

ORACLE®



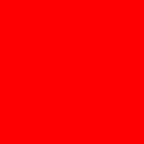
**ORACLE
DEVELOP**

ORACLE®

Advanced and Efficient Java Persistence with JDBC, UCP, Java in the Database

Kuassi Mensah

Database Access Services, Database APIs, and Net Services



The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.

The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Who is Kuassi Mensah ...

- Director of Product Manager for Database Access Frameworks (Net Services, DRCP, etc) and Database APIs (Java, C/C++, PHP, Ruby, Python, Perl).
- Hold a MS in Computer Sciences from the Programming Institute of the University of Paris VI.
- Has published several articles and a book @ <http://www.amazon.com/exec/obidos/ASIN/1555583296>
- Is a frequent speaker at Oracle and IT events and maintain a blog @ <http://db360.blogspot.com>, as well as Facebook, LinkedIn, and Twitter (<http://twitter.com/kmensah>) pages.

Optimizing Network Data Traffic

What is Session Data Unit (SDU) and Why Tune it?

- Controls SQL*Net packet size
 - Default is 2k in pre 11.2 and 8192 from 11.2
 - Max is 64K with 11.2
- For bulk data transfer (LOB, XML), increase to 64k in
 - URL `jdbc:oracle:thin:@(DESCRIPTION=...(SDU=8192) ...)`
 - `sqlnet.ora`: set `DEFAULT_SDU_SIZE`
 - `tnsnames.ora`: set SDU in ADDRESS
- Larger SDU gives
 - Better Network throughput
 - Fewer system calls to send and receive data
 - Less CPU usage – system and user
 - Beware of larger memory footprint

JDBC Memory Management

Best Practices

How JDBC Memory Allocation Works

- defineBytes and defineChars arrays
 - VARCHAR(4000) ->8k bytes, 4k for RAW and LOB columns, and 32 bytes for everything else
 - A query returning 200 VARCHAR(4000) columns with 100 fetch size will allocate $(2 * 4000) * 200 * 100 = 160MB$*
- **Problem** Zeroing defineBytes and defineChars array buffers is required by Java lang. spec. but is expensive
- **Best Practices:**
 - Define tables carefully
 - Code queries carefully
 - Set the fetchSize carefully
 - Trade Space for Speed or vice versa

JDBC Memory Management

Trading Space for Speed

- Keep ResultSet Buffers with Statement Caching
 - Eliminates the overhead of repeated cursor creation
 - Prevents repeated statement parsing and creation
 - Oracle JDBC Drivers furnish two Statement Caching
 - Implicit statement caching
 - Explicit statement caching

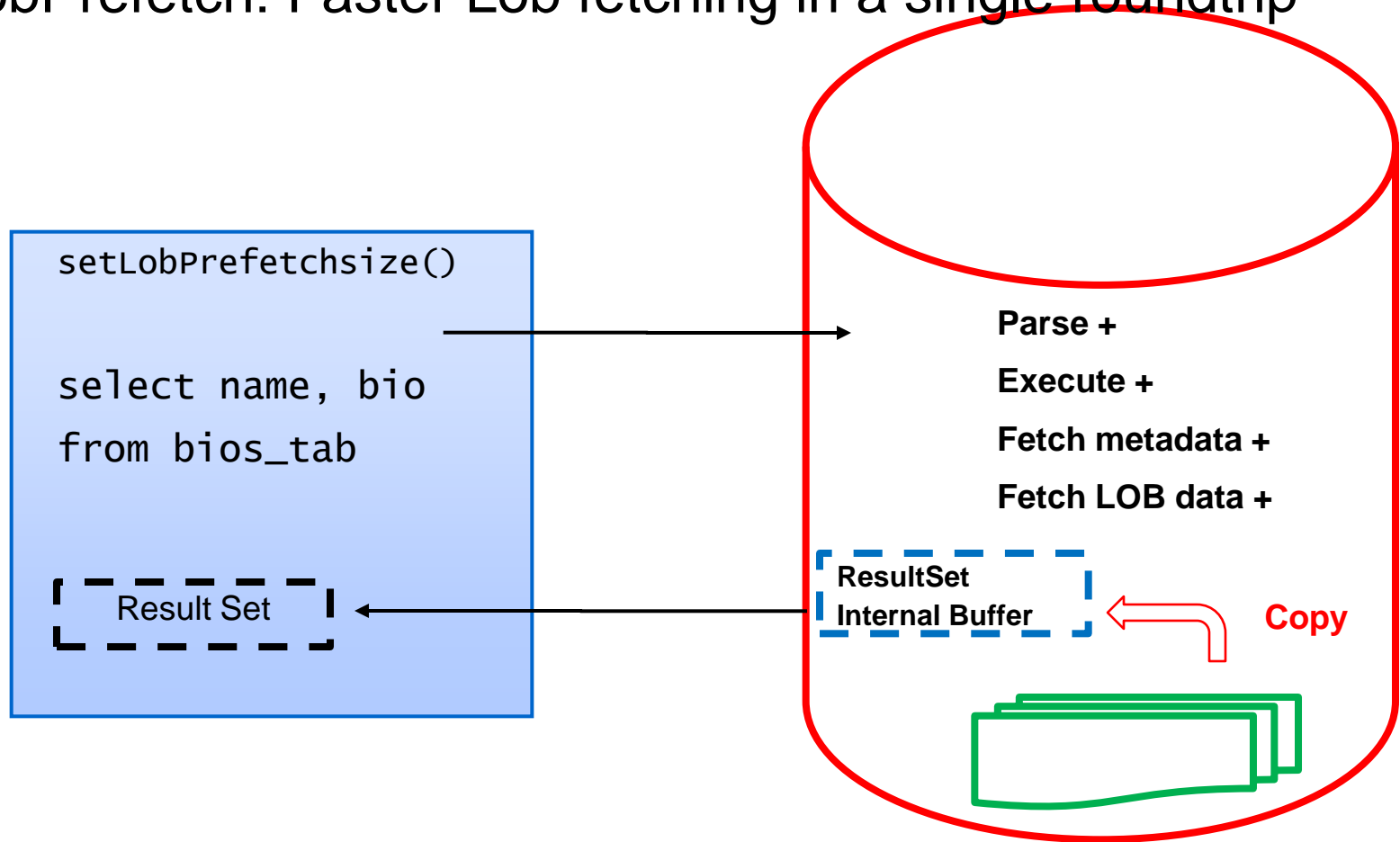
JDBC Memory Allocation

Trading Speed for Space

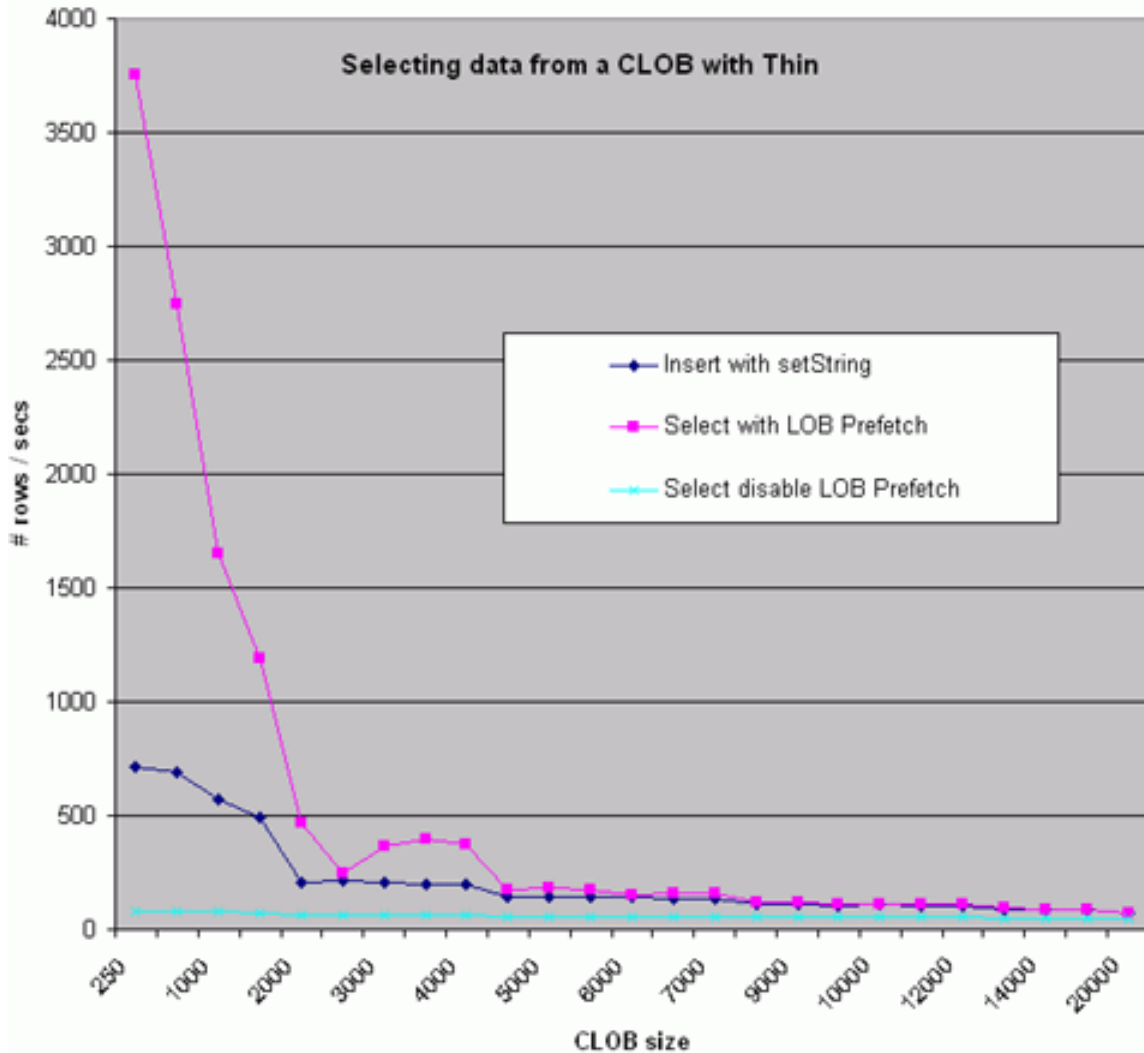
- Release buffers when statements return to the cache
 - In 10.2 JDBC Set
`oracle.jdbc.freeMemoryOnEnterImplicitCache`
However, if the application has many open statements at once, this can be a problem.
 - In 11.1.07 and up, set `maxCachedBufferSize` to mitigate this problem.
 - JDBC 11.2 driver has more sophisticated Memory Allocation
- More details in the JDBC Memory Management White paper
<http://www.oracle.com/technetwork/database/enterprise-edition/memory.pdf>

Optimize LOB operations

LobPrefetch: Faster Lob fetching in a single roundtrip



LOB PreFetching Performance



Throughput (per sec)

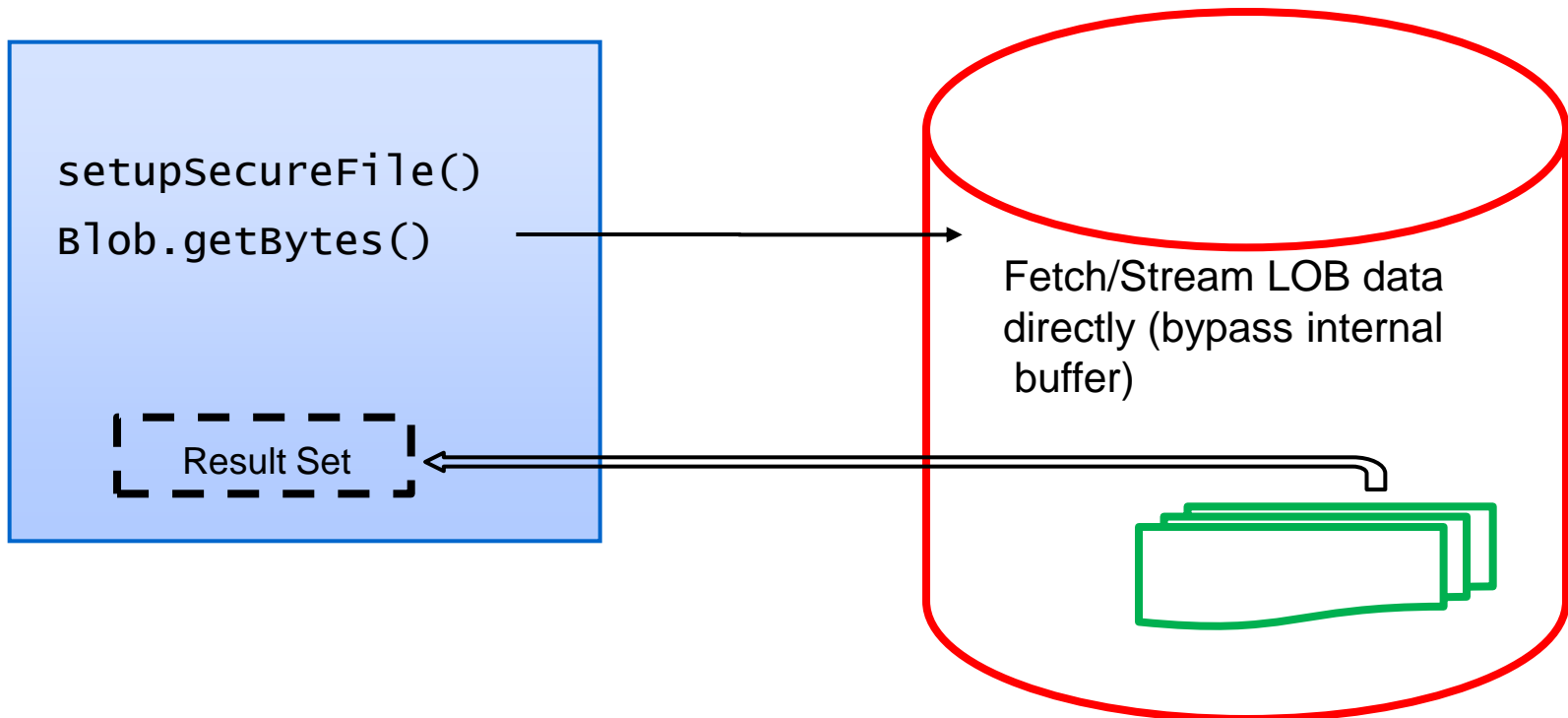
LOB Size	Insert with setString	Select with LOB Prefetch	Select disable LOB Prefetch
250	711.019	3,753.53	79.002
500	688.664	2,742.17	82.62
1000	574.091	1,652.56	75.60
1500	494.781	1,193.19	74.461
2000	202.502	464.699	65.473
2500	211.488	248.171	65.746
3000	205.418	362.545	65.40
3500	196.82	396.878	65.01
4000	198.538	374.439	63.614
4500	141.702	171.863	59.447
5000	144.79	181.954	59.51
5500	144.967	177.582	59.196
6000	140.27	148.714	58.196
6500	134.843	154.846	57.56
7000	134.523	157.19	58.28
8500	113.841	120.184	55.33
9000	111.022	119.236	54.685
9500	104.606	110.377	53.646
10000	108.343	108.133	53.516
11000	103.353	110.278	52.666
12000	104.051	107.921	52.099
13000	86.639	93.416	50.214
14000	87.264	89.678	50.906
15000	86.422	86.232	49.902
20000	70.573	74.393	45.495

Optimize Large LOBs ops & Bulk Data Xfer

SecureFiles LOBs:

Large Reads/Writes

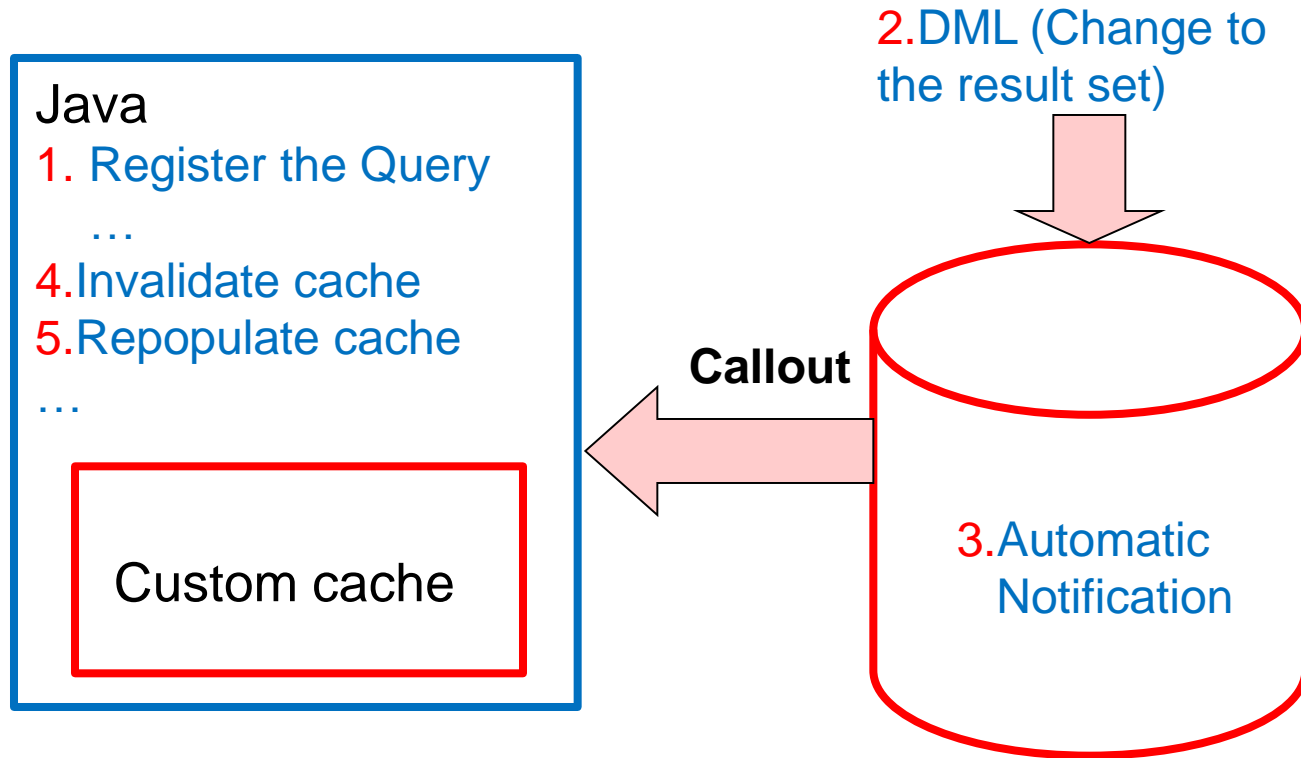
- BASIC LOBs: internal buffer copy are expensive
- SECUREFILE LOBS: *“Zero-copy IO” or “Vectored i/o mechanism”*



Query Change Notification

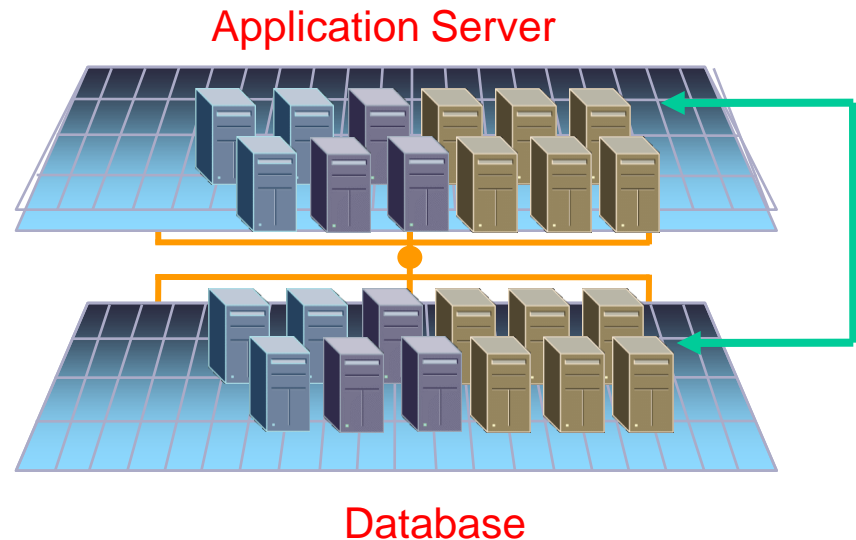
Automatic Reports Caching/Refreshing

- a) Execute Query
- b) Return ResultSet without executing the query as long as ResultSet is “valid”
- c) When changes invalidate ResultSet, RDBMS sends Notification



Consistent Client-side ResultSet Caching

- Configure Database (init.ora)
`client_result_cache_size=200M`
`client_result_cache_lag=5000`
- Configure Client (sqlnet.ora)
`OCI_QUERY_CACHE_SIZE=200M`
`OCI_QUERY_CACHE_MAXROWS=20`



- **11.2**

Transparent ResultSet Caching -- No code change

```
alter table emp result_cache;
```

- **11.1**

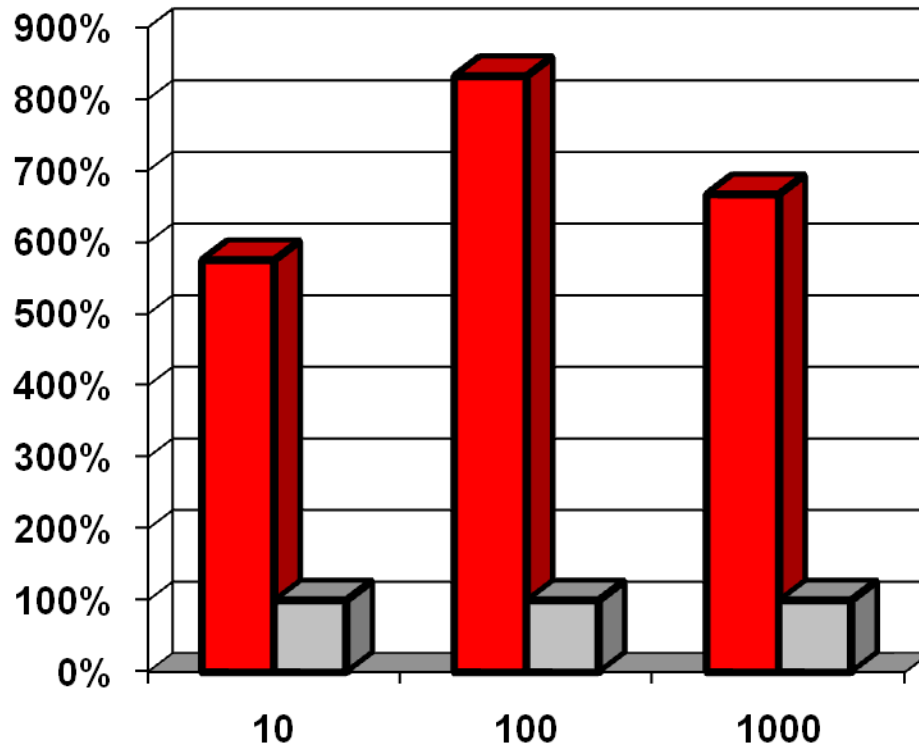
Set hints for Caching the Result Set

```
select /*+ result_cache */ * from employees
```

- Available with JDBC-OCI, C, C++, PHP, Ruby, ODP.Net, ODBC

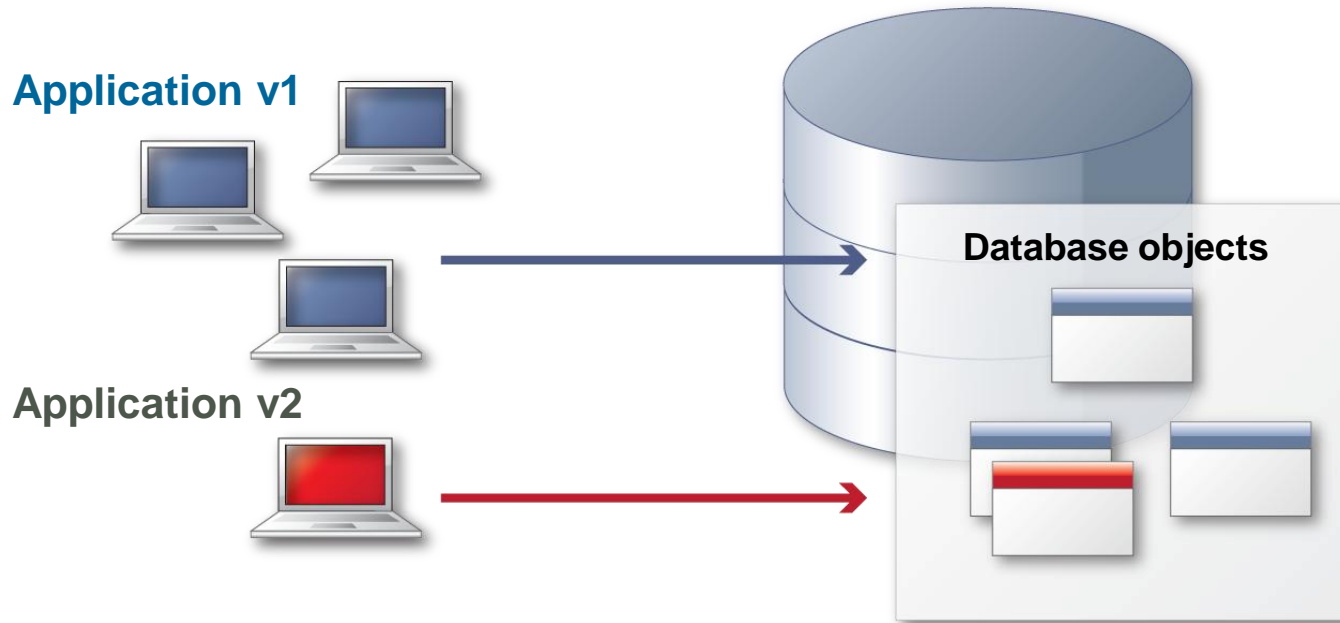
Consistent Client-side ResultSet Caching Performance

5-8 x Faster (Query Serviced in Client-Cache)



Online Application Upgrade & Patching

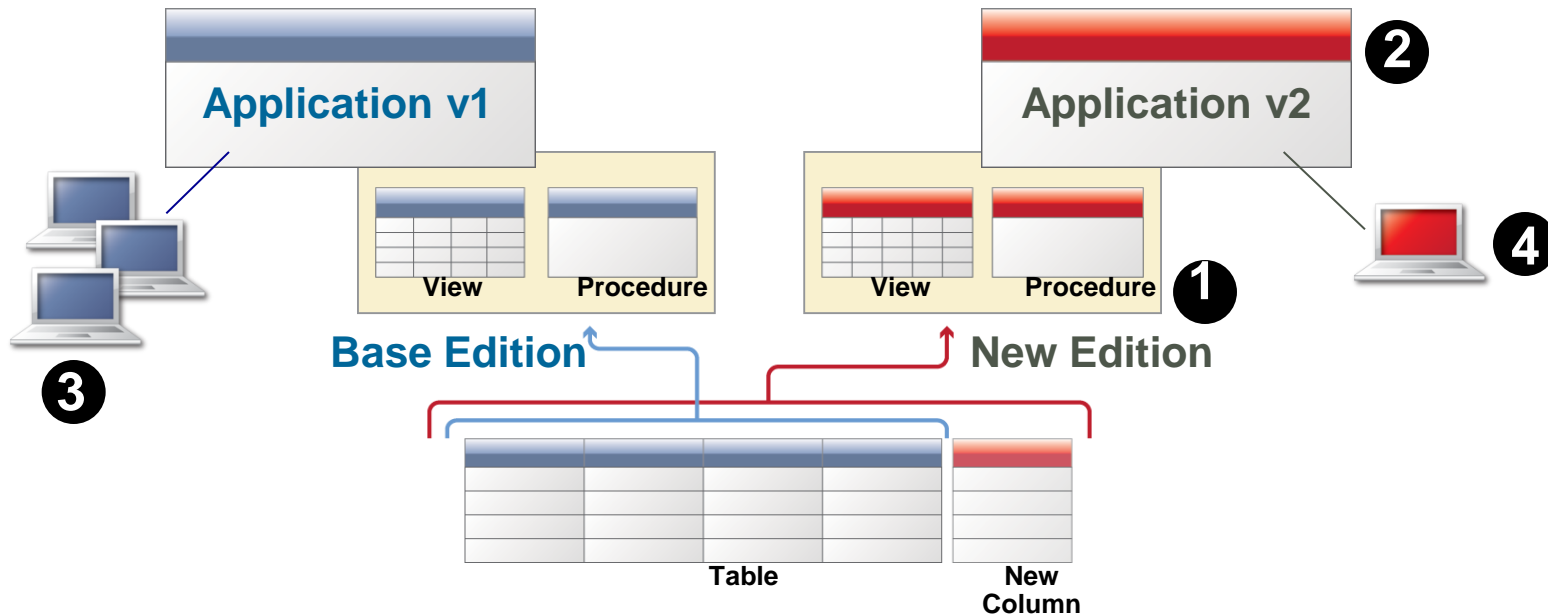
Requirements & Challenge



- Allow making changes database objects while application is running
- Make changes transparent to application

Online Application Upgrade & Patching

Solution: Edition-Based Redefinition



- 1 Change objects in new edition w/o affecting online users
- 2 Deploy new application referencing new edition
- 3 Current users are not effected, they still reference base edition
- 4 Phase in new application version over time and drop the old one

Upgrading & Patching 24x7 Java Applications

App V1

- Default “EDITION”
- Initial Database schema (tables, PL/SQL packages)
- v1.java

App V2

- Create new “EDITION”
- Update Database schema (tables, PL/SQL packages)
- v2.java
 - set session in edition name with conn. property CONNECTION_PROPERTY_EDITION_NAME
 - **Update application logic to use new schema.**

Advanced Security with JDBC

- For company wide security policies, Oracle JDBC furnishes the following advanced security features:
 - Encryption: AES (128/192/256-bit) ,3DES (112/168-bit), RC4
 - Data Integrity: MD5 (16 bytes), SHA1 (20 bytes)
 - Authentication: Radius, Kerberos, SSL, OS, Password

Example (Kerberos)

```
create user "BOB@US.ORACLE.COM" identified
externally;

prop.setProperty(
OracleConnection.CONNECTION_PROPERTY_THIN_NET_AUTHENTICA
TION_SERVICES, "( KERBEROS )");
```

- Otherwise, we recommend SSL as single technology for Encryption + Data Integrity + Strong Authentication

Universal Connection Pool

- A Single/Universal Java Connection Pool
 - Supports any type of Java connection (JDBC, XA, JCA, LDAP)
 - Supports stand-alone deployment (BPEL, Toplink, Tomcat)
 - Supports any database (Oracle, non-Oracle)
 - Seamless integration with Oracle RAC and Data Guard
- HA and Scalability Services
 - Fast Connection Failover (FCF)
 - Runtime Connection Load Balancing (RCLB)
 - Web Session based Affinity to RAC instance
 - Transaction based Affinity to RAC instance

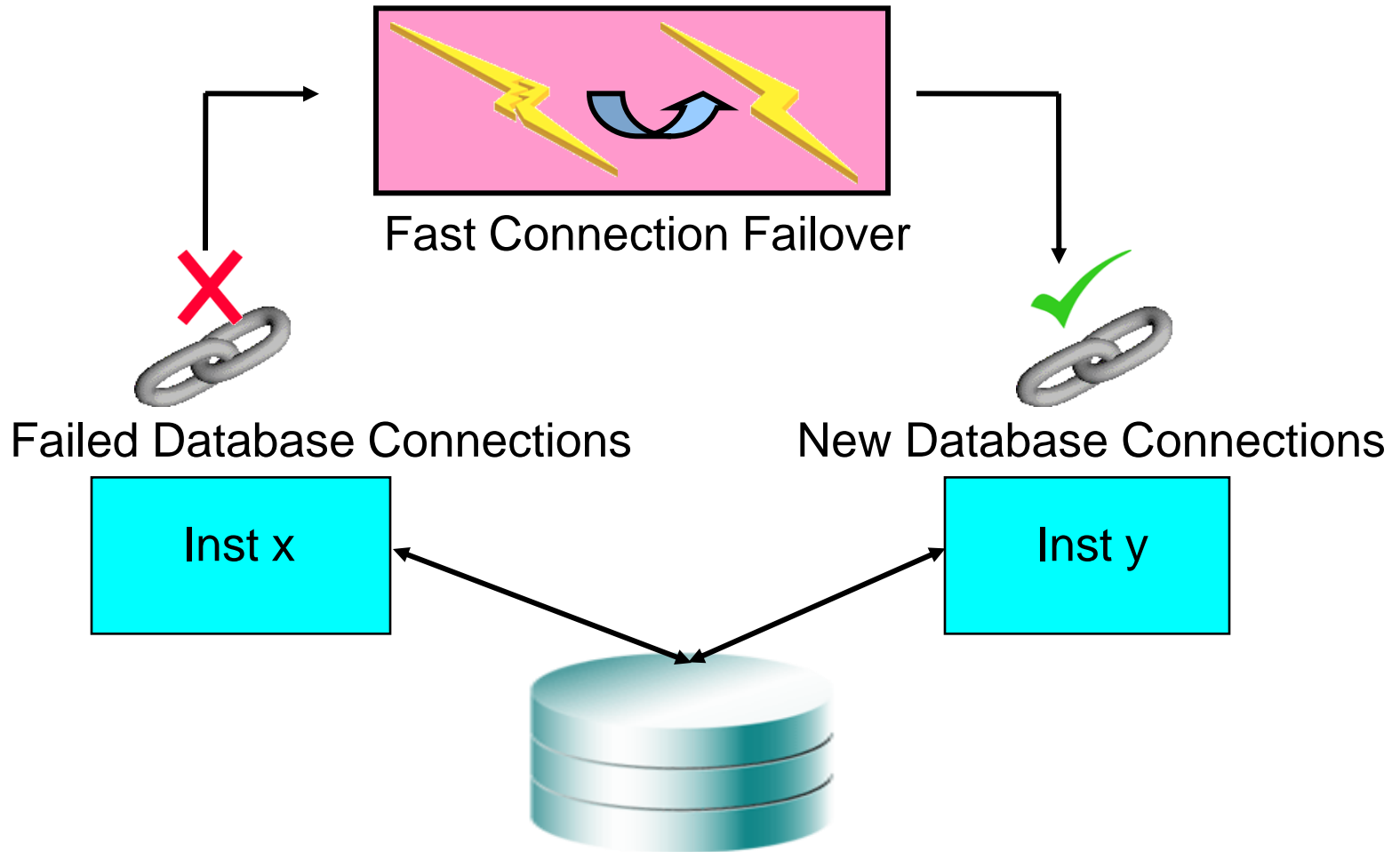
UCP

Fast Connection Failover (FCF)

- Rapid database failure detection
- Aborts and removes invalid connections from the pool
- Supports unplanned and planned outages
- Recognizes new nodes that join an Oracle RAC cluster
- Distributes runtime work requests to all active Oracle RAC instances

Fast Connection Failover (FCF)

Application View



Implement FCF

```
PoolDataSource pds = PoolDataSourceFactory.getPoolDataSource();
pds.setONSConfiguration("nodes=racnode1:4200,racnode2:4200");
pds.setFastConnectionFailoverEnabled(true);
.....

boolean retry = false;
do {
    try {
        Connection conn = pds.getConnection("scott", "tiger");

        // Necessary recovery if retrying
        // Data operations using conn object

        retry = false;
    } catch (SQLException sqlexc) {
        if (conn == null || !((ValidConnection)conn).isValid())
            retry = true;
    }
} while (retry);
.....
```

FCF Staticstics – Example

```
PoolDataSource pds = PoolDataSourceFactory.getPoolDataSource();
.....
OracleJDBCConnectionPoolStatistics ostats =
    (OracleJDBCConnectionPoolStatistics) pds.getStatistics();
String fcfInfo = ostats.getFCFProcessingInfo();
```

```
-----
Jan 29, 2009 11:14:52 SUCCESS <Reason:unplanned> <Type:SERVICE_DOWN> \
  <Service:"sn_1"> <Instance:"dsin_1"> <Db:"db_1"> \
  Connections:(Available=6 Affected=2 MarkedDown=2 Closed=2) \
  (Borrowed=6 Affected=2 MarkedDown=2 Closed=2)
Jan 29, 2009 11:14:53 SUCCESS <Type:HOST_DOWN> <Host:"h_1"> \
  Connections:(Available=6 Affected=4 MarkedDown=4 Closed=4) \
  (Borrowed=6 Affected=4 MarkedDown=4 Closed=4)
Jan 29, 2009 11:14:54 SUCCESS <Reason:planned> <Type:SERVICE_UP> \
  <Service:"sn_1"> <Instance:"dsin_1"> <Db:"db_1"> \
  Connections:(Available=6 Affected=2 MarkedDown=2 Closed=2) \
  (Borrowed=6 Affected=2 MarkedDown=2 Closed=2) TornDown=2 \
  MarkedToClose=2 Cardinality=2
```

FCF – WebLogic config

The screenshot displays the Oracle WebLogic Console interface. The browser address bar shows the URL: `http://localhost:7001/console/console.portal?CreateGlobalJDBCGridLinkDataS`. The console title is "Welcome to WebLogic Server" and the current page is "Create a New JDBC GridLi...".

The left sidebar shows the "Change Center" and "Domain Structure" for the `wl_server` domain. The "Domain Structure" tree includes: Environment, Deployments, Services, Security Realms, Interoperability, and Diagnostics.

The main content area is titled "Create a New JDBC GridLink Data Source". It features a navigation bar with "Back", "Next", "Finish", and "Cancel" buttons. The "ONS Client Configuration" section is active, with the instruction "Define ONS Client Configuration". A checkbox for "FAN Enabled" is checked. Below this, there is a text input field for "ONS host and port" and an "Add" button. A list box contains the following entries:

- racnode1:6200
- racnode2:6200
- racnode3:6200

A "Remove" button is located to the right of the list box. The bottom status bar of the console shows "Done".

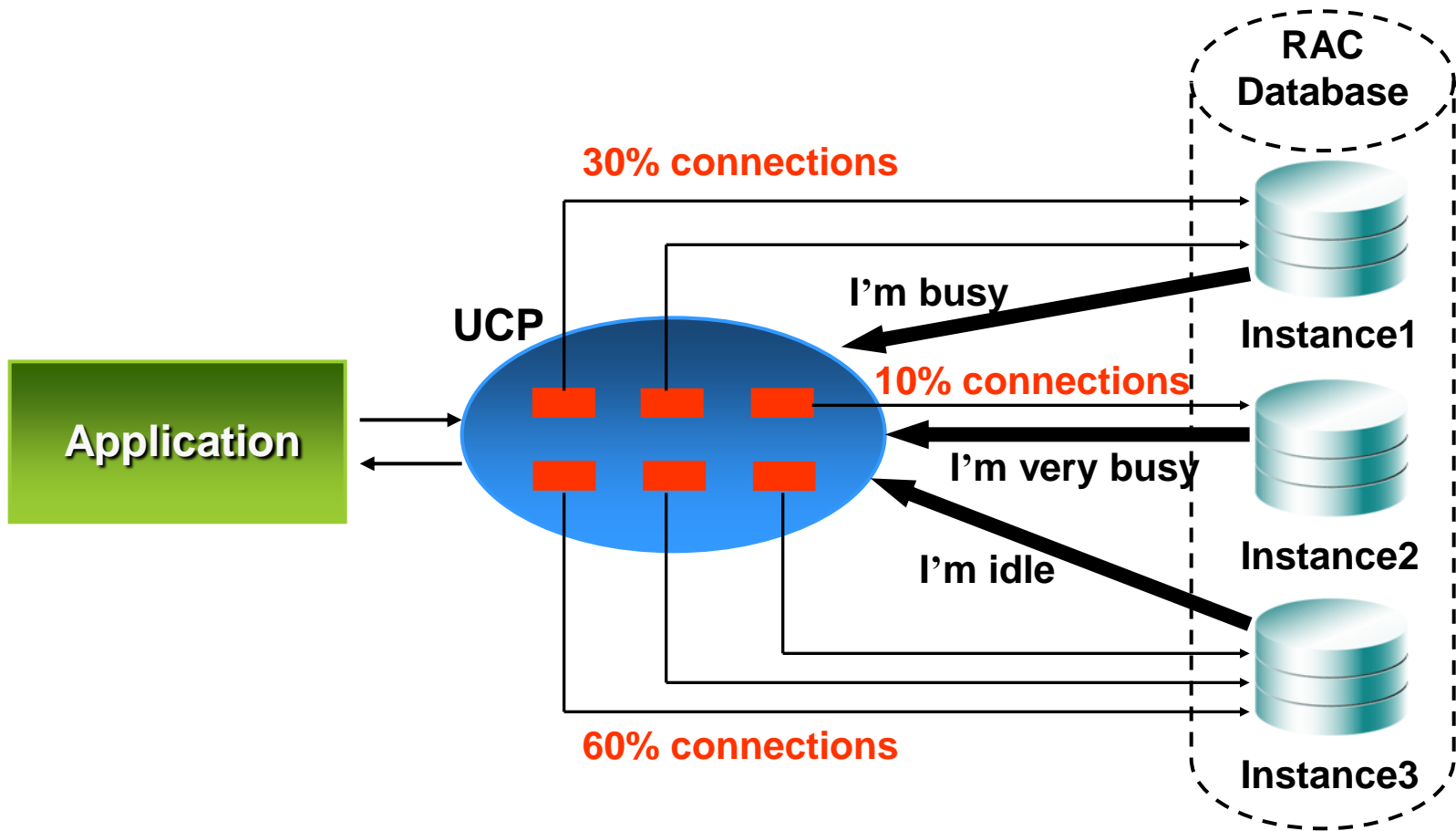
UCP

Runtime Connection Load Balancing

- Manages pooled connections for high performance and scalability
- Receives continuous recommendations on the percentage of work to route to Database instances
- Adjusts distribution of work based on different backend node capacities such as CPU capacity or response time
- Reacts quickly to changes in Cluster Reconfiguration, Application workload, overworked nodes or hangs

Runtime Connection Load Balancing

How It Works



Scale Java Connections with RCLB

- Client / mid-tier: enable Fast Connection Failover
- Server: enable RAC load-balancing advisory (LBA)
- Server: set LBA GOAL for each DB service
 - NONE (default)
 - GOAL_SERVICE_TIME – best overall service time
 - GOAL_THROUGHPUT – best overall throughput
- Server: set DB listener CLB_GOAL to SHORT
 - CLB_GOAL_SHORT – connection load balancing uses RAC Load Balancing Advisory
 - CLB_GOAL_LONG – for applications that have long-lived connections

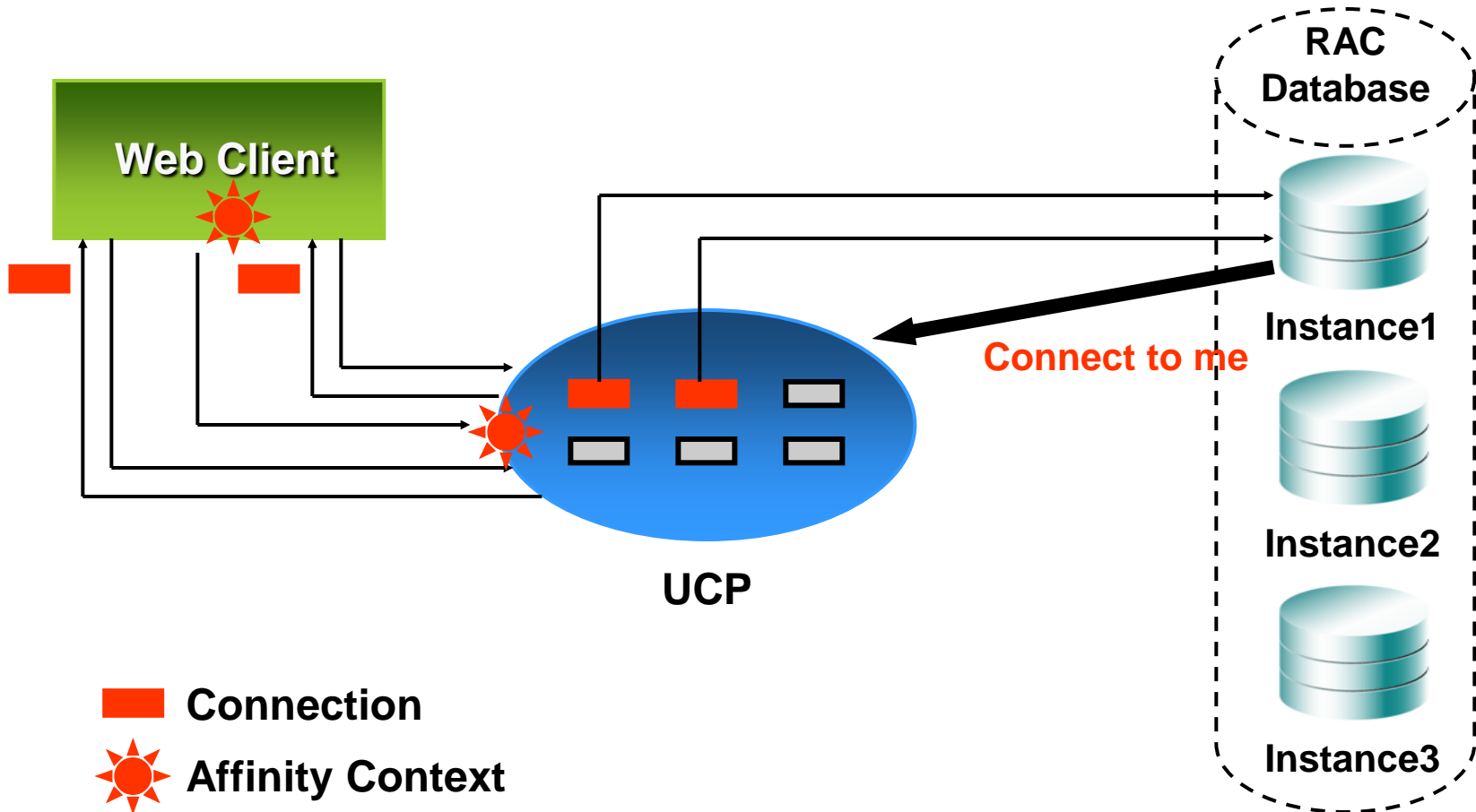
UCP

Web-Session Affinity

- The first connection request uses RCLB to select a connection
- Subsequent requests enforce Affinity
- Connection selection falls back to RCLB after Affinity ends
- Affinity is only a hint
 - Pool resorts to RCLB whenever a desired connection is not found

Web-Session Affinity

How It Works



Implement Web-Session Affinity

```
PoolDataSource pds = PoolDataSourceFactory.getPoolDataSource();
pds.setONSConfiguration("nodes=racnode1:4200,racnode2:4200");
pds.setFastConnectionFailoverEnabled(true);

ConnectionAffinityCallback cbk = new MyCallback();
Pds.registerConnectionAffinityCallback(cbk);
.....

class MyCallback implements ConnectionAffinityCallback
{
    .....
    Object appAffinityContext = null;
    AffinityPolicy policy = AffinityPolicy.WEBSESSION_AFFINITY;

    public boolean setConnectionAffinityContext(Object cxt)
    { appAffinityContext = cxt; }

    public Object getConnectionAffinityContext()
    { return appAffinityContext; }
    .....
}
```

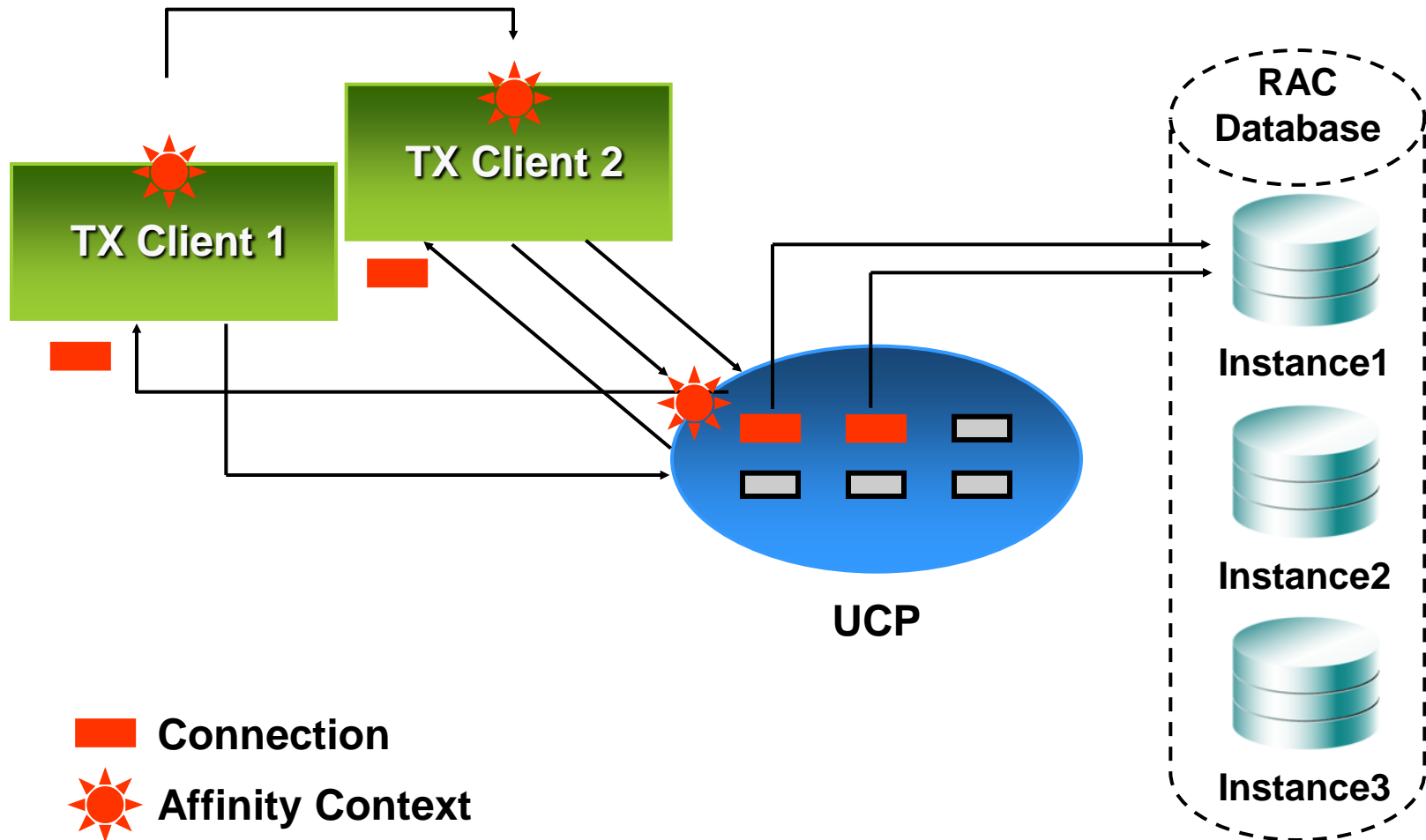

UCP

Transaction Based Affinity

- Transaction affinity is the ability to automatically localize a global transaction to a single RAC instance
- Enables XA and RAC to work together with optimal performance – Eliminates single DTP service limitation for XA/RAC
- Transaction Affinity scope is the life of a global transaction
 - First connection request for a global transaction uses RCLB
 - Subsequent requests uses affinity and are routed to the same RAC instance when XA first started

Transaction Based Affinity

How It Works



Implement Transaction Based Affinity

```
PoolDataSource pds = PoolDataSourceFactory.getPoolDataSource();
pds.setFastConnectionFailoverEnabled(true);

ConnectionAffinityCallback cbk = new MyCallback();
Pds.registerConnectionAffinityCallback(cbk);
.....

class MyCallback implements ConnectionAffinityCallback
{
    .....
    Object appAffinityContext = null;
    AffinityPolicy policy = AffinityPolicy.TRANSACTION_AFFINITY;

    public boolean setConnectionAffinityContext(Object cxt)
    { appAffinityContext = cxt; }

    public Object getConnectionAffinityContext()
    { return appAffinityContext; }
    .....
}
```


Java in the Database

Basic Example

```
public class Hello
{
    static public void world()
    {
        System.out.println("Hello world");
        return; }
}
```

**Java
Source**

```
> loadjava -v -r -u scott/tiger Hello.class
```

Load

```
SQL> create or replace procedure hello
    as language java
    name 'Hello.world()';
```

**Publish
to SQL**

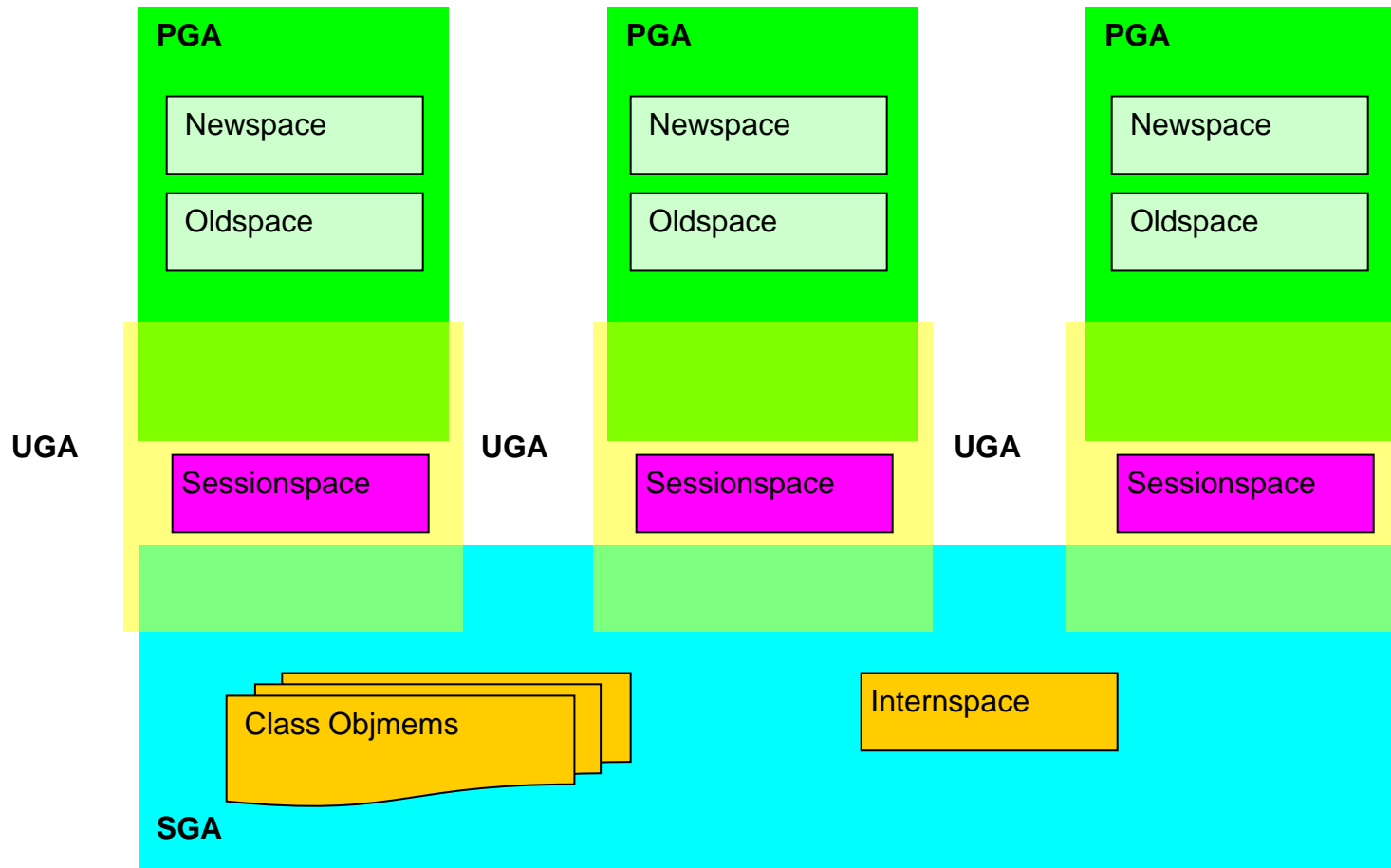
```
SQL> call hello();
Hello world
```

Invoke

Rationales for Java in the Database

- Why Java
 - General purpose programming language
 - #1 programming language (TIOBE index)
- Java VM
 - Ensures Java Portability
 - Not just Java, any language that compiles to Java byte codes
 - <http://www.is-research.de/info/vmlanguages/>
- Why in the database
 - In-place data processing
 - Java methods as triggers and stored procedures
 - Ability to extend database capabilities with Java libraries

Java in the Database Architecture



Decide When to Adopt Java in the Database

Why Not PL/SQL

- PL/SQL is better for direct manipulation of SQL types
- Java more effective for algorithmic (data logic)
- There are more Java developers
- Java has superior built-in functionalities
- Java has a wide range of libraries
- There are things you cannot do (easily) in PL/SQL

Decide When to Adopt Java in the Database

Why Not in a Middle-tier or Client

- Java EE belongs to Middle-tier
- Java SE with little /no-SQL works better in the mid-tier
- Java SE with SQL (JDBC) works better/faster in the database
 - Avoid the round trip cost for massive data processing
 - System and Application classes shared across the instance
 - Extreme Scalability
 - Relatively simple to move code from client to database and back

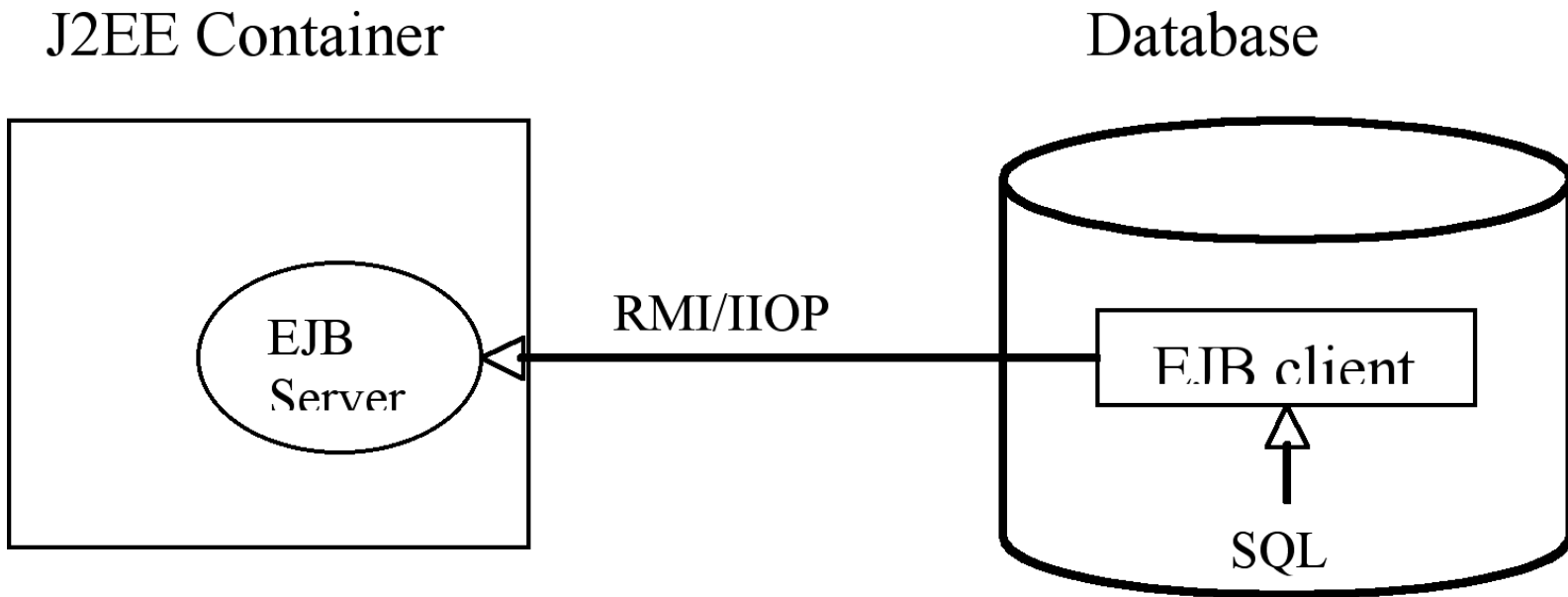
Java in the Database

What For

- Portable Data Logic
- Custom Alert applications that monitor business data
- Parsers for various File Formats (txt, zip, xml, binary)
- Custom Md5 CRC
- Produce PDF files from Result Set
- Trigger-based Notification System using RMI
- Secure Credit-Card Processing using JSSE
- Sending emails with attachment from within the database
- Execute external OS commands and external procedures
- Implement Image Transformation and Format Conversion (GIF, PNG, JPEG, etc)
- Implement database-resident Content Management System
- HTTP Call-Out
- JDBC Call-Out
- RMI Call-Out to SAP
- Web Services Call-Out
- Messaging across Tiers
- **RESTful Database Web Services***
- **Lucene Domain Index***
- JDBC belongs to the database
 - No Data Shipping
 - In-place LOB Manipulation

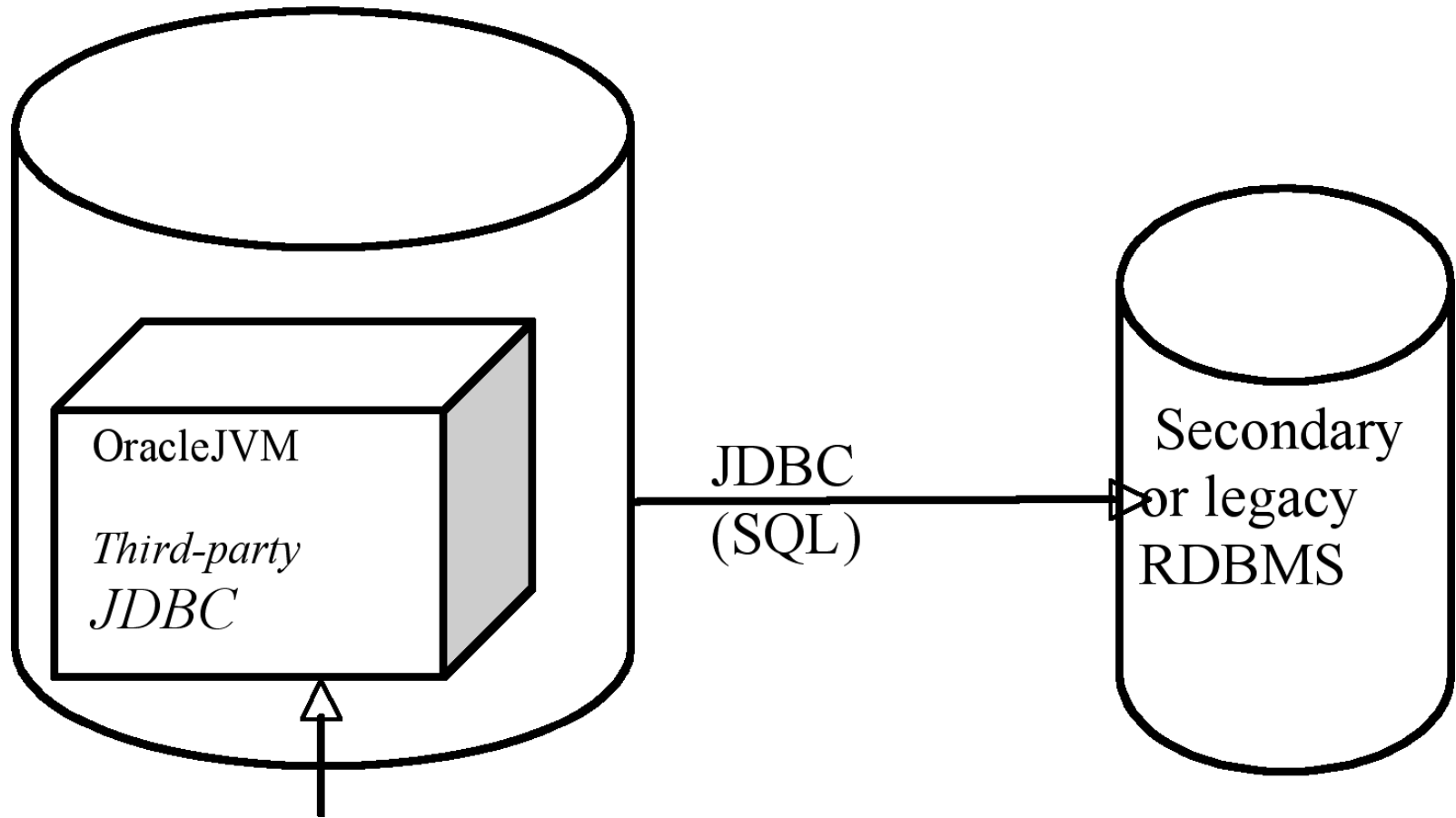
Java in the Database: What For?

EJB Callout



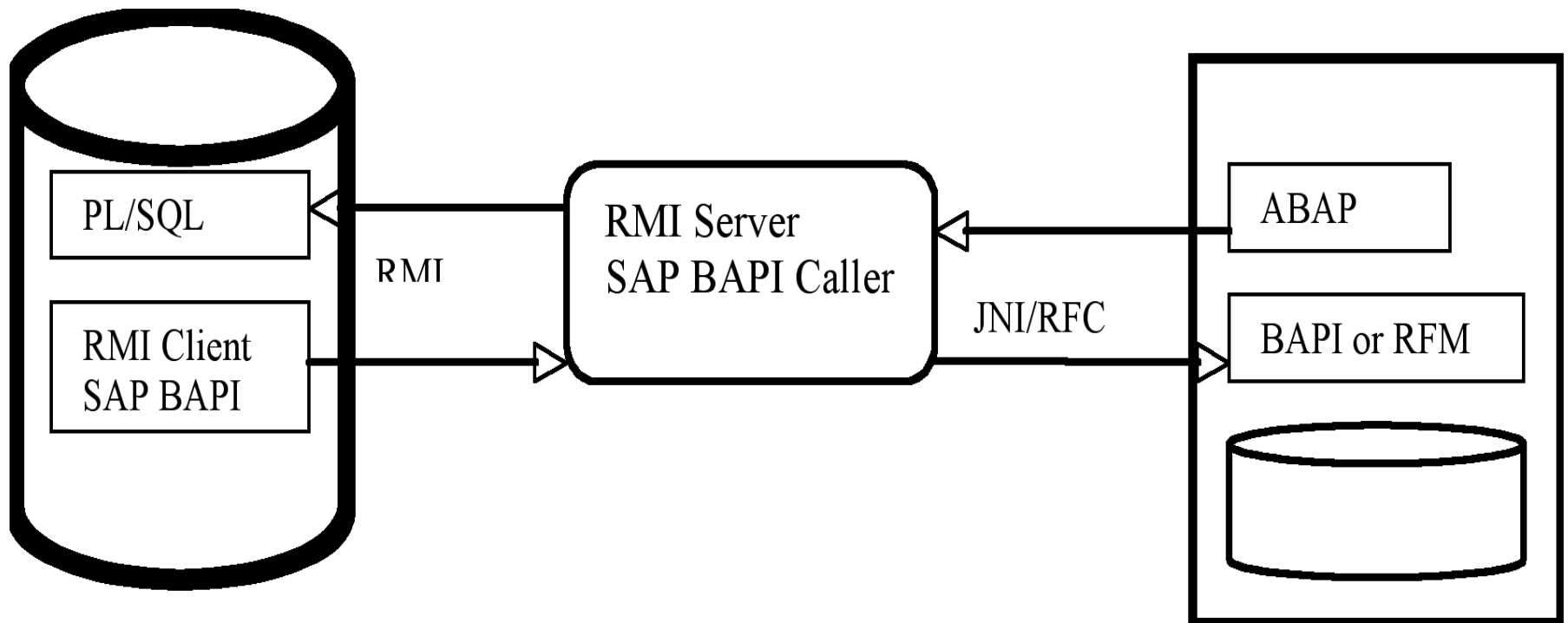
Java in the Database: What For?

JDBC Callout to Non-Oracle RDBMS



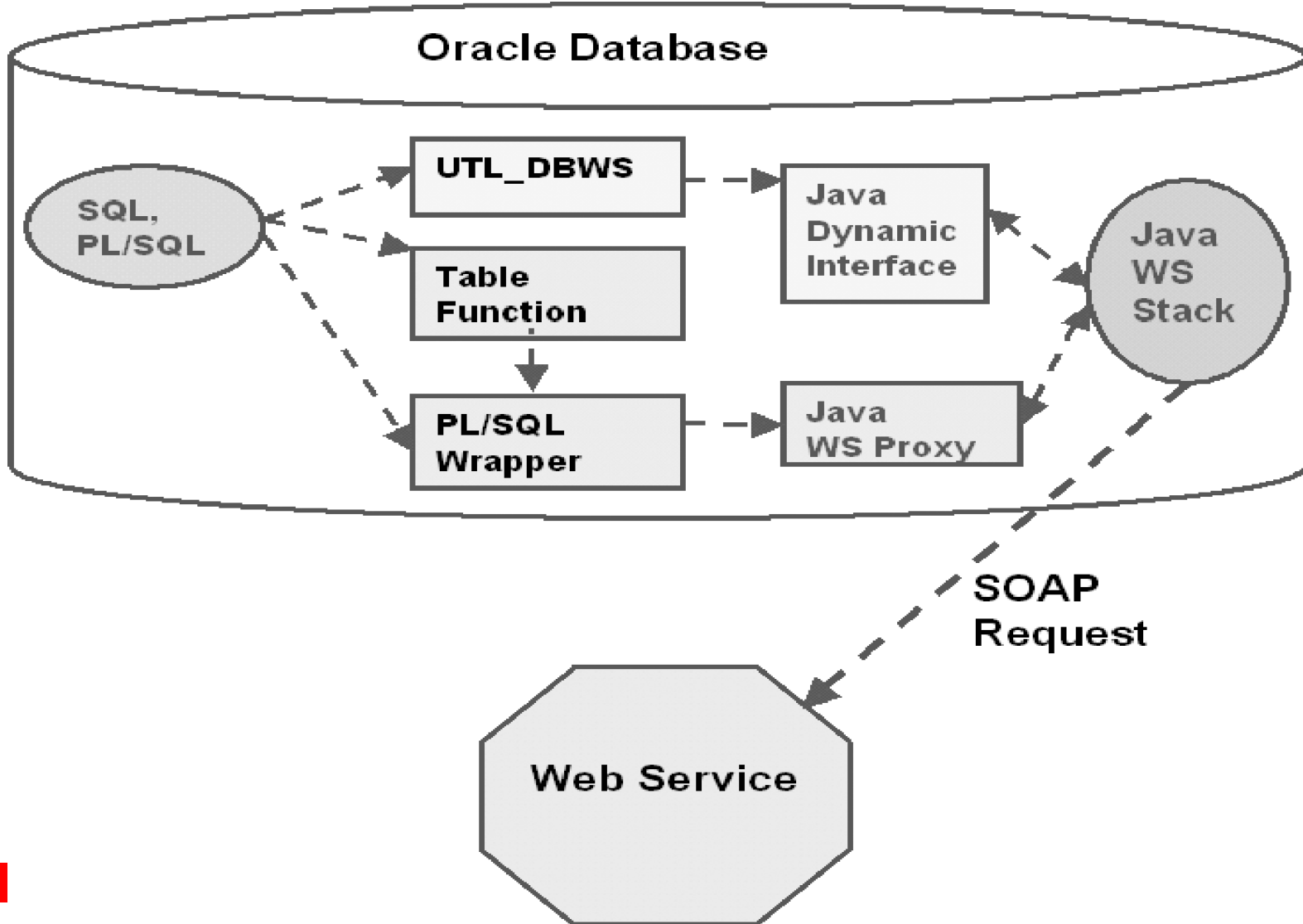
Java in the Database: What For?

SAP Callout



Java in the Database: What For?

Database as Web Services Consumer



Java in the Database Summary

- Business Logic Runs Directly in the Database and Process Data *In Situ*,
- Eliminates Network Roundtrip
- Code Shipping is Cheap
- Less Moving Parts
- Perfect for Data-Intensive Problems
- Good for Data-and-Compute intensive

More Resources

- OTN Portal Oracle Database Java Products

<http://www.oracle.com/technetwork/database/enterprise-edition/index-097123.html>

- Java Developers, JDBC & UCP Guides

http://www.oracle.com/pls/db112/to_pdf?pathname=java.112/e16548.pdf

http://www.oracle.com/pls/db112/to_pdf?pathname=java.112/e12265.pdf

http://download.oracle.com/docs/cd/B28359_01/java.111/b31225.pdf

- Java Developer's Perspective on Oracle database 11g

<http://www.oracle.com/technetwork/database/enterprise-edition/appdev-java-developers-perspective--132536.pdf>

- JDBC and SQLJ Forums

<http://forums.oracle.com/forums/category.jspa?categoryID=288>

<http://forums.oracle.com/forums/forum.jspa?forumID=65>



Q & A

Hardware and Software

ORACLE®

Engineered to Work Together

ORACLE®