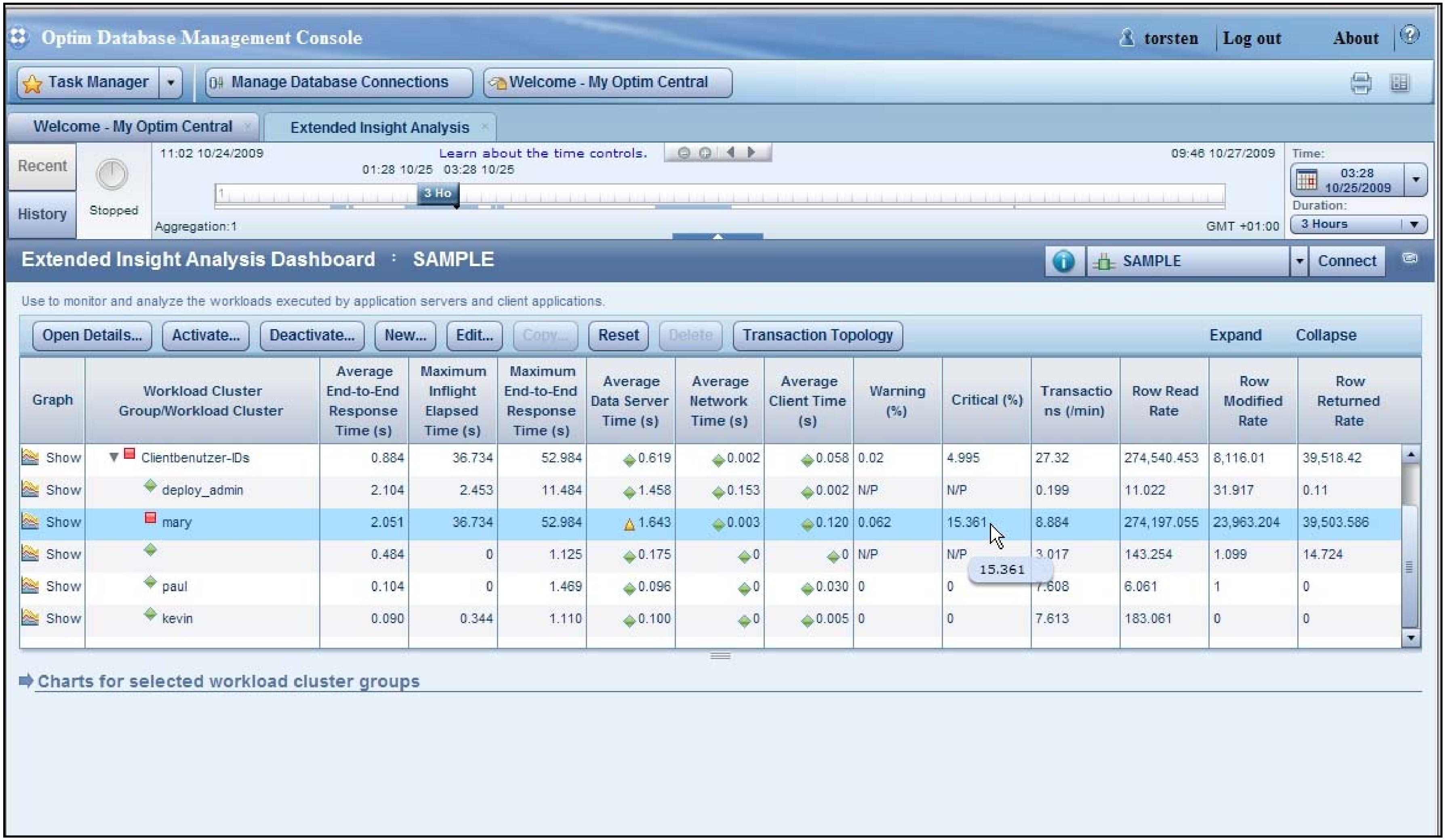
Capture SQL with pureQuery runtime

```
pdq.captureMode=ON
pdq.executionMode=DYNAMIC
pdq.pureQueryXml=pureQueryFolder/capture.pdqxml
pdq.cmx.controllerURL=9.30.77.61:60000
```

Java class	Java package	Method	Source line number	Build version	Source expression	Method Signature	Application Name	Metadata File
TestOPM	my.test	main	13	blahVer	N/P	N/P	blah	capture...

SELECT * FROM DB2ADMIN.INVENTORY

OPM Extended Insight (EI) Overview dashboard



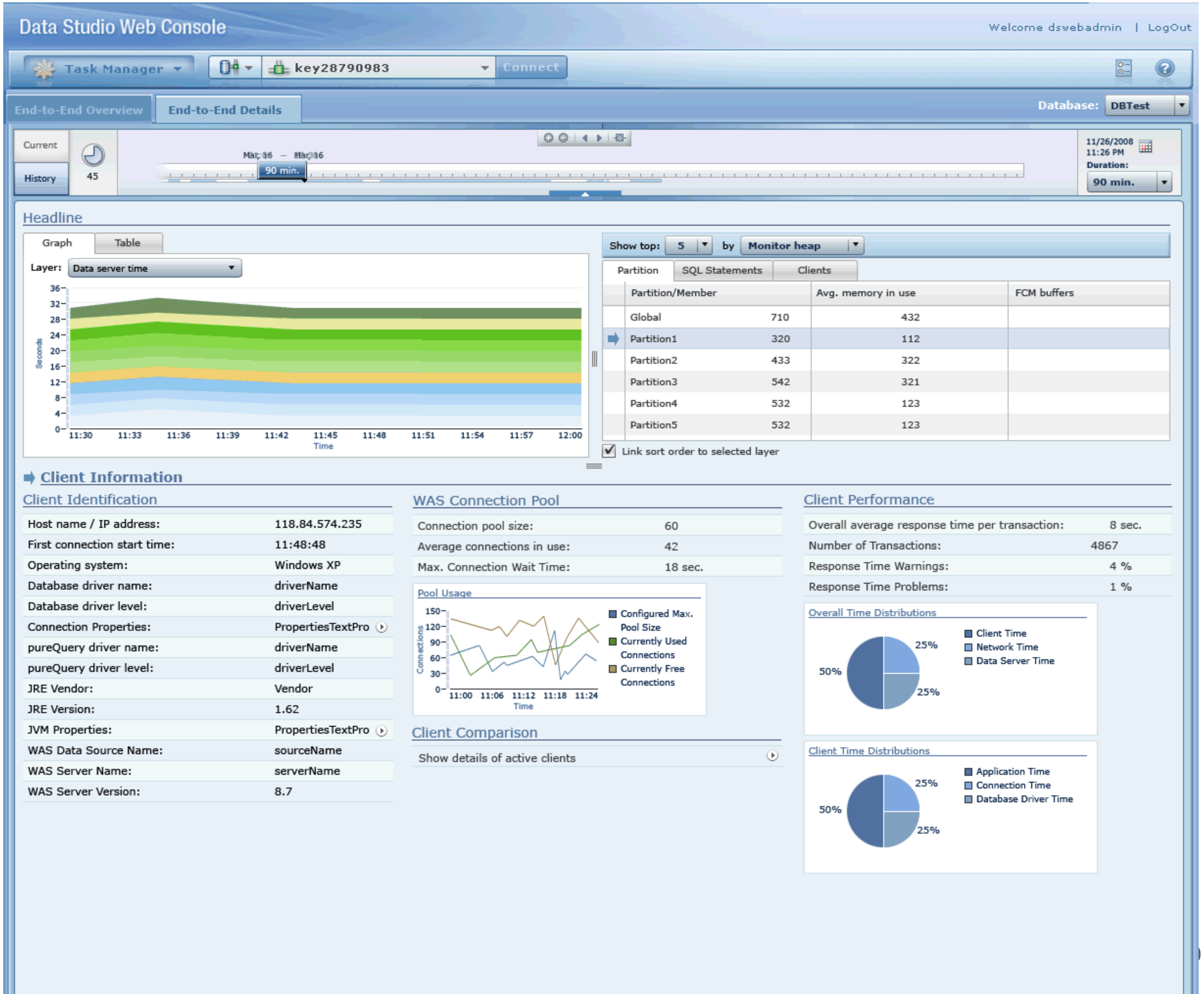
Extended Insight Analysis Dashboard : SAMPLE

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

Graph	Workload Cluster Group/Workload Cluster	Average End-to-End Response Time (s)	Maximum Inflight Elapsed Time (s)	Maximum End-to-End Response Time (s)	Average Data Server Time (s)	Average Network Time (s)	Average Client Time (s)	Warning (%)	Critical (%)	Transactions (/min)	Row Read Rate	Row Modified Rate	Row Returned Rate
Show	▼ 🟥 Clientbenutzer-IDs	0.884	36.734	52.984	◆0.619	◆0.002	◆0.058	0.02	4.995	27.32	274,540.453	8,116.01	39,518.42
Show	◆ deploy_admin	2.104	2.453	11.484	◆1.458	◆0.153	◆0.002	N/P	N/P	0.199	11.022	31.917	0.11
Show	🟥 mary	2.051	36.734	52.984	△1.643	◆0.003	◆0.120	0.062	15.361	8.884	274,197.055	23,963.204	39,503.586
Show	◆	0.484	0	1.125	◆0.175	◆0	◆0	N/P	N/P	3.017	143.254	1.099	14.724
Show	◆ paul	0.104	0	1.469	◆0.096	◆0	◆0.030	0	0	7.608	6.061	1	0
Show	◆ kevin	0.090	0.344	1.110	◆0.100	◆0	◆0.005	0	0	7.613	183.061	0	0

Charts for selected workload cluster groups

OPM Extended Insight Dashboard – Client Details



Partition/Member		Avg. memory in use	FCM buffers
Global	710	432	
Partition1	320	112	
Partition2	433	322	
Partition3	542	321	
Partition4	532	123	
Partition5	532	123	

☑ Link sort order to selected layer

Client Information

Client Identification

Host name / IP address:	118.84.574.235
First connection start time:	11:48:48
Operating system:	Windows XP
Database driver name:	driverName
Database driver level:	driverLevel
Connection Properties:	PropertiesTextPro
pureQuery driver name:	driverName
pureQuery driver level:	driverLevel
JRE Vendor:	Vendor
JRE Version:	1.62
JVM Properties:	PropertiesTextPro
WAS Data Source Name:	sourceName
WAS Server Name:	serverName
WAS Server Version:	8.7

WAS Connection Pool

Connection pool size:	60
Average connections in use:	42
Max. Connection Wait Time:	18 sec.

Pool Usage
Connections
150
120
90
60
30
0
11:00
11:06
11:12
11:18
11:24
Time
Configured Max. Pool Size
Currently Used Connections
Currently Free Connections

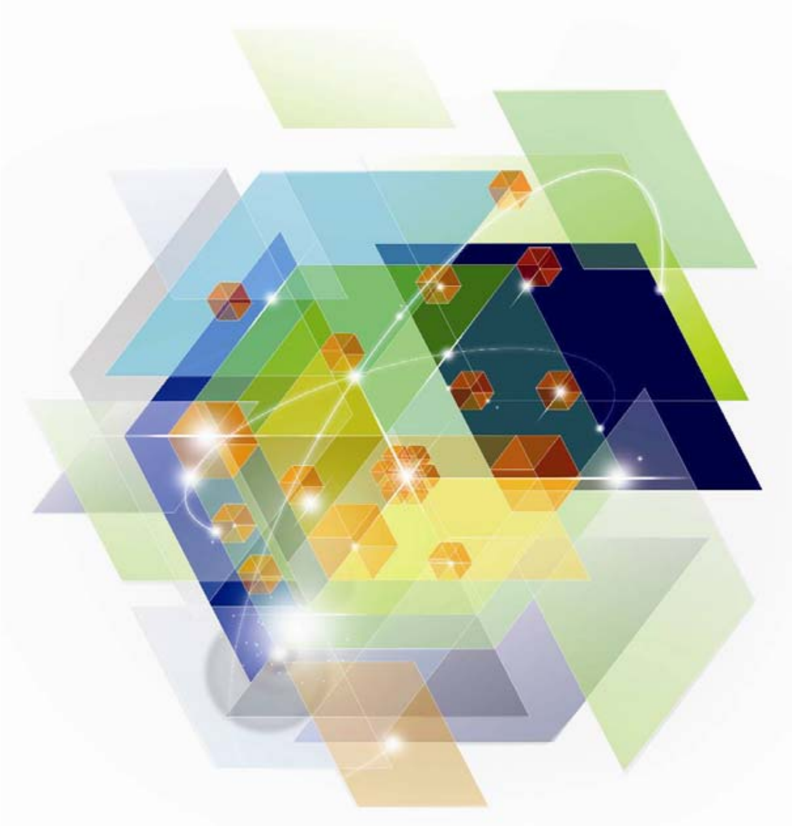
Client Comparison

Show details of active clients

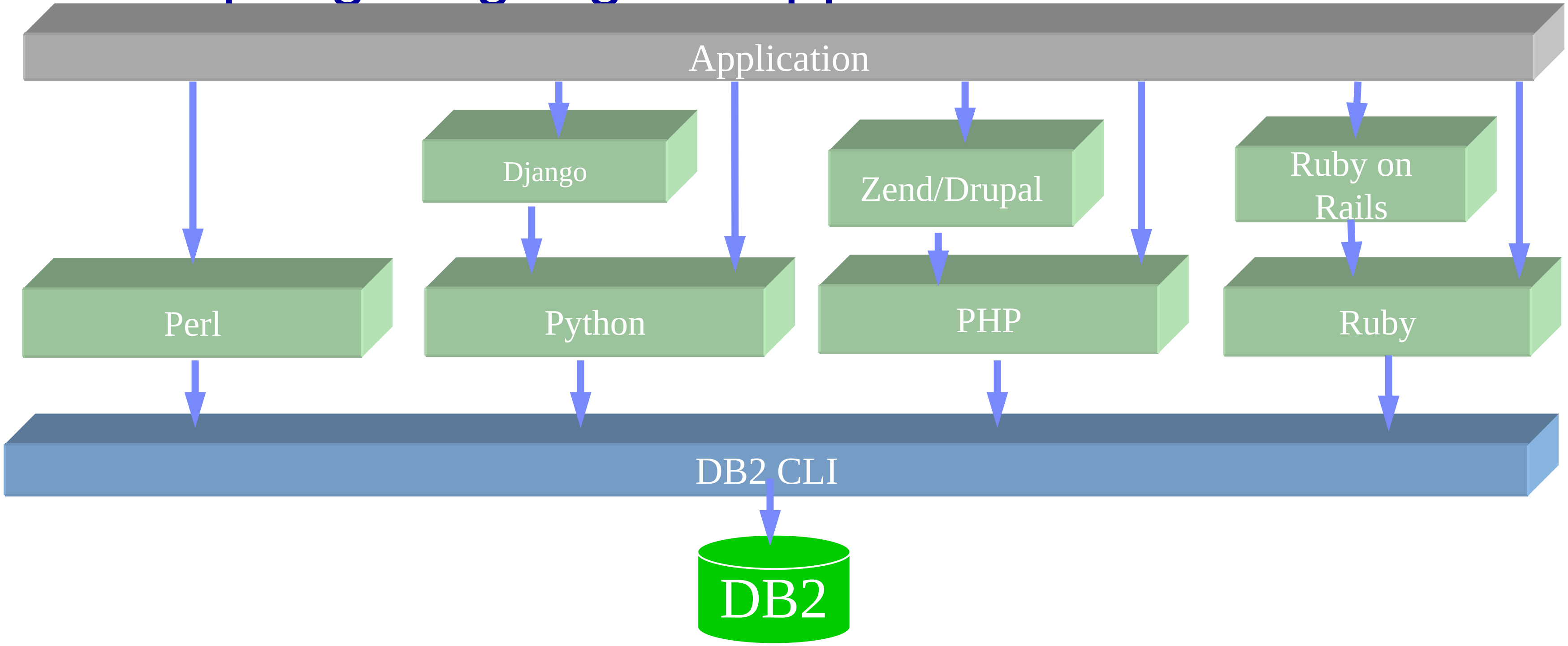
Client Performance

Overall average response time per transaction:	8 sec.
Number of Transactions:	4867
Response Time Warnings:	4 %
Response Time Problems:	1 %

Open Source Scripting Languages



DB2 Scripting languages support



Up-to-date with latest Django/Rails/Zend releases.

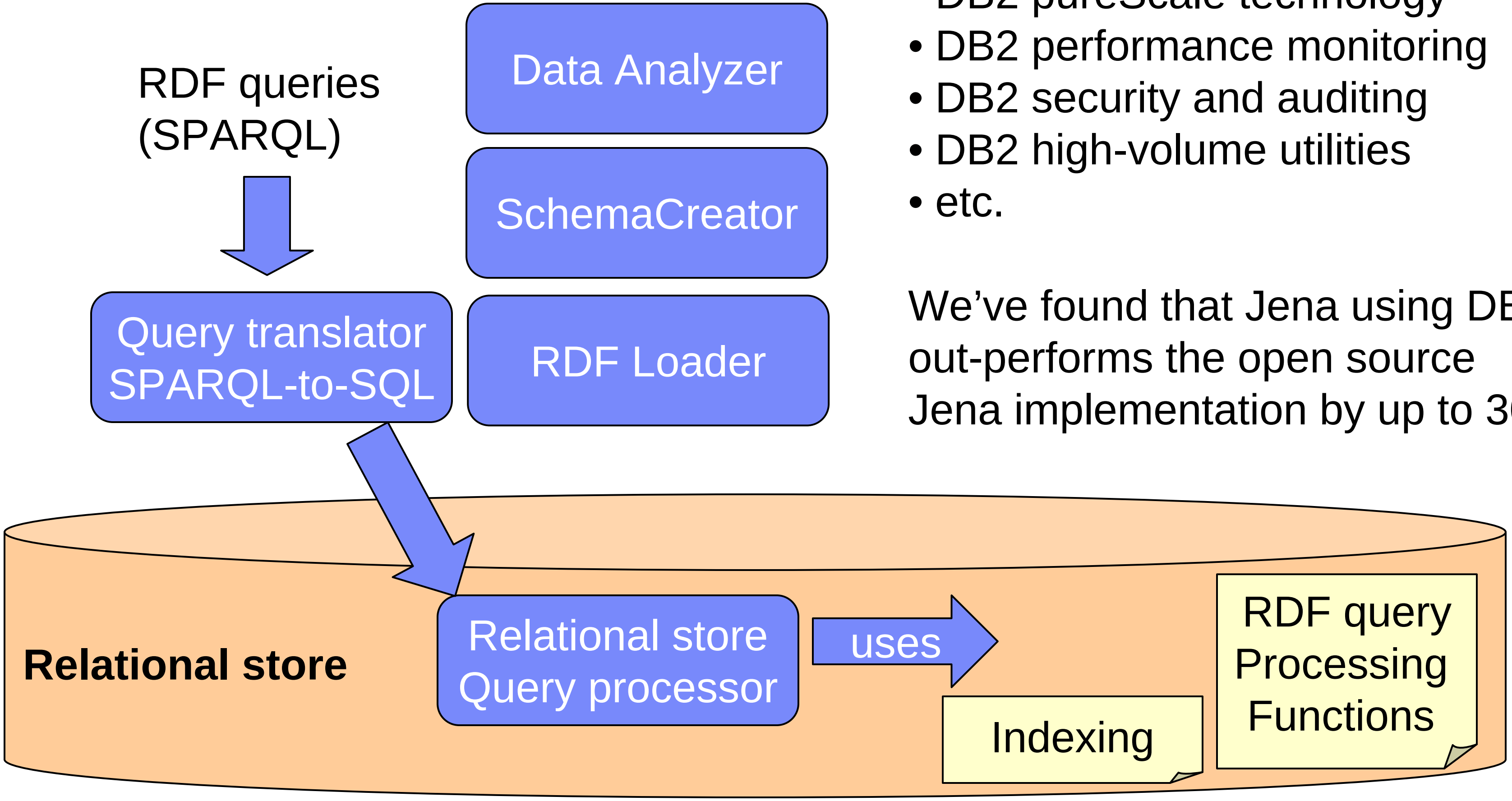
- All open source drivers and adapters are available on the DB2 media
- Python: http://code.google.com/p/ibm-db/
- Ruby: http://rubyforge.org/projects/rubyibm/
- PHP: http://pecl.php.net/package/ibm_db2/ , http://pecl.php.net/package/PDO_IBM
- Perl: http://search.cpan.org/~ibmtordb2/

In-the-works

- DB2 Drupal Support
  - A widely used PHP based Web Content Management System
  - DB2 support for Drupal 6 publicly available shortly
  - Drupal 7 support to follow

- SQL Generation for Java API based Query Systems before deployment
  - Complete Accelerator support for Hibernate / JPA Criteria Queries

RDF and Jena Built on Top of DB2 Infrastructure

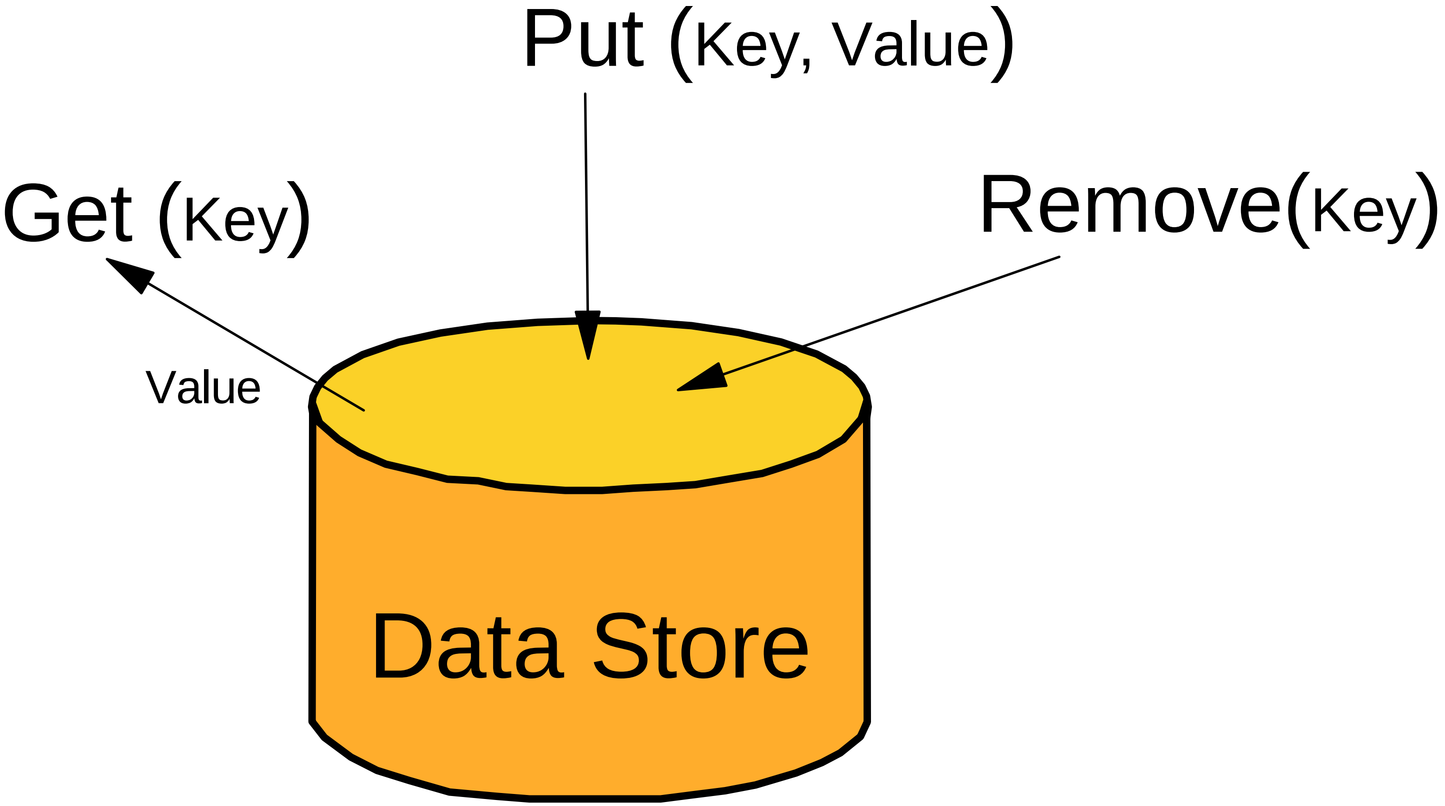

RDF queries (SPARQL)
Data Analyzer
SchemaCreator
Query translator SPARQL-to-SQL
RDF Loader
Relational store
Relational store Query processor
uses
Indexing
RDF query Processing Functions

Immediately takes advantage of:

- DB2 storage infrastructure
- DB2 backup/recovery
- DB2 pureScale technology
- DB2 performance monitoring
- DB2 security and auditing
- DB2 high-volume utilities
- etc.

We've found that Jena using DB2 out-performs the open source Jena implementation by up to 300%.

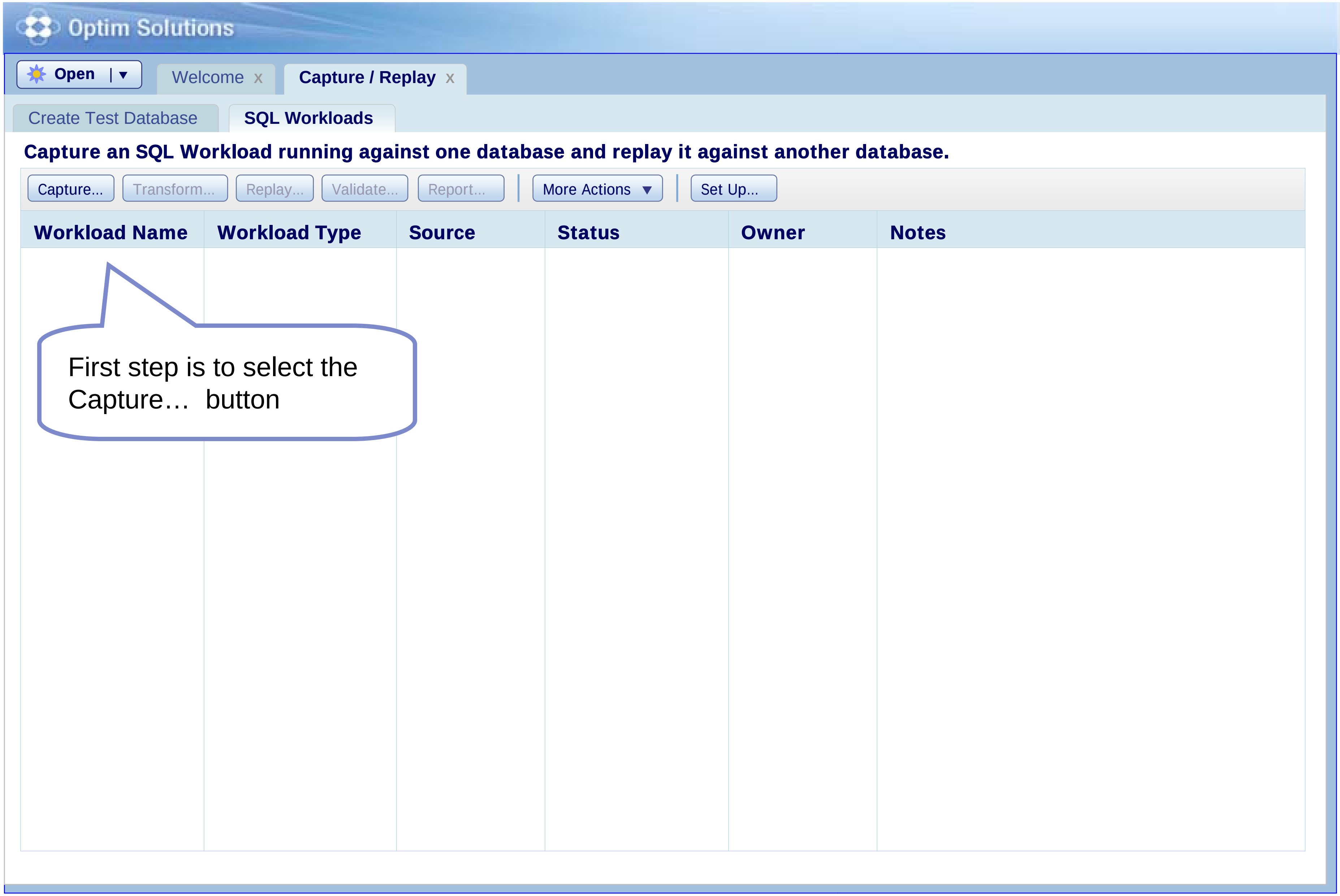
DB2 is making investments to support Key Value data (Redis)



Key/value access is very well optimized with the recently GA support for hash data access in DB2 11 for z/OS.   Range partitioning and DPSIs also help optimize for key/value access patterns.

Capture Replay Technology Preview

Capture Replay

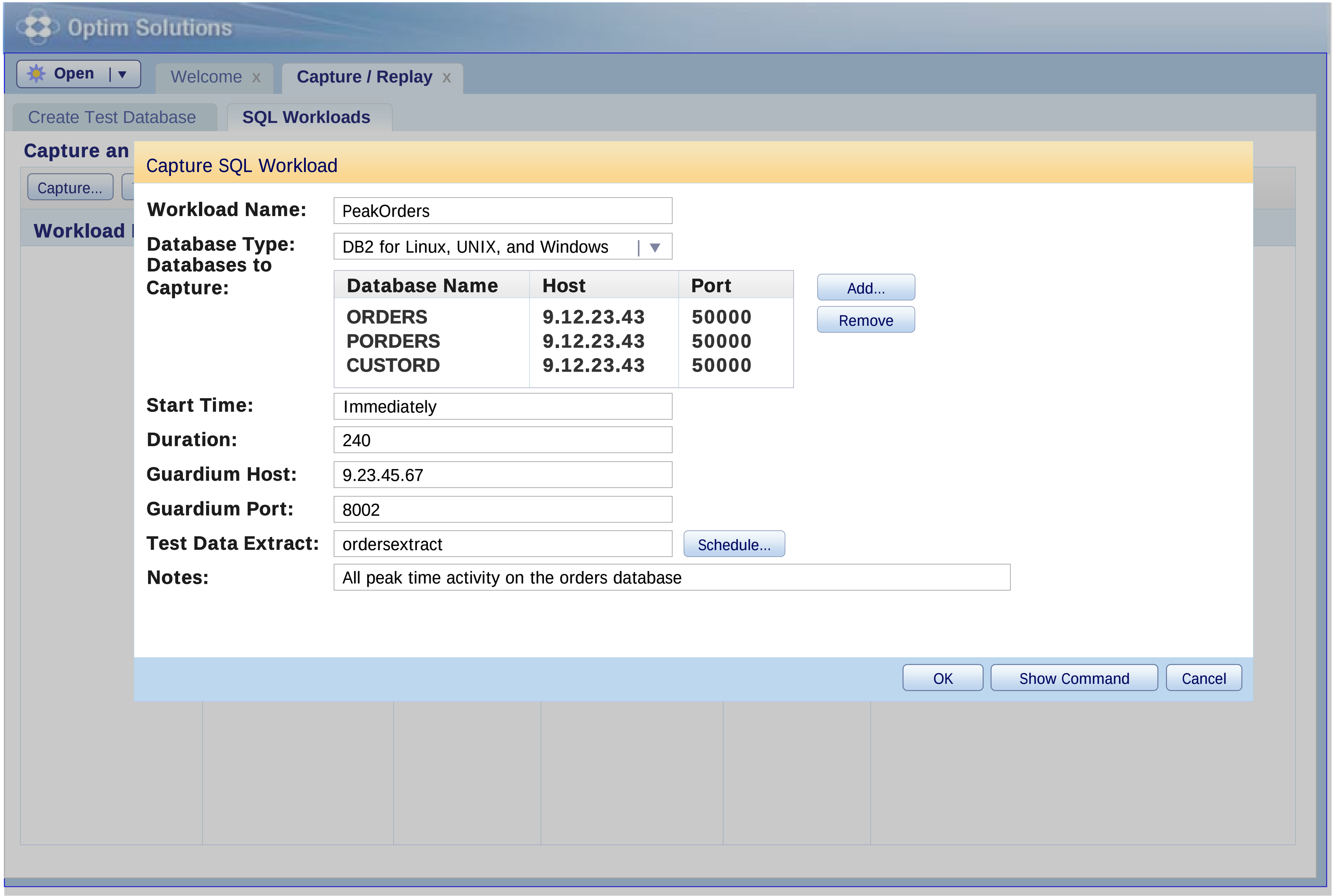


Capture an SQL Workload running against one database and replay it against another database.

Workload Name	Workload Type	Source	Status	Owner	Notes

First step is to select the Capture… button

Capture Replay



Capture SQL Workload

Workload Name: PeakOrders

Database Type: DB2 for Linux, UNIX, and Windows ▼

Databases to Capture:

Database Name	Host	Port
ORDERS	9.12.23.43	50000
PORDERS	9.12.23.43	50000
CUSTORD	9.12.23.43	50000

Start Time: Immediately

Duration: 240

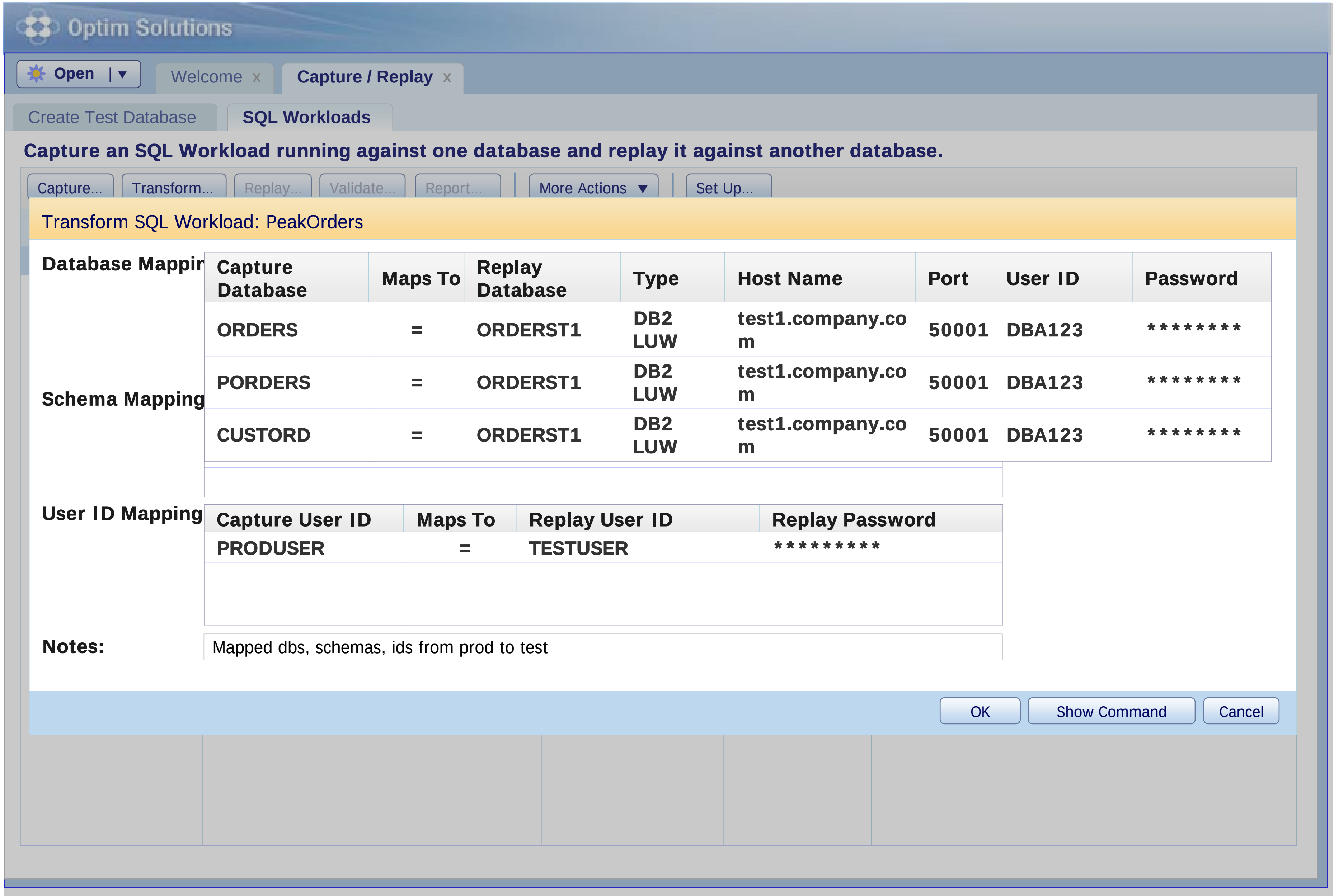
Guardium Host: 9.23.45.67

Guardium Port: 8002

Test Data Extract: ordersextract

Notes: All peak time activity on the orders database

Capture Replay



Capture an SQL Workload running against one database and replay it against another database.

Transform SQL Workload: PeakOrders

Database Mappin

Capture Database	Maps To	Replay Database	Type	Host Name	Port	User ID	Password
ORDERS	=	ORDERST1	DB2 LUW	test1.company.com	50001	DBA123	********
PORDERS	=	ORDERST1	DB2 LUW	test1.company.com	50001	DBA123	********
CUSTORD	=	ORDERST1	DB2 LUW	test1.company.com	50001	DBA123	********

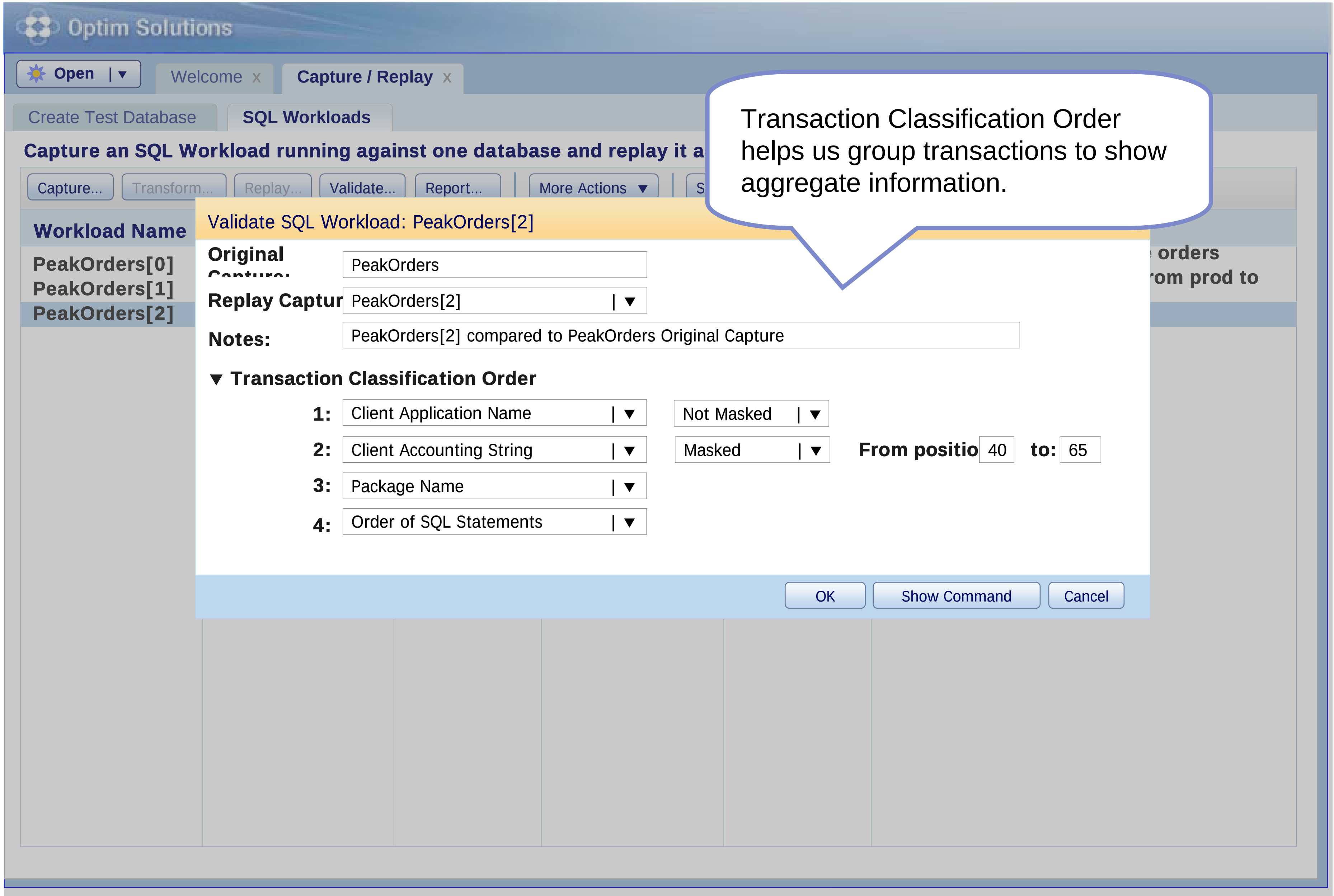
Schema Mapping

User ID Mapping

Capture User ID	Maps To	Replay User ID	Replay Password
PRODUSER	=	TESTUSER	*********

Notes: Mapped dbs, schemas, ids from prod to test

Capture Replay



Validate SQL Workload: PeakOrders[2]

Original Capture:	PeakOrders
Replay Captur	PeakOrders[2] ▼
Notes:	PeakOrders[2] compared to PeakOrders Original Capture

▼ Transaction Classification Order

1: Client Application Name ▼ Not Masked ▼

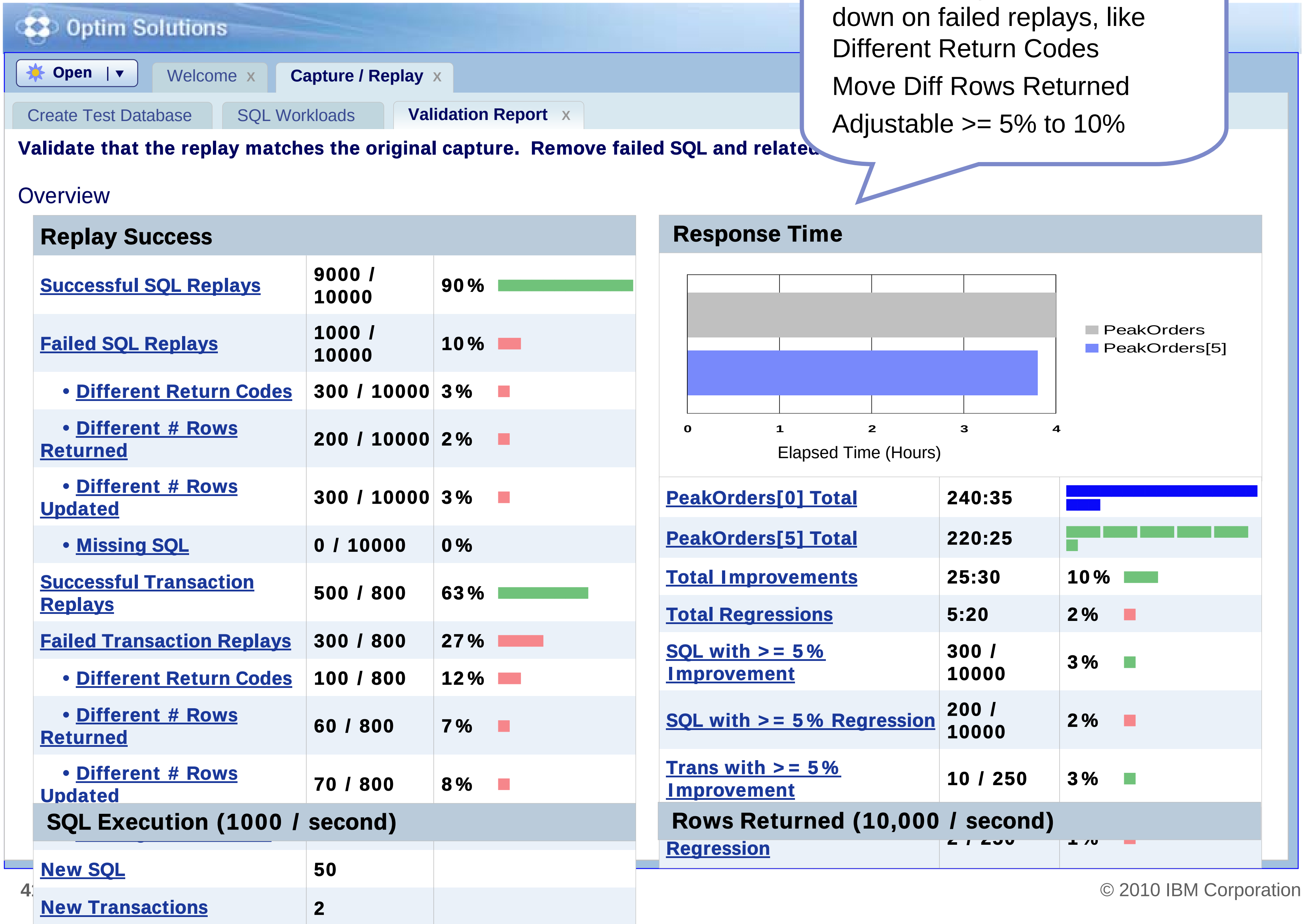
2: Client Accounting String ▼ Masked ▼ From positio 40 to: 65

3: Package Name ▼

4: Order of SQL Statements ▼

Transaction Classification Order helps us group transactions to show aggregate information.

Capture Replay



Optim Solutions

Validation report enables drill-down on failed replays, like Different Return Codes

Move Diff Rows Returned

Adjustable >= 5% to 10%

Validate that the replay matches the original capture. Remove failed SQL and related

Overview

Replay Success

Successful SQL Replays	9000 / 10000	90%	🟩
Failed SQL Replays	1000 / 10000	10%	🟥
• Different Return Codes	300 / 10000	3%	🟥
• Different # Rows Returned	200 / 10000	2%	🟥
• Different # Rows Updated	300 / 10000	3%	🟥
• Missing SQL	0 / 10000	0%	
Successful Transaction Replays	500 / 800	63%	🟩
Failed Transaction Replays	300 / 800	27%	🟥
• Different Return Codes	100 / 800	12%	🟥
• Different # Rows Returned	60 / 800	7%	🟥
• Different # Rows Updated	70 / 800	8%	🟥

SQL Execution (1000 / second)

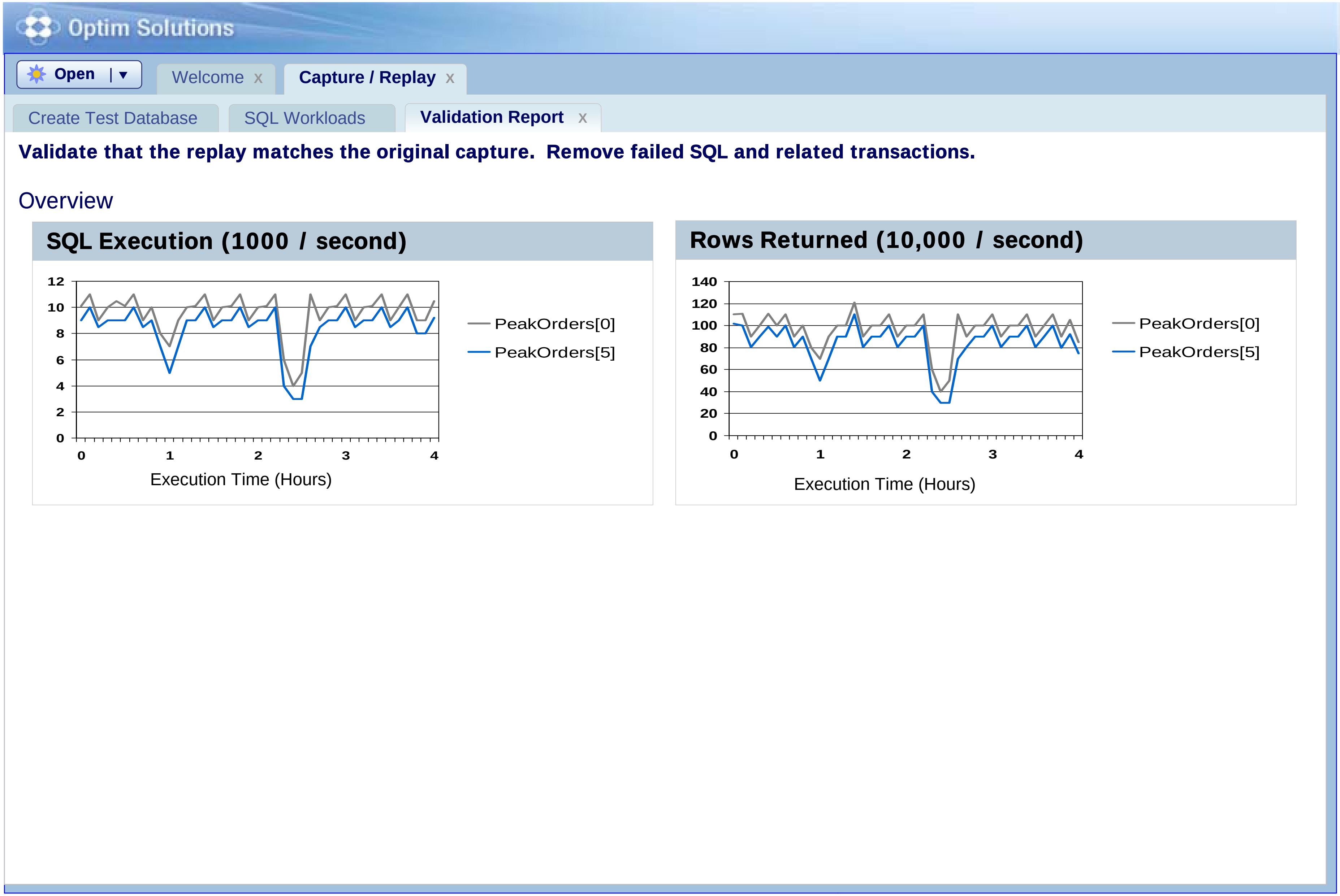
New SQL	50	
New Transactions	2	

Response Time

PeakOrders[0] Total	240:35	🟦
PeakOrders[5] Total	220:25	🟩
Total Improvements	25:30	10% 🟩
Total Regressions	5:20	2% 🟥
SQL with >= 5% Improvement	300 / 10000	3% 🟩
SQL with >= 5% Regression	200 / 10000	2% 🟥
Trans with >= 5% Improvement	10 / 250	3% 🟩
Regression	2 / 250	1% 🟥

Rows Returned (10,000 / second)

Capture Replay



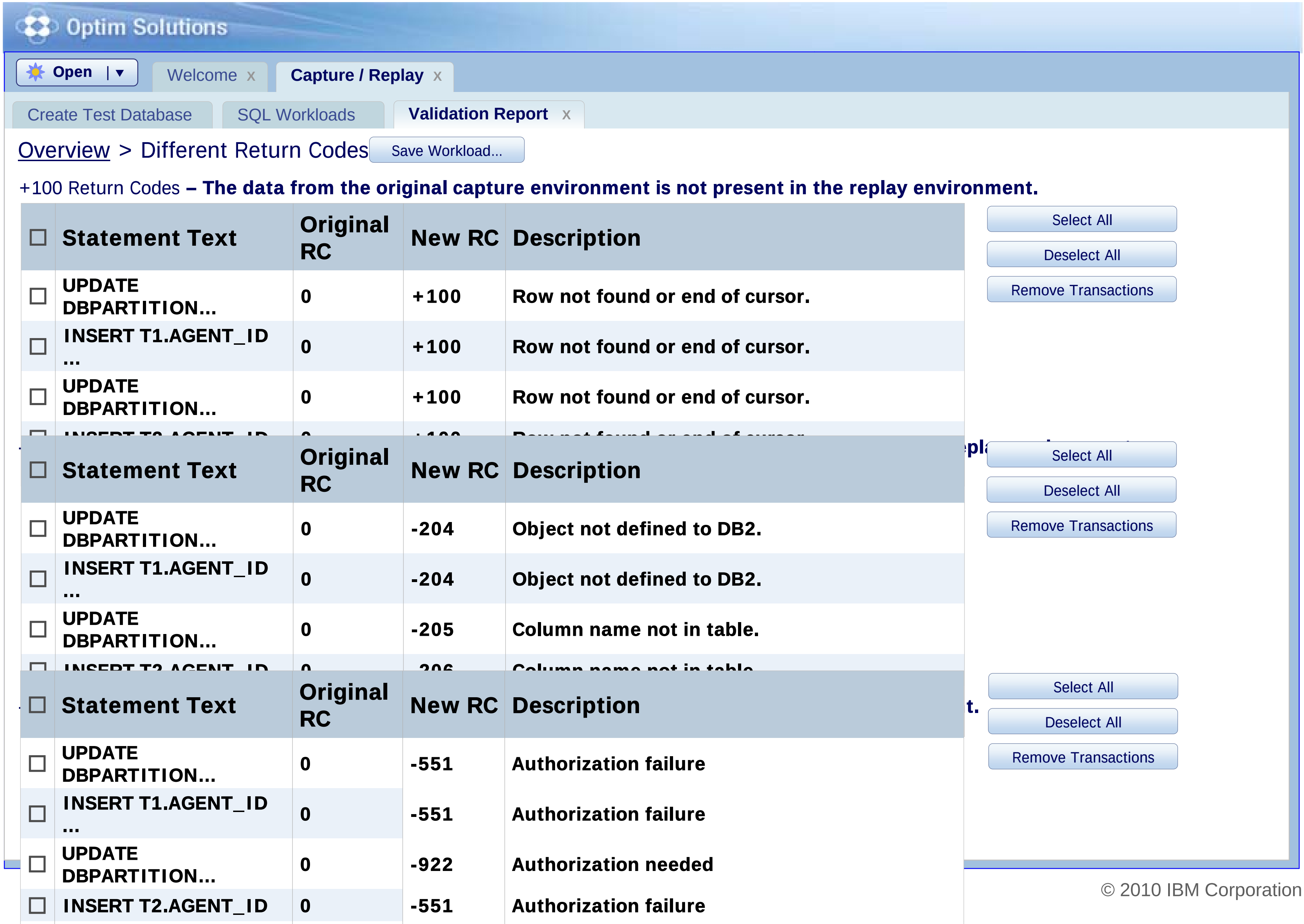
Validate that the replay matches the original capture. Remove failed SQL and related transactions.

Overview

SQL Execution (1000 / second)

Rows Returned (10,000 / second)

Capture Replay



Overview > Different Return Codes

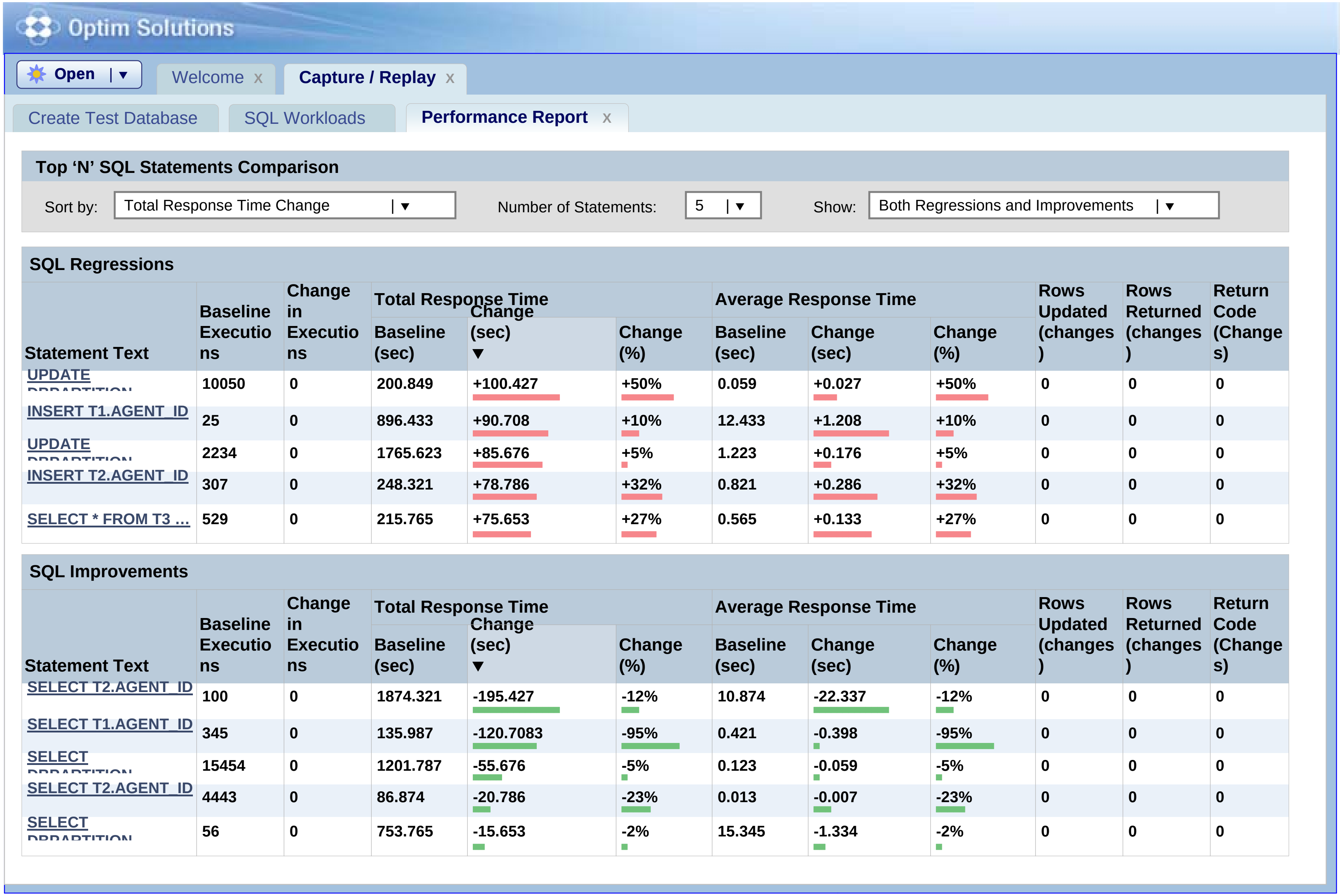
+100 Return Codes – **The data from the original capture environment is not present in the replay environment.**

□	Statement Text	Original RC	New RC	Description
□	UPDATE DBPARTITION...	0	+100	Row not found or end of cursor.
□	INSERT T1.AGENT_ID ...	0	+100	Row not found or end of cursor.
□	UPDATE DBPARTITION...	0	+100	Row not found or end of cursor.
□	INSERT T2.AGENT_ID	0	+100	Row not found or end of cursor.

□	Statement Text	Original RC	New RC	Description
□	UPDATE DBPARTITION...	0	-204	Object not defined to DB2.
□	INSERT T1.AGENT_ID ...	0	-204	Object not defined to DB2.
□	UPDATE DBPARTITION...	0	-205	Column name not in table.
□	INSERT T2.AGENT_ID	0	-206	Column name not in table.

□	Statement Text	Original RC	New RC	Description
□	UPDATE DBPARTITION...	0	-551	Authorization failure
□	INSERT T1.AGENT_ID ...	0	-551	Authorization failure
□	UPDATE DBPARTITION...	0	-922	Authorization needed
□	INSERT T2.AGENT_ID	0	-551	Authorization failure

Capture Replay



Top ‘N’ SQL Statements Comparison

Sort by: Total Response Time Change | ▼ Number of Statements: 5 | ▼ Show: Both Regressions and Improvements | ▼

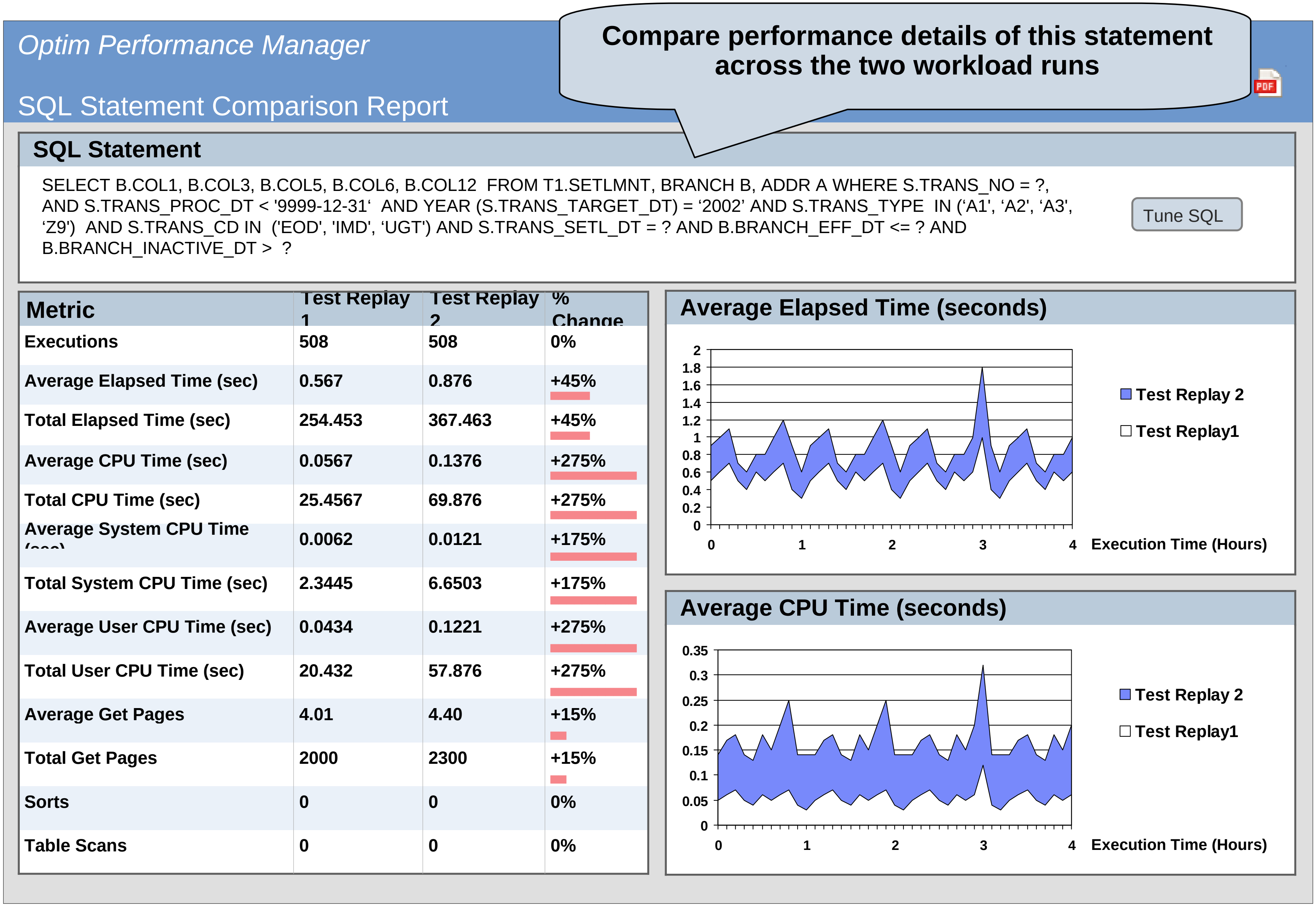
SQL Regressions

Statement Text	Baseline Executions	Change in Executions	Total Response Time Baseline (sec)	Total Response Time Change (sec) ▼	Total Response Time Change (%)	Average Response Time Baseline (sec)	Average Response Time Change (sec)	Average Response Time Change (%)	Rows Updated (changes)	Rows Returned (changes)	Return Code (Changes)
UPDATE	10050	0	200.849	+100.427	+50%	0.059	+0.027	+50%	0	0	0
INSERT T1.AGENT_ID	25	0	896.433	+90.708	+10%	12.433	+1.208	+10%	0	0	0
UPDATE	2234	0	1765.623	+85.676	+5%	1.223	+0.176	+5%	0	0	0
INSERT T2.AGENT_ID	307	0	248.321	+78.786	+32%	0.821	+0.286	+32%	0	0	0
SELECT * FROM T3 ...	529	0	215.765	+75.653	+27%	0.565	+0.133	+27%	0	0	0

SQL Improvements

Statement Text	Baseline Executions	Change in Executions	Total Response Time Baseline (sec)	Total Response Time Change (sec) ▼	Total Response Time Change (%)	Average Response Time Baseline (sec)	Average Response Time Change (sec)	Average Response Time Change (%)	Rows Updated (changes)	Rows Returned (changes)	Return Code (Changes)
SELECT T2.AGENT_ID	100	0	1874.321	-195.427	-12%	10.874	-22.337	-12%	0	0	0
SELECT T1.AGENT_ID	345	0	135.987	-120.7083	-95%	0.421	-0.398	-95%	0	0	0
SELECT	15454	0	1201.787	-55.676	-5%	0.123	-0.059	-5%	0	0	0
SELECT T2.AGENT_ID	4443	0	86.874	-20.786	-23%	0.013	-0.007	-23%	0	0	0
SELECT	56	0	753.765	-15.653	-2%	15.345	-1.334	-2%	0	0	0

Compare performance details of this statement across the two workload runs



Optim Performance Manager

SQL Statement Comparison Report

SQL Statement

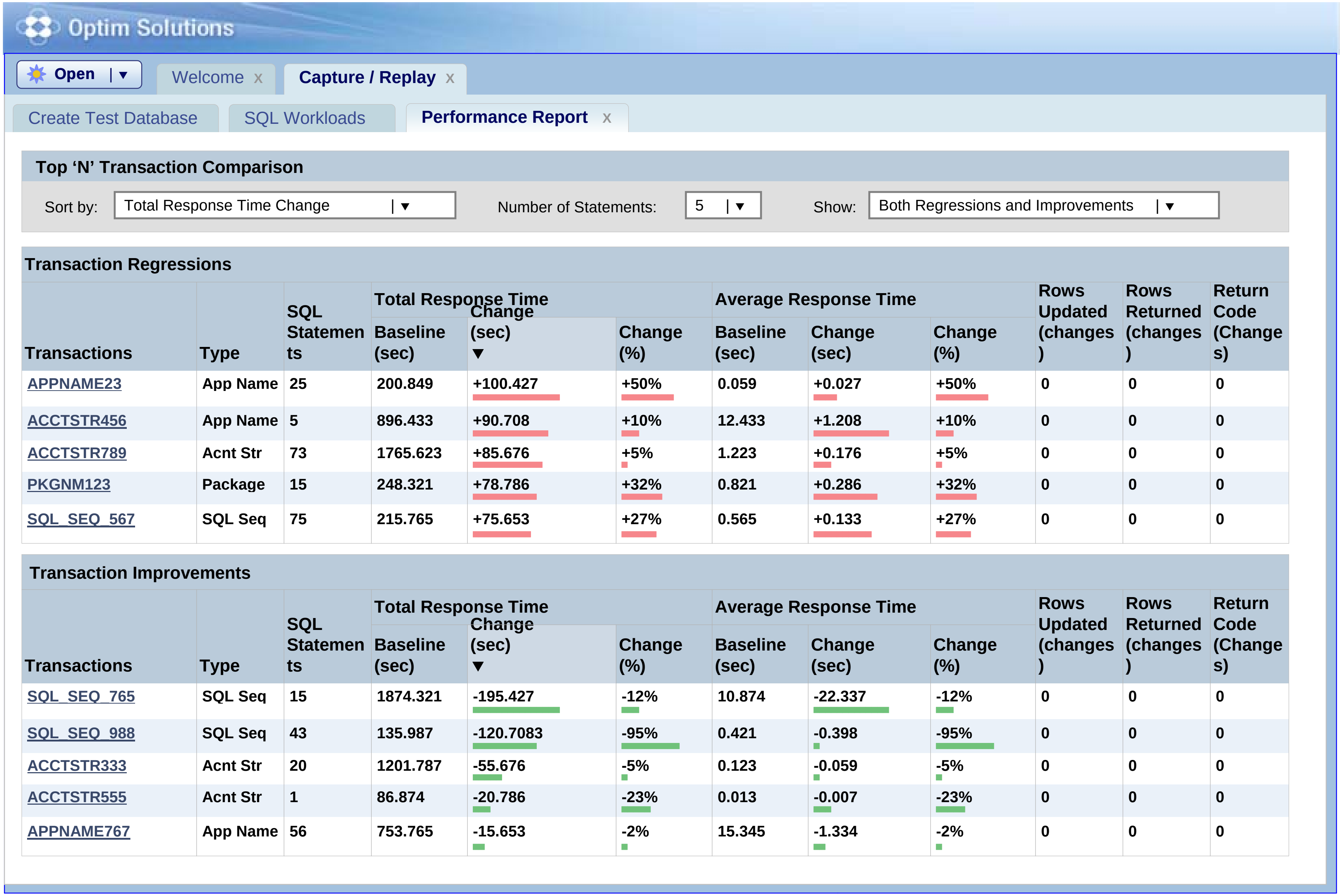
SELECT B.COL1, B.COL3, B.COL5, B.COL6, B.COL12 FROM T1.SETLMNT, BRANCH B, ADDR A WHERE S.TRANS_NO = ?, AND S.TRANS_PROC_DT < '9999-12-31‘ AND YEAR (S.TRANS_TARGET_DT) = ‘2002’ AND S.TRANS_TYPE IN (‘A1', ‘A2', ‘A3', ‘Z9') AND S.TRANS_CD IN ('EOD', 'IMD', ‘UGT') AND S.TRANS_SETL_DT = ? AND B.BRANCH_EFF_DT <= ? AND B.BRANCH_INACTIVE_DT > ?

Metric	Test Replay 1	Test Replay 2	% Change
Executions	508	508	0%
Average Elapsed Time (sec)	0.567	0.876	+45%
Total Elapsed Time (sec)	254.453	367.463	+45%
Average CPU Time (sec)	0.0567	0.1376	+275%
Total CPU Time (sec)	25.4567	69.876	+275%
Average System CPU Time (sec)	0.0062	0.0121	+175%
Total System CPU Time (sec)	2.3445	6.6503	+175%
Average User CPU Time (sec)	0.0434	0.1221	+275%
Total User CPU Time (sec)	20.432	57.876	+275%
Average Get Pages	4.01	4.40	+15%
Total Get Pages	2000	2300	+15%
Sorts	0	0	0%
Table Scans	0	0	0%

Average Elapsed Time (seconds)

Average CPU Time (seconds)

Capture Replay



Top ‘N’ Transaction Comparison

Sort by: Total Response Time Change ▼ Number of Statements: 5 ▼ Show: Both Regressions and Improvements ▼

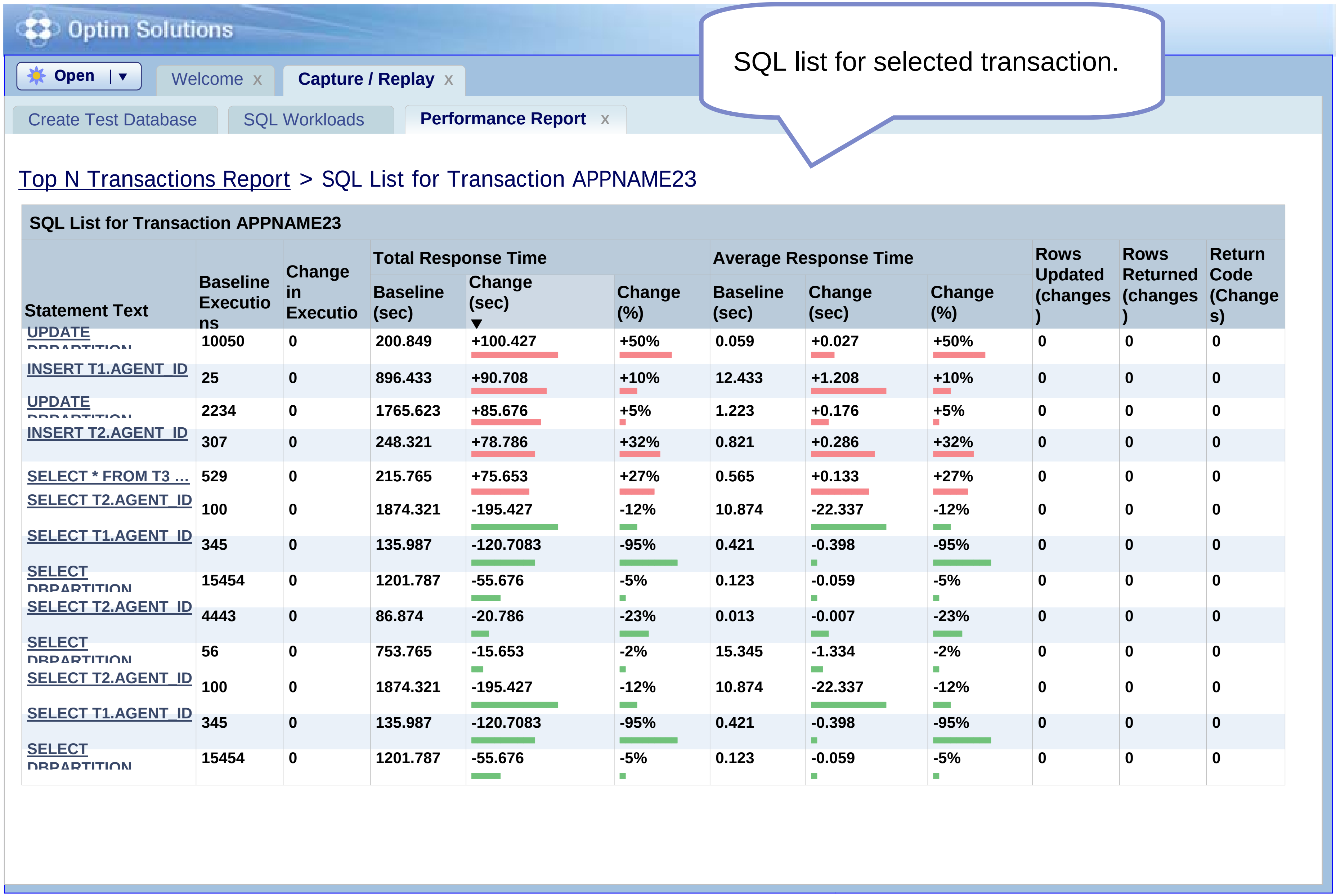
Transaction Regressions

			Total Response Time			Average Response Time					
Transactions	Type	SQL Statements	Baseline (sec)	Change (sec) ▼	Change (%)	Baseline (sec)	Change (sec)	Change (%)	Rows Updated (changes)	Rows Returned (changes)	Return Code (Changes)
APPNAME23	App Name	25	200.849	+100.427	+50%	0.059	+0.027	+50%	0	0	0
ACCTSTR456	App Name	5	896.433	+90.708	+10%	12.433	+1.208	+10%	0	0	0
ACCTSTR789	Acnt Str	73	1765.623	+85.676	+5%	1.223	+0.176	+5%	0	0	0
PKGNM123	Package	15	248.321	+78.786	+32%	0.821	+0.286	+32%	0	0	0
SQL_SEQ_567	SQL Seq	75	215.765	+75.653	+27%	0.565	+0.133	+27%	0	0	0

Transaction Improvements

			Total Response Time			Average Response Time					
Transactions	Type	SQL Statements	Baseline (sec)	Change (sec) ▼	Change (%)	Baseline (sec)	Change (sec)	Change (%)	Rows Updated (changes)	Rows Returned (changes)	Return Code (Changes)
SQL_SEQ_765	SQL Seq	15	1874.321	-195.427	-12%	10.874	-22.337	-12%	0	0	0
SQL_SEQ_988	SQL Seq	43	135.987	-120.7083	-95%	0.421	-0.398	-95%	0	0	0
ACCTSTR333	Acnt Str	20	1201.787	-55.676	-5%	0.123	-0.059	-5%	0	0	0
ACCTSTR555	Acnt Str	1	86.874	-20.786	-23%	0.013	-0.007	-23%	0	0	0
APPNAME767	App Name	56	753.765	-15.653	-2%	15.345	-1.334	-2%	0	0	0

Capture Replay



SQL list for selected transaction.

Top N Transactions Report > SQL List for Transaction APPNAME23

SQL List for Transaction APPNAME23

Statement Text	Baseline Executions	Change in Executions	Total Response Time			Average Response Time			Rows Updated (changes)	Rows Returned (changes)	Return Code (Changes)
			Baseline (sec)	Change (sec) ▼	Change (%)	Baseline (sec)	Change (sec)	Change (%)			
UPDATE DBPARTITION	10050	0	200.849	+100.427	+50%	0.059	+0.027	+50%	0	0	0
INSERT T1.AGENT_ID	25	0	896.433	+90.708	+10%	12.433	+1.208	+10%	0	0	0
UPDATE DBPARTITION	2234	0	1765.623	+85.676	+5%	1.223	+0.176	+5%	0	0	0
INSERT T2.AGENT_ID	307	0	248.321	+78.786	+32%	0.821	+0.286	+32%	0	0	0
SELECT * FROM T3 ...	529	0	215.765	+75.653	+27%	0.565	+0.133	+27%	0	0	0
SELECT T2.AGENT_ID	100	0	1874.321	-195.427	-12%	10.874	-22.337	-12%	0	0	0
SELECT T1.AGENT_ID	345	0	135.987	-120.7083	-95%	0.421	-0.398	-95%	0	0	0
SELECT DBPARTITION	15454	0	1201.787	-55.676	-5%	0.123	-0.059	-5%	0	0	0
SELECT T2.AGENT_ID	4443	0	86.874	-20.786	-23%	0.013	-0.007	-23%	0	0	0
SELECT DBPARTITION	56	0	753.765	-15.653	-2%	15.345	-1.334	-2%	0	0	0
SELECT T2.AGENT_ID	100	0	1874.321	-195.427	-12%	10.874	-22.337	-12%	0	0	0
SELECT T1.AGENT_ID	345	0	135.987	-120.7083	-95%	0.421	-0.398	-95%	0	0	0
SELECT DBPARTITION	15454	0	1201.787	-55.676	-5%	0.123	-0.059	-5%	0	0	0

- IBM Data Studio
  - www.ibm.com/software/data/studio
    - FAQs / Tutorials
    - Downloads
    - Forum / Blogs
    - Join the IBM Data Studio user community

- **Data Studio Book**
  - http://bit.ly/dstudiobook

Thank
You