
ORACLE®

ORACLE®

Working with PHP and Oracle

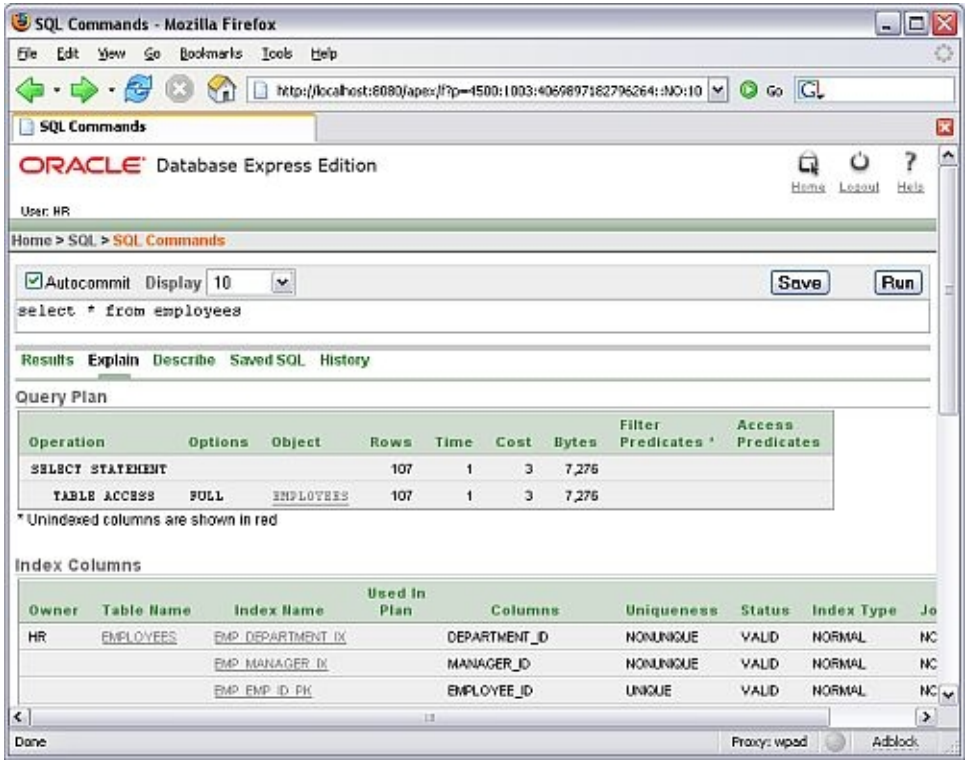
Alison Holloway
Product Manager

Working with PHP and Oracle

- Oracle products
- OCI8 extension
- Connection management
- Improving performance
- XML
- Oracle resources

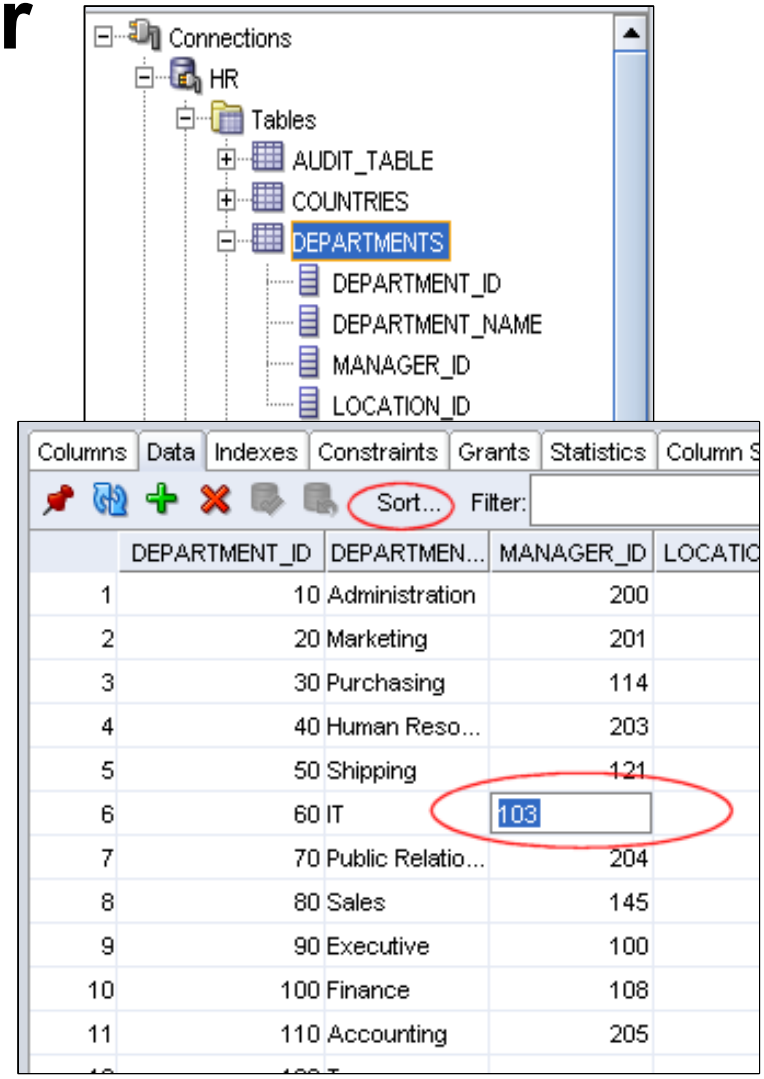
Oracle Database Express Edition

- **Free** to download, develop, deploy, and distribute
- Small footprint
- 32-bit Linux and Windows
- Free OTN community forum for support
- Uses native install on Windows and Linux
- Application Express GUI



Oracle SQL Developer

- **Free and supported**
- Browse, create, and update schema objects
- Create, edit, and debug PL/SQL
- SQL Worksheet (commands, scripts)
- Run or create reports
- Linux, Windows, Mac OS
- Oracle Database 9.2.0.1 or later



PHP and Oracle

- Long time commitment to PHP and Zend
- Thousands of developers use Oracle and PHP
- Joint effort with Zend
  - Zend Core for Oracle
  - Improved and refactored OCI8 extension
- Participates in communities and expert groups
  - Java and Scripting (JSR 223)
  - PHP Collaboration Project

PHP Extensions

- Abstraction Libraries
  - Database independence
  - Lowest common denominator
  - PEAR DB
  - PEAR MDB2
  - ADOdb
- ODBC
- PDO
- OCI8

OCI8 Extension

- Refactored by Oracle and Zend
  - Zend Core for Oracle
- Contributed back to PHP community
  - PHP 5.1.2 onwards
- Same API as original OCI8
- New php.ini parameters for
  - Persistent connection management
  - Row prefetching
  - Client side statement cache

OCI8 Extension

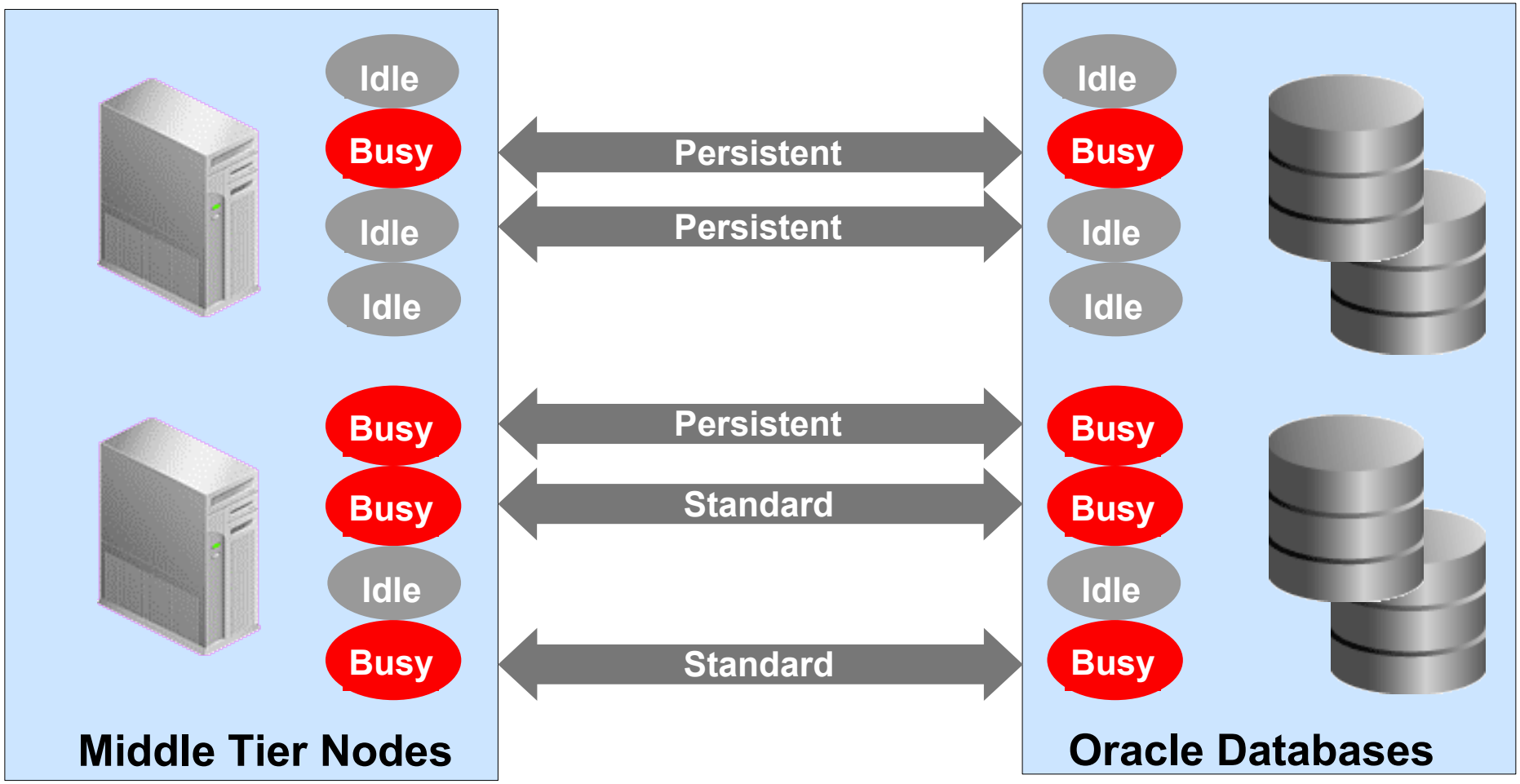
- Main Oracle database driver

```
<?php
  $conn = oci_connect('hr', 'hr', '//localhost/XE');
  $sql = oci_parse($conn, 'select city from locations');
  oci_execute($sql);
  while ($row = oci_fetch_assoc($sql))
    echo $row['CITY'] . "<br>";
?>
```

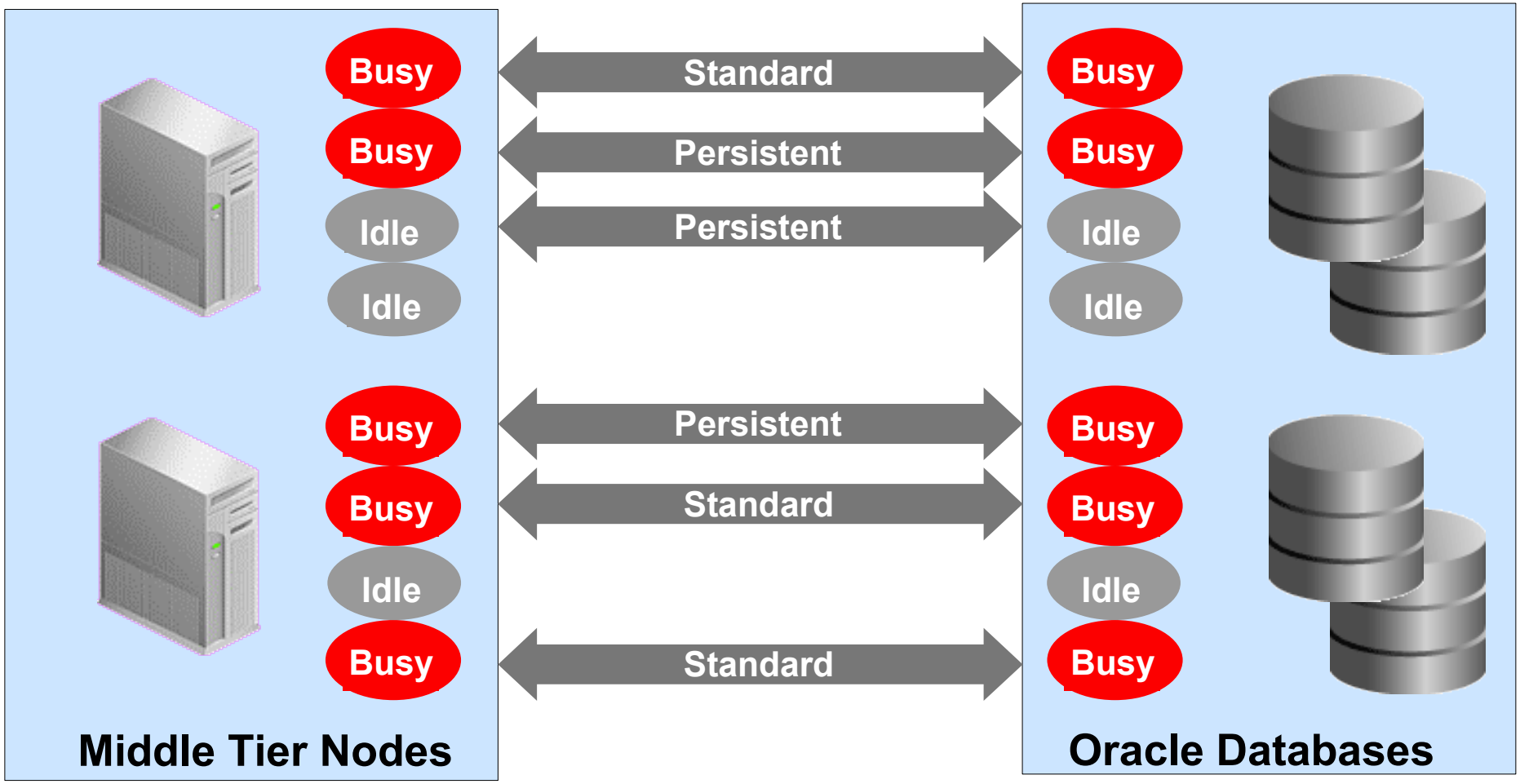
OCI8 Connections

- Standard Connection
  - oci_connect()
- Multiple Unique Connections
  - oci_new_connect()
- Persistent Connection
  - oci_pconnect()
  - oci8.max_persistent
  - oci8.persistent_timeout
  - oci8.ping_interval
- Close connections
  - oci_close()
  - oci8.old_oci_close_semantics

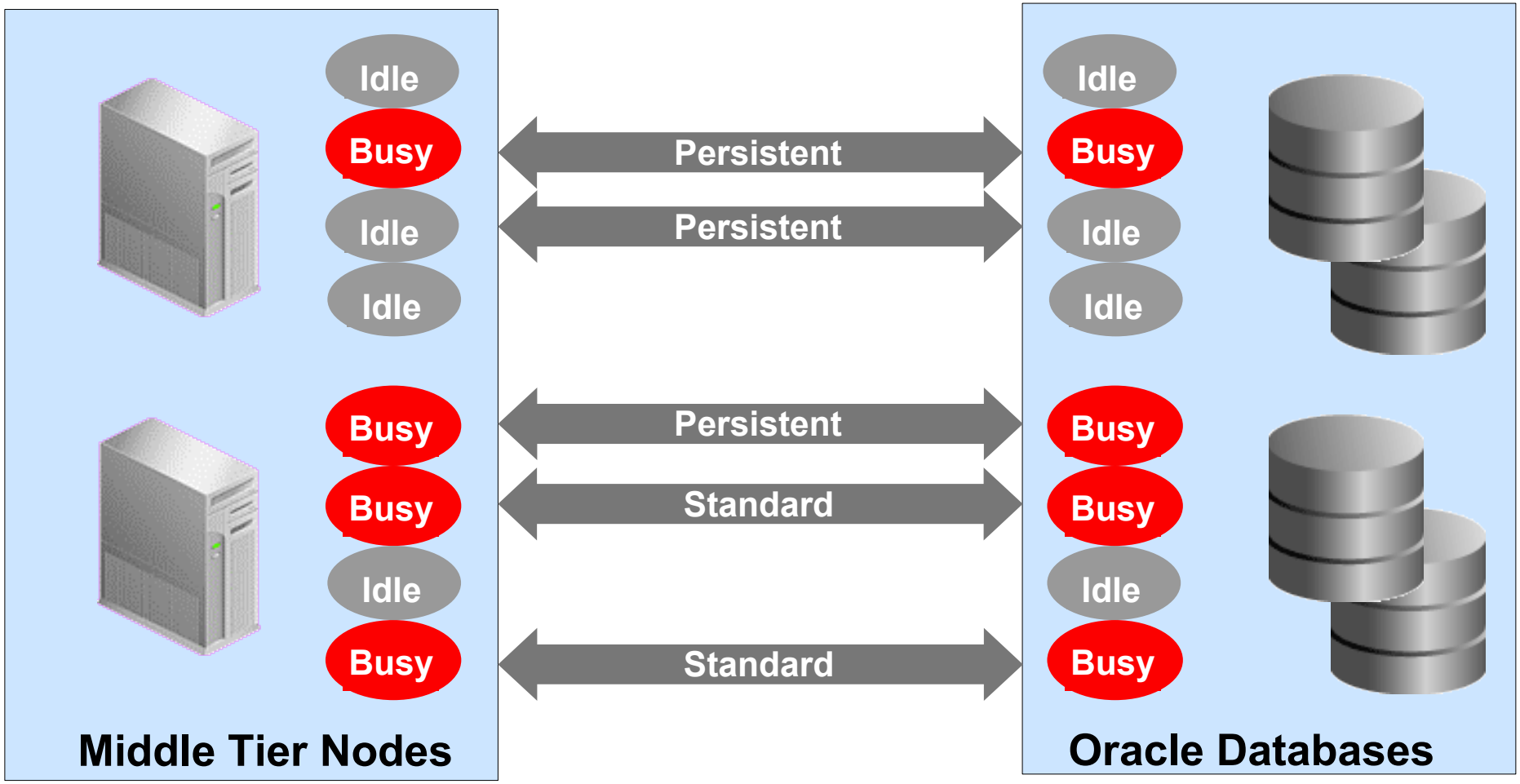
Database Connections



Database Connections



Database Connections



Connection Pooling

- Apache single threaded for PHP
- Options:
  - Repeatedly connect/disconnect using standard connections
  - Use persistent connections
- Inefficient use of the database resources

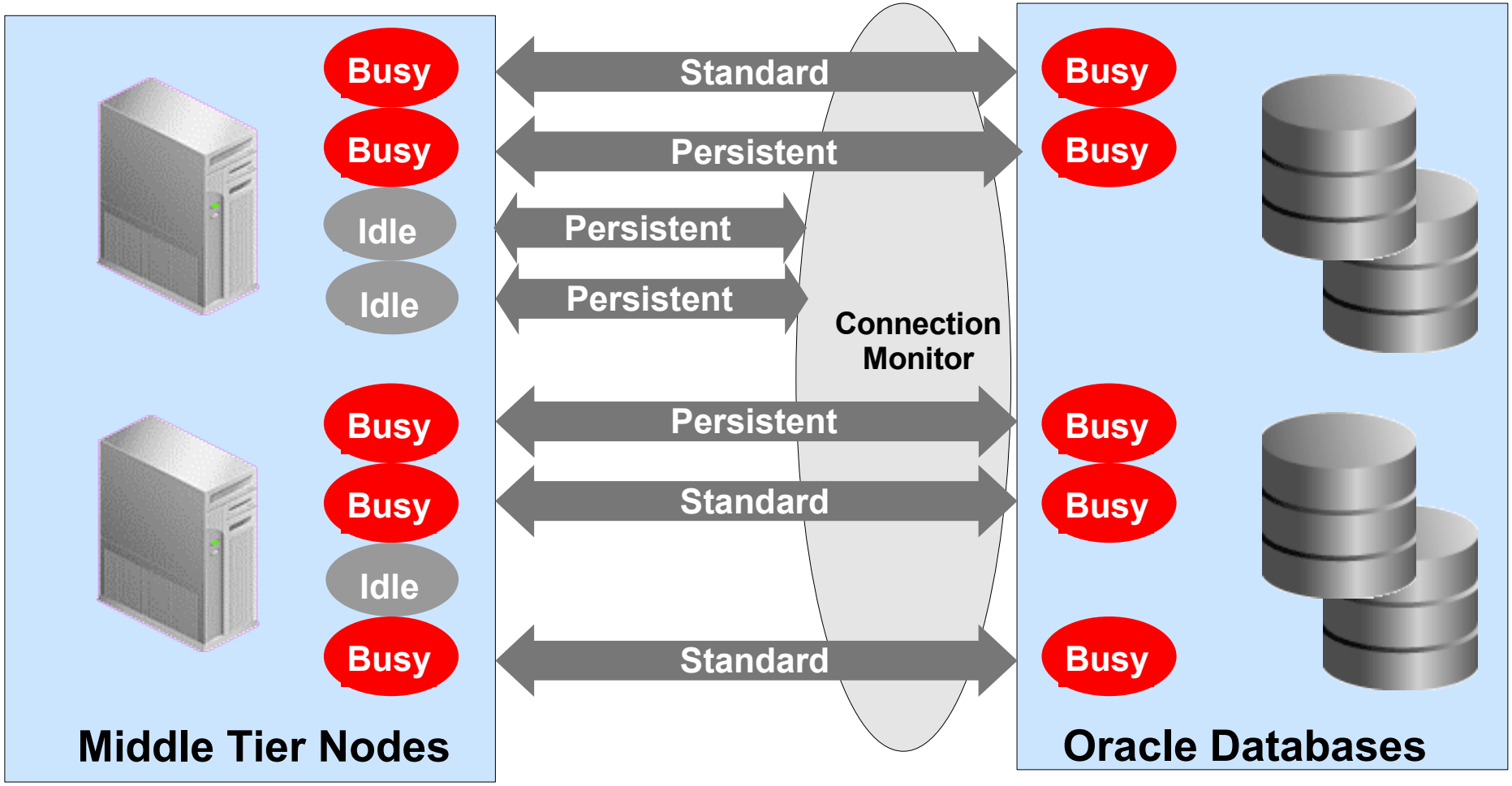
Database Resident Connection Pooling

- Pools Oracle dedicated server connections
- Shares connection pool across Apache processes
- Shares connection pool across middle tier instances
- Co-exists with Oracle Dedicated Servers, Shared Servers
- Works in RAC

Database enhancements described here are in the next major release

PHP enhancements described here are proposed

Database Resident Connection Pooling



Database Resident Connection Pooling

- Reduces number of connections
- Pooling is enabled by DBA on database server
- Change PHP connect string
  - `hostname/service:POOLED`
  - `(SERVER=POOLED)`
- Set parameter in php.ini:
  - oci8.connection_class = "<*application_name*>"
- Client directed to Database Resident Connection Pool
- Connection in pool "locked" when requested by client
- Connection "released" back to pool when client done

Database Resident Connection Pooling

- Enhanced OCI8 extension
- Oracle database and client libraries
- Code compatible: no application code changes
- Version compatibile: application works as is with
  - Existing Oracle databases (no pooling)
  - Database with pooling disabled
  - Database with pooling enabled
- DBA monitoring and configuration options

Database Resident Connection Pooling with Standard APIs

- oci_pconnect()
  - Very fast binding to an existing connection from pool
  - Re-uses pooled server process from pool
  - Re-uses pooled server session
- oci_connect/oci_new_connect()
  - Faster new connections!
  - Re-uses pooled server process from pool
  - Session is recreated
- oci_close()
  - Releases connection back to pool

Bind Variables

- Like "%s" print format specifiers
- Application re-executes statements with different values
- Improves database throughput
- Help prevent SQL Injection attacks

Bind Variables

```
$s = oci_parse($c,
      "select last_name from employees
       where employee_id = :eidbv");
$myeid = 101;
oci_bind_by_name($s, ":EIDBV", $myeid);
oci_execute($s);
oci_fetch_all($s, $res);
echo "Last name is: ". $res['LAST_NAME'][0] ."<br>\n";
$myeid = 102;
oci_execute($s);   // No need to re-parse
oci_fetch_all($s, $res);
echo "Last name is: ". $res['LAST_NAME'][0] ."<br>\n";
```

PL/SQL

- Procedural language extension to SQL
  - Variables
  - Conditions
  - Arrays
  - Exceptions
- Stored in database
- Stable, portable, secure
- Procedures, functions, triggers
- Presupplied packages
- Fastest in Oracle 10.2

PL/SQL Stored Procedure and PHP

```
create or replace procedure
myproc(a in varchar2, b in number) as
begin
   insert into mytable (mydata, myid) values (a, b);
end;
/
```

```
<?php
    $c = oci_connect('hr', 'hr', '//localhost/XE');
    $s = oci_parse($c, "call myproc('mydata', 123)");
    oci_execute($s);
?>
```

Database Transactions

- oci_execute() auto-commits
- Commits cause
  - Round-trips to the database
  - Database I/O

```
$s = oci_parse($c, "insert into t values ('data')");
oci_execute($s, OCI_DEFAULT);   // Not committed
...
oci_commit($c);
```

Non Transactional Insert

This is slow

```
function do_simple_insert($c, $a)
{
  foreach ($a as $v) {
    $s = oci_parse($c,
     "insert into ptab (pdata) values ('".$v."')");
    $r = oci_execute($s); // Committed
  }
}
```

Transactional Insert

This is better

```
function do_transactional_insert($c, $a)
{
  $s = oci_parse($c,
        'insert into ptab (pdata) values (:bv)');
  oci_bind_by_name($s, ':bv', $v, 20, SQLT_CHR);
  foreach ($a as $v)
    $r = oci_execute($s, OCI_DEFAULT); // Not committed
  oci_commit($c); // Committed
}
```

Bulk Inserts in PL/SQL

```
create or replace package mypkg as
  type arrtype is table of varchar2(20)
    index by pls_integer;
  procedure myproc(p1 in arrtype);
end mypkg;
create or replace package body mypkg as
  procedure myproc(p1 in arrtype) is
  begin
    forall i in indices of p1
      insert into ptab values (p1(i));
  end myproc;
end mypkg;
```

Bulk Inserts in PHP

May be fastest

```
function do_bulk_insert($c, $a)
{
  $s = oci_parse($c,
          'begin mypkg.myproc(:c1); end;');
  oci_bind_array_by_name($s, ":c1", $a,
count($a), -1, SQLT_CHR);
  oci_execute($s);
}
```

XML

- XML DB
- Stored as
  - Relational data
  - Linear as LOBs
  - XML Schema structure
- SELECT XMLELEMENT
- PL/SQL Package DBMS_XMLGEN
- XQuery

SELECT XMLELEMENT

```
$query =
  'SELECT XMLELEMENT("Employees",
     XMLELEMENT("Name", first_name),
     XMLELEMENT("Id", employee_id)) as result
    FROM employees WHERE department_id = 10';
$s = oci_parse($c, $query);
oci_execute($s);
while ($row = oci_fetch_row($s))
  foreach ($row as $item)
    echo htmlentities($item)." ";
```

```
<Employees><Name>Joanne</Name><Id>210</Id></Employees>
<Employees><Name>Jennifer</Name><Id>200</Id></Employees>
```

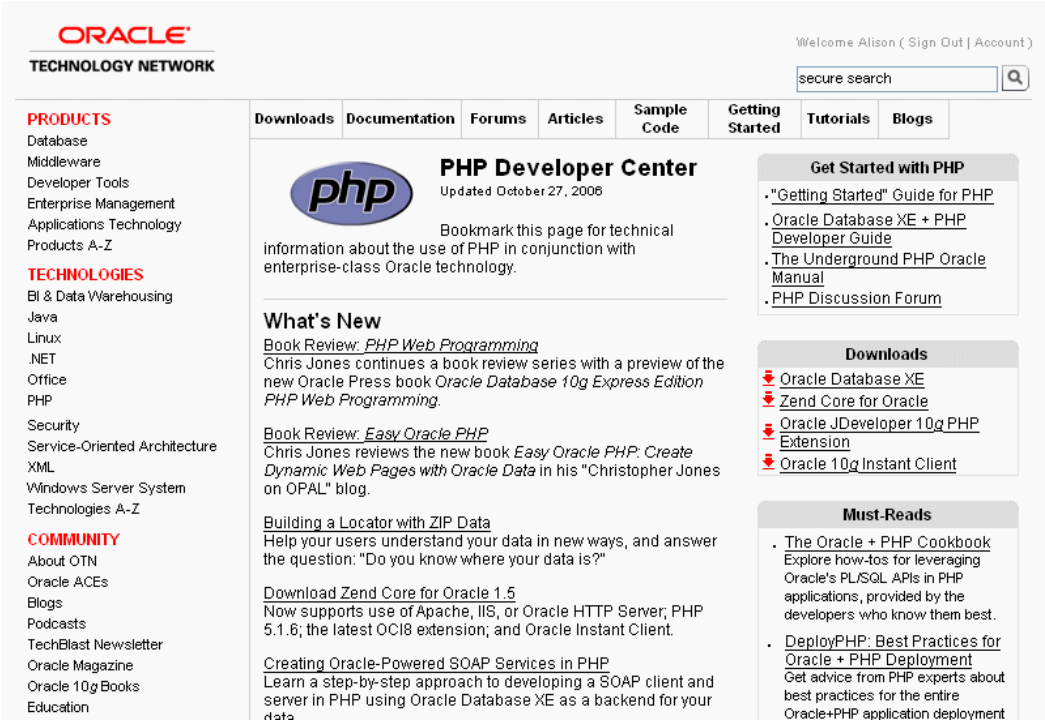
DBMS_XMLGEN Package

```
$query = "select dbms_xmlgen.getxml('
          select first_name, employee_id from employees
          where department_id = 10') xml from dual";
$s = oci_parse($c, $query);
oci_execute($s);
$res = oci_fetch_row($s);
$mylob = $res[0]->load(); // treat result as a LOB descriptor
```

```
<?xml version="1.0"?>
<ROWSET>
      <ROW>
           <FIRST_NAME>Joanne</FIRST_NAME>
           <EMPLOYEE_ID>210</EMPLOYEE_ID>
      </ROW>
      ...
</ROWSET>
```

Oracle Technology Network PHP Developer Centre

- Technical Articles
- Install guides
- Underground PHP and Oracle Manual
- Discussion forum
- JDeveloper 10*g* PHP extension
- Blogs



otn.oracle.com/php

Oracle Resources

- OTN Open Source Developer Centre
  - otn.oracle.com/opensource
- Oracle Unbreakable Linux
  - www.oracle.com/linux
- ISVs and hardware vendor
  - oraclepartnernetwork.oracle.com
- Contact me
  - alison.holloway@oracle.com
- Blogs
  - blogs.oracle.com/alison (me!)
  - blogs.oracle.com/opal (Christopher Jones)

ORACLE IS THE INFORMATION COMPANY


ORACLE®

The preceding 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.