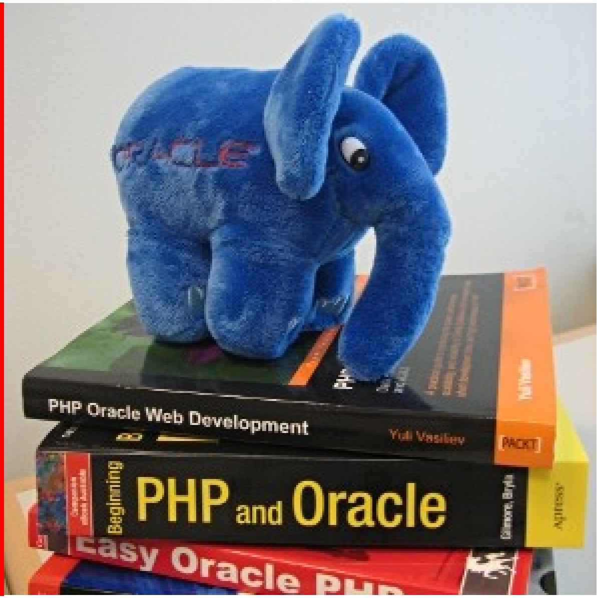
ORACLE®



# Best Practices - PHP and the Oracle Database

Christopher Jones, Product Development, Oracle

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.

# Menu

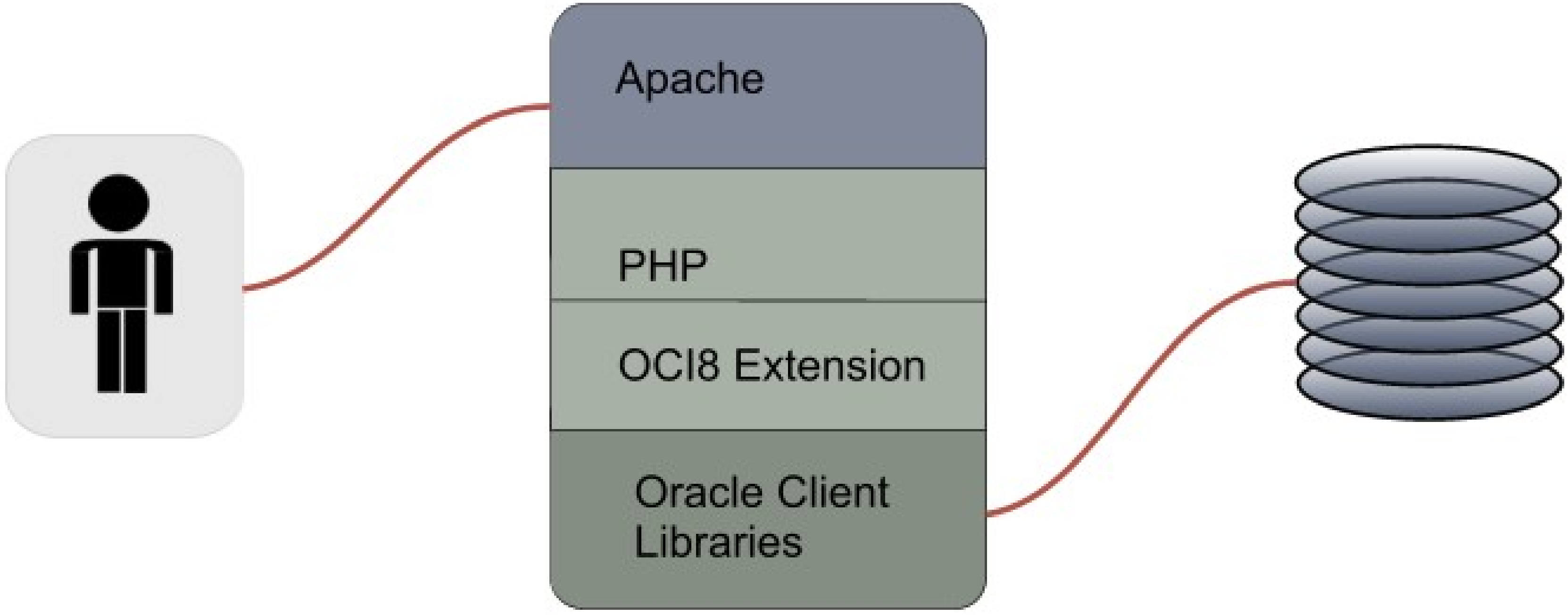
- Appetizer
- Connection Techniques
- Statement Techniques
- Caching Techniques
- Oracle 11gR2 and PHP OCI8 1.4

# What is PHP OCI8?

- Recommended Oracle Database extension
- Uses prepare/execute/fetch procedural model

```
$c = oci_connect('un', 'pw', 'localhost/orcl');
$s = oci_parse($c, 'select * from employees');
oci_execute($s);
while (($row = oci_fetch_array($s, OCI_ASSOC)) != false)
  foreach ($row as $item)
    print $item;
```

# Using OCI8 with the Oracle Database



Apache

PHP

OCI8 Extension

Oracle Client
Libraries

*Web User*

*Oracle Client Libraries*

9*i*R2, 10*g*, 11*g*
Any Platform

*Oracle Database*

8*i*, 9*i*, 10*g*, 11*g*
Any Platform

# Oracle Database 11.1 and PHP

- Oracle 11.1 was released August 2007
- Connection Pooling - DRCP
- Database Server Query Result Cache
- Client (aka PHP) Query Result Cache
- Continuous Query Notification
- Cube Organized Materialized Views
- . . .

# PHP OCI8 1.3

- php.net
  - PHP 5.3 Source code, Windows binaries
- PECL - PHP Extension Community Library
  - For updating PHP 4 - 5.2 with OCI8 1.3
- http://oss.oracle.com/projects/php
  - RPMs for Linux with OCI8
- Unbreakable Linux Network
  - Oracle's Linux Support program
  - OCI8 RPM available for PHP
- Zend Server
  - Linux, Windows, Mac
  - Support from Zend

# Menu

- Appetizer
- ►Connection Techniques
- Statement Techniques
- Caching Techniques
- Oracle 11gR2 and PHP OCI8 1.4

# Standard OCI8 connections

```
$c = oci_connect($un, $pw, $db);
```

- High CPU Overhead
  - Connection establishment slow: Oracle DB is process based
  - Cannot handle the Digg effect
  - One database server process per PHP user
  - Maxes out database server memory

# Persistent OCI8 Connections

```
$c = oci_pconnect($un, $pw, $db);
```

- Fast for subsequent connections
- Not closable (prior to OCI8 1.3)
- Some control configurable in php.ini

  `oci8.max_persistent`

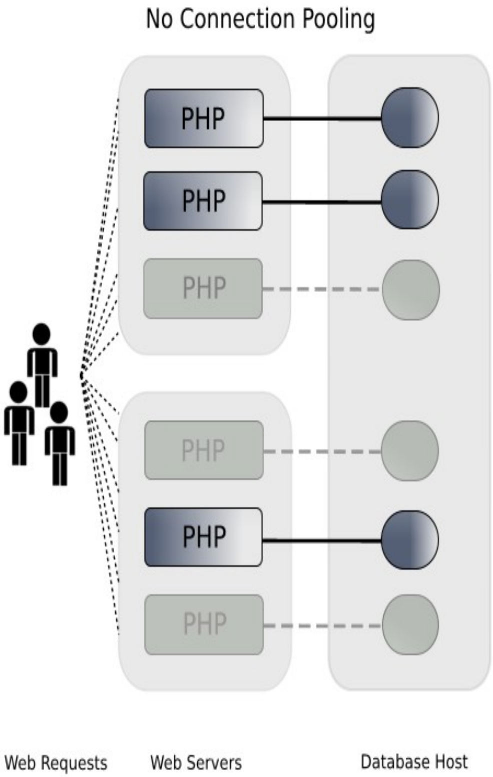
  Number of connections per PHP process

  `oci8.persistent_timeout`

  “Idle” expiry time

  `oci8.ping_interval`

  Ping after retrieving from PHP cache

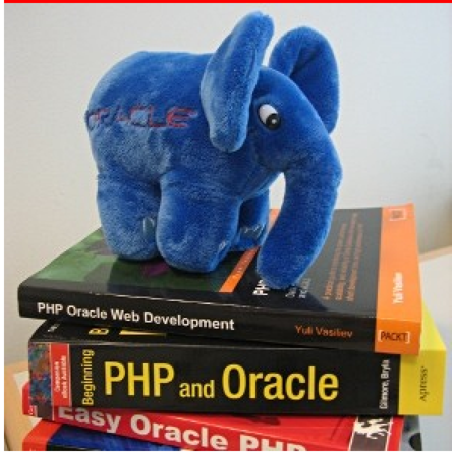


# Sidebar: Connection Quick Tips

- Connection is faster when a character set is given

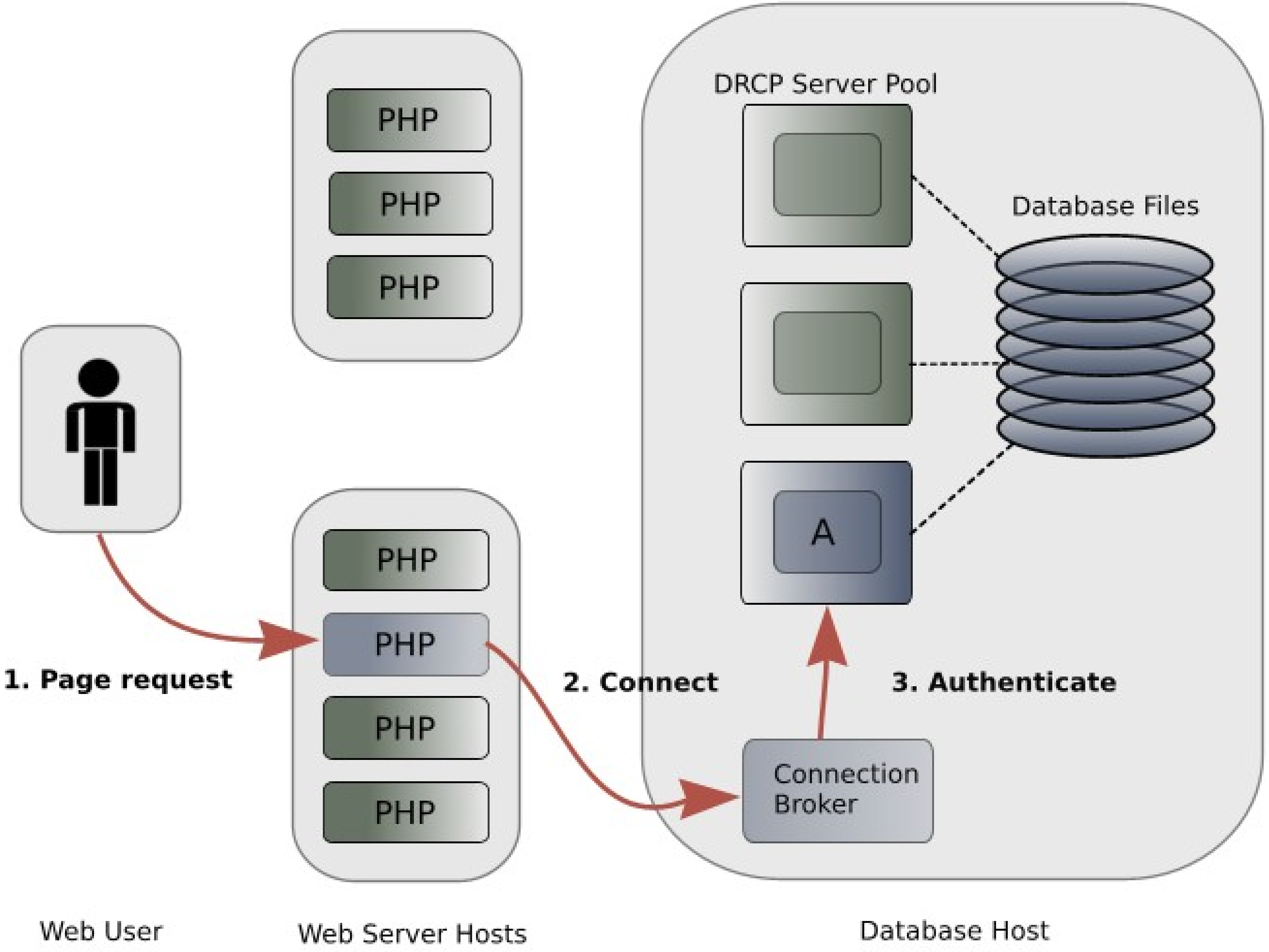
```
$c = oci_pconnect("hr", "welcome", "MYDB", 'AL32UTF8');
```

- For oci_connect() and oci_new_connect(), minimize PATH length and environment variable size of `oracle` user on database machine
  - => reduces process creation time

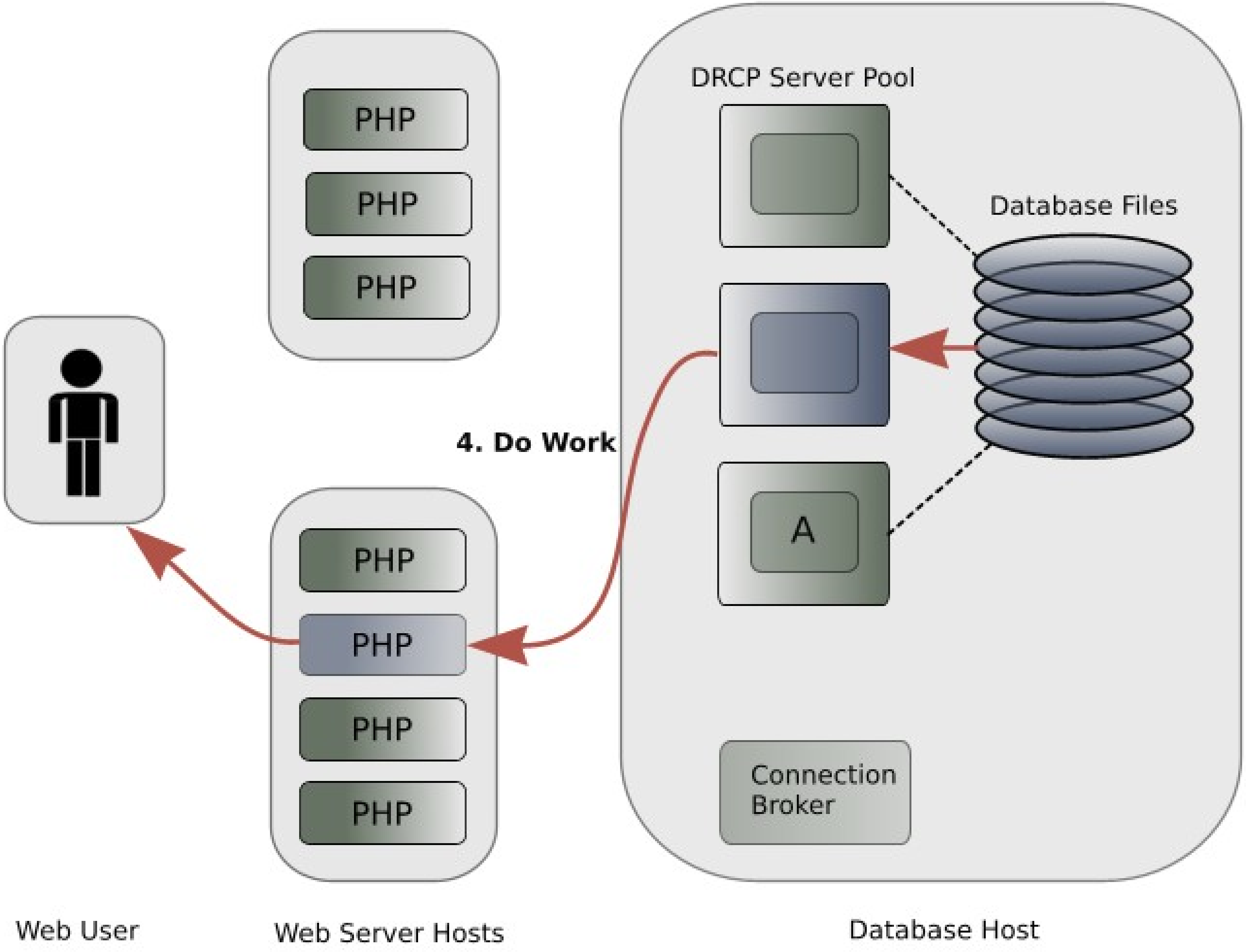


# Database Resident Connection Pooling in 11g

# Database Resident Connection Pooling

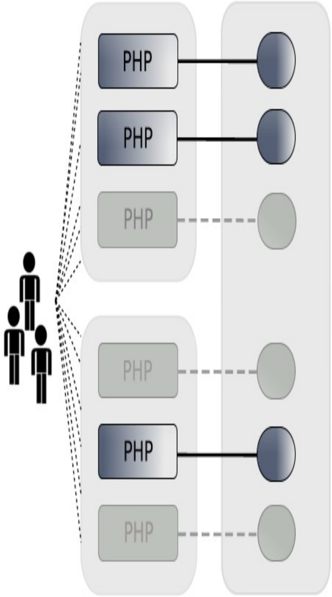


# Database Resident Connection Pooling

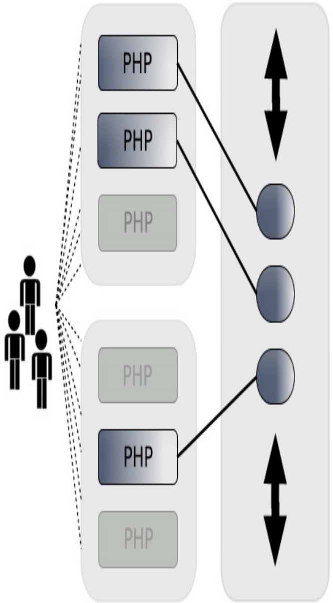


# Saving Memory with DRCP

5000 users; DRCP pool size of 100



| Dedicated Mode | | DRCP Mode |
|---|---|---|
| *5000 x 4 MB +* | DB Processes | *100 x 4 MB +* |
| *5000 x 400 KB +* | Session Memory | *100 x 400 KB +* |
| *0 =* | DRCP Broker | *5000 x 35 KB =* |
| **21 GB** | Total | **610 MB** |



# DRCP in PHP

- DRCP support is available in OCI8 1.3
  - when linked with Oracle 11g client libraries and connected to Oracle 11g
- Works with oci_connect(), oci_new_connect(), oci_pconnect()
- Use for short lived connections by same DB user

# Configuring and Starting the Pool

- Install PHP OCI8 1.3 and Oracle 11g
- Configure the pool (optional)

```
SQL> execute dbms_connection_pool.configure_pool(
pool_name              => 'SYS_DEFAULT_CONNECTION_POOL',
minsize                => 4,
maxsize                => 40,
incrsize               => 2,
session_cached_cursors => 20,
inactivity_timeout     => 300,
max_think_time         => 600,
max_use_session        => 500000,
max_lifetime_session   => 86400);
```

- Sart the pool:

```
SQL> execute dbms_connection_pool.start_pool();
```

- Set oci8.connection_class in php.ini

```
oci8.connection_class = MY_PHP_APP
```

# Using DRCP

- Add “*POOLED*” to the connect string:

```
$c = oci_pconnect($un, $pw, 'myhost/sales:POOLED');
```

Or in tnsnames.ora file connect descriptor:

```
SALES = (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)
        (HOST=myhost.dom.com)
        (PORT=1521))
        (CONNECT_DATA=(SERVICE_NAME=sales)
        (SERVER=POOLED)))
```

- Application decision to use DRCP or not

# DRCP Recommendations.

- Read the PHP DRCP whitepaper

  http://www.oracle.com/technology/tech/php/index.html
- Make sure `oci8.connection_class` is set
- Have > 1 Broker, but only a few
- Close connections when doing non-DB processing
- Explicitly control commits and rollbacks
  - Avoid unexpectedly open transactions when an `oci_close()` or end-of-scope occurs
  - Scripts coded like this can use `oci_close()` to take full advantage of DRCP but still be portable to older versions of the OCI8 extension
- Monitor `V$CPOOL_STATS` view to determine best pool size
- Don't use for long batch processes

# Sidebar: Transaction Tips

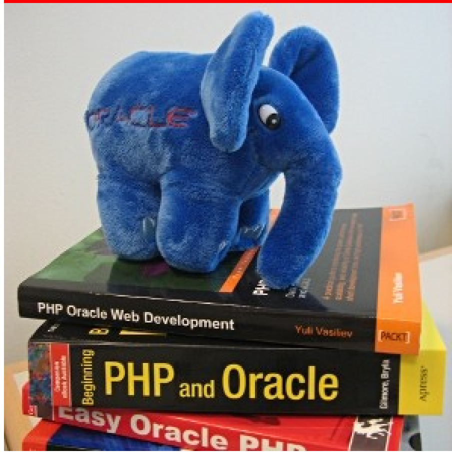
- Use transactions to avoid unnecessary commits

```
oci_execute($c, OCI_DEFAULT);
. . .
oci_commit($c);
```

- Any oci_execute($c), even for a query, will commit an open transaction

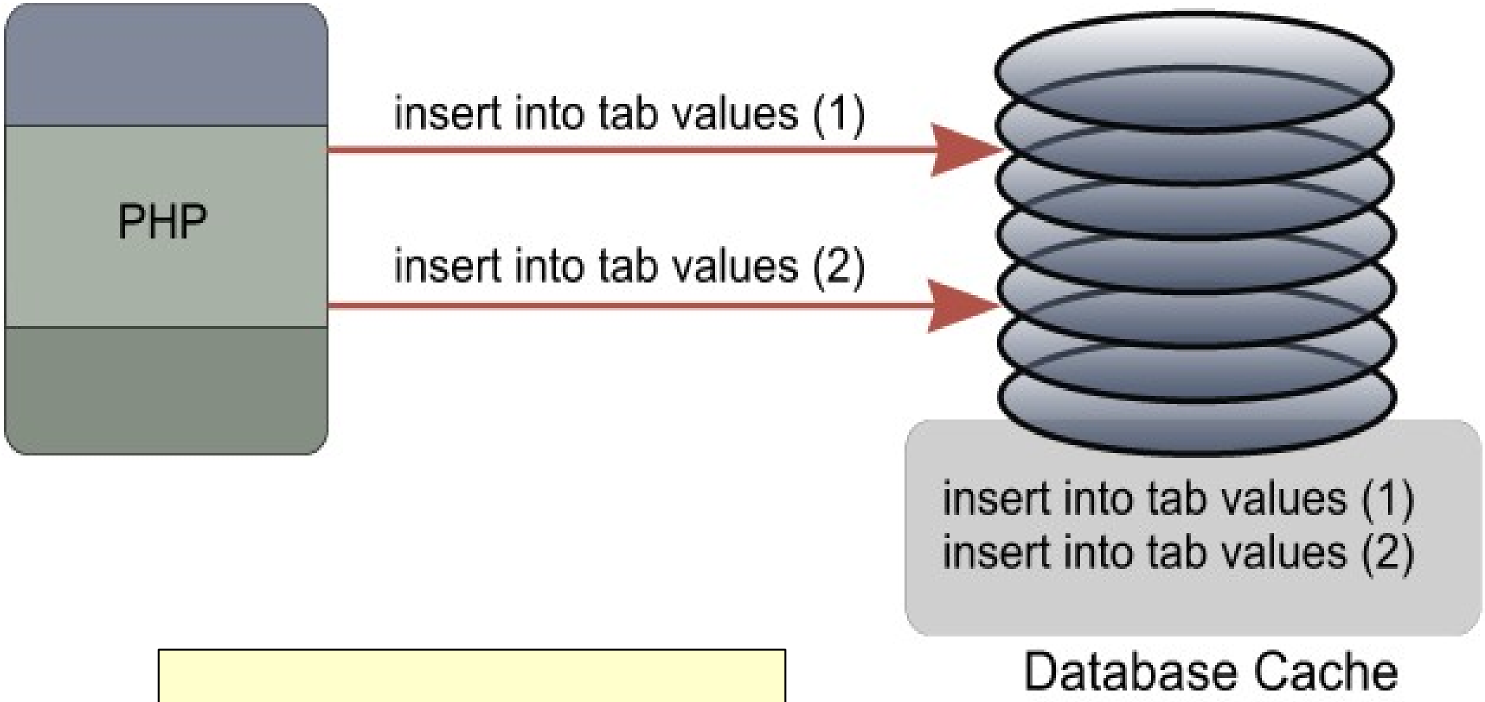
# Menu

- Appetizer
- Connection Techniques
- ►Statement Techniques
- Caching Techniques
- Oracle 11gR2 and PHP OCI8 1.4



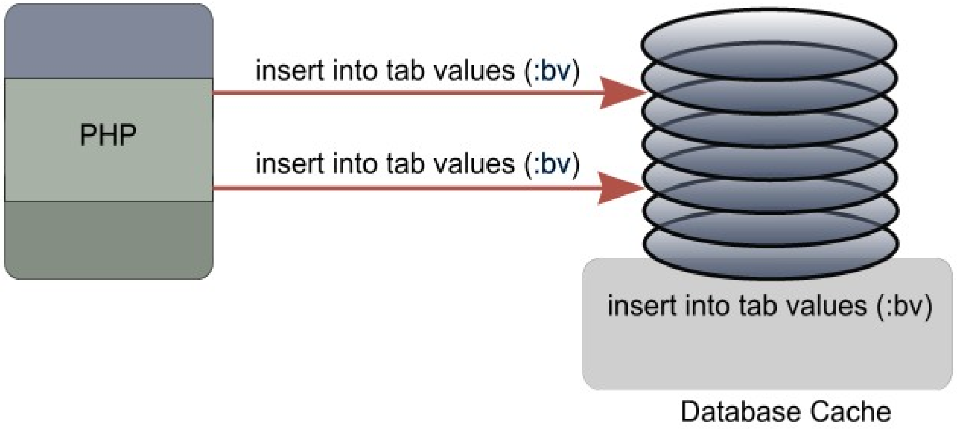
# Binding with SQL Statements

# Not Binding Gives Poor Cache Use


PHP
insert into tab values (1)
insert into tab values (2)
insert into tab values (1)
insert into tab values (2)
Database Cache

Poor use of cache

# Binding Scalars


insert into tab values (:bv)
PHP
insert into tab values (:bv)
insert into tab values (:bv)
Database Cache

```
$c = oci_connect('hr', 'hrpwd', 'localhost/orcl');
$s = oci_parse($c,'insert into tab values (:bv)');
$name = 'Jones';
oci_bind_by_name($s, ':bv', $name);
oci_execute($s);
```

# Binding Benefits

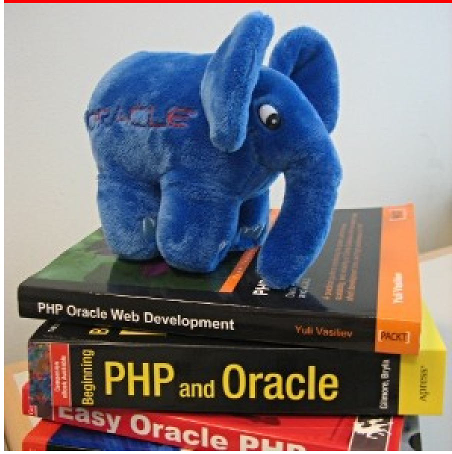
From a query example by Tom Kyte:

| | Without | With |
|---|---|---|
| Parse count (hard) | 5,000 | 1 |
| Parse time elapsed | 485 | 35 |
| Latches | 328,496 | 118,614 |

- Can change bind values and re-execute without re-parsing
- No SQL Injection worry
- Easier to write than adding quote escaping
- Overall system is more efficient
- PHP user elapsed time directly benefits

# Binding Best Practices

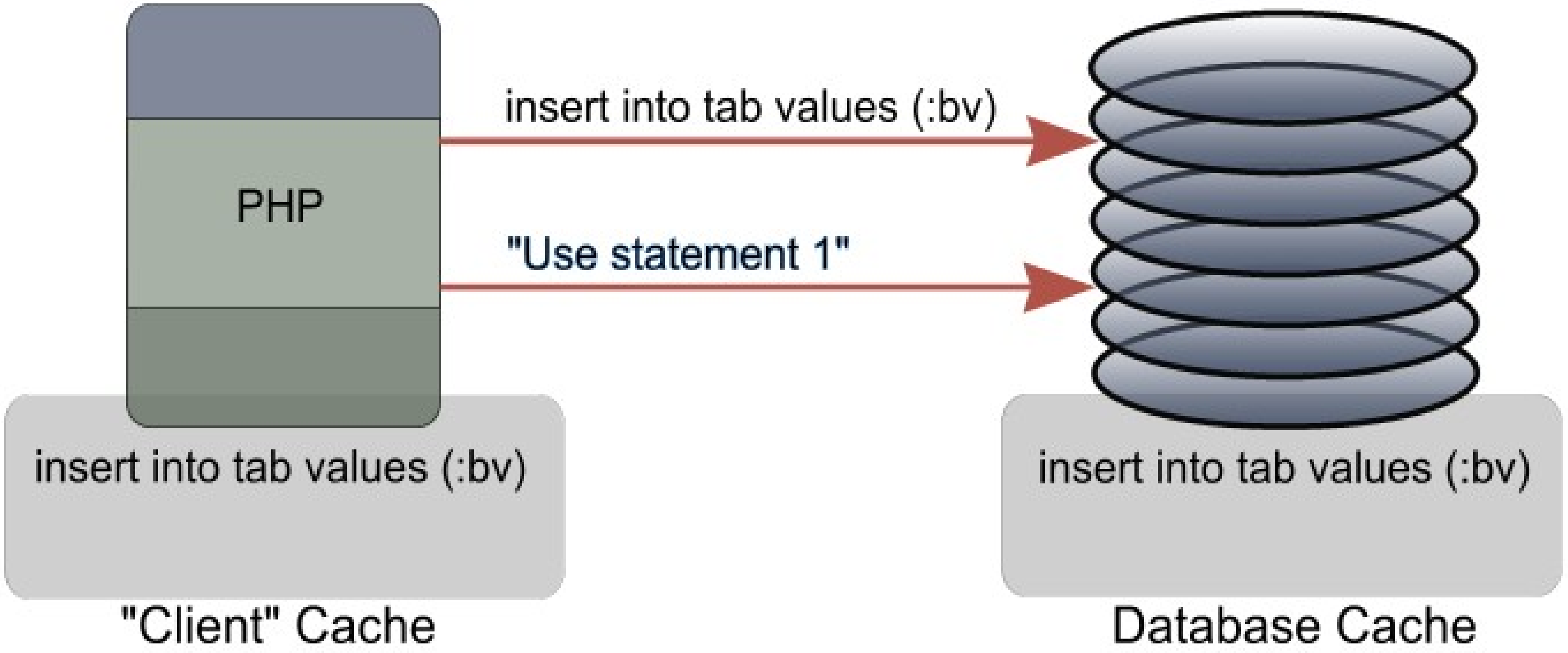
- Set length parameter to your upper data size for re-executed IN binds `oci_bind_by_name($s, “:b”, $b, 40);`
- Don't bind constants
  - Let the optimizer see them
- Long running unique queries may not benefit
  - Parse time is a relatively small cost
- CURSOR_SHARING parameter
  - Set in “session” or database init.ora
  - Makes every statement appear to have bound data, but optimizer now doesn't see constants
  - For bind-unfriendly applications
- Oracle 11g has Adaptive Cursor Sharing
  - Can have multiple execution plans for same statement



# Statement Caching

# Client (aka PHP) Statement Caching

Oracle Client library cache of statement text & meta data


insert into tab values (:bv)
PHP
"Use statement 1"
insert into tab values (:bv)
"Client" Cache
insert into tab values (:bv)
Database Cache

Less traffic and DB CPU

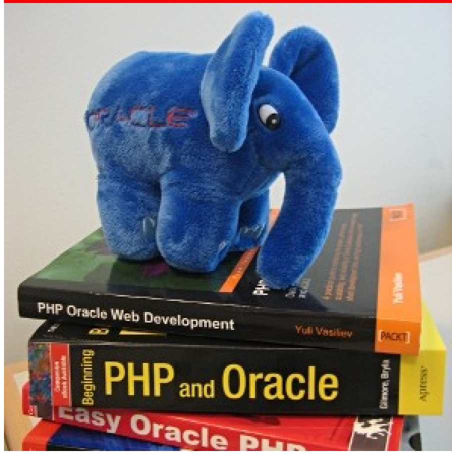
# Statement Caching Best Practices

- Enabled by default in php.ini

```
oci8.statement_cache_size = 20
```

  Unit is number of statements

- Set it big enough for working set of statements

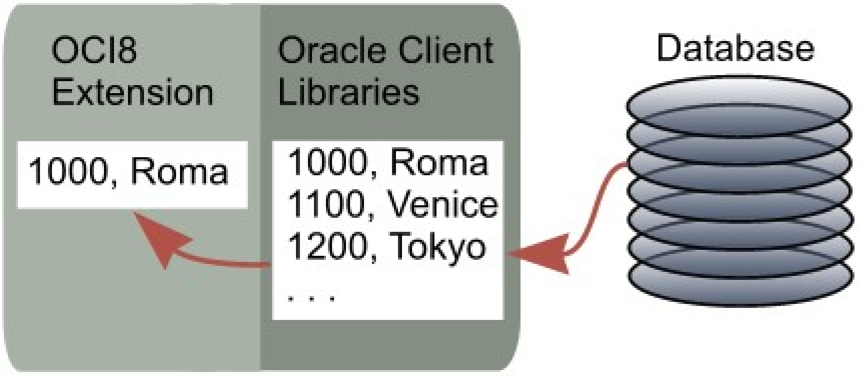


# Row Prefetching

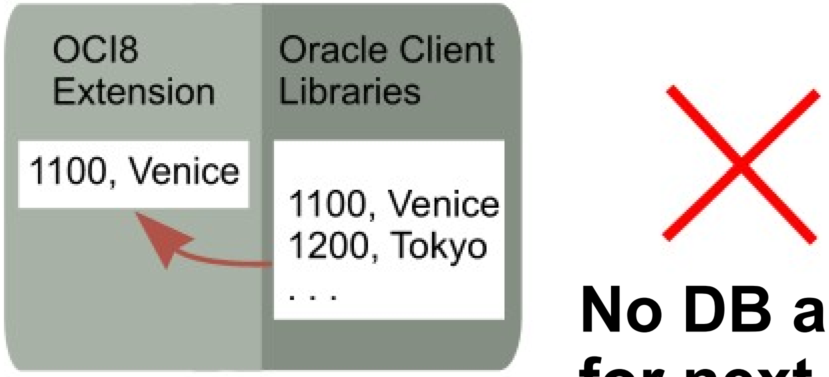
# Prefetching Reduces Roundtrips

Temporary buffer cache for query duration

```
$r = oci_fetch_array(...);
var_dump($r);
// array('1000', 'Roma')
```



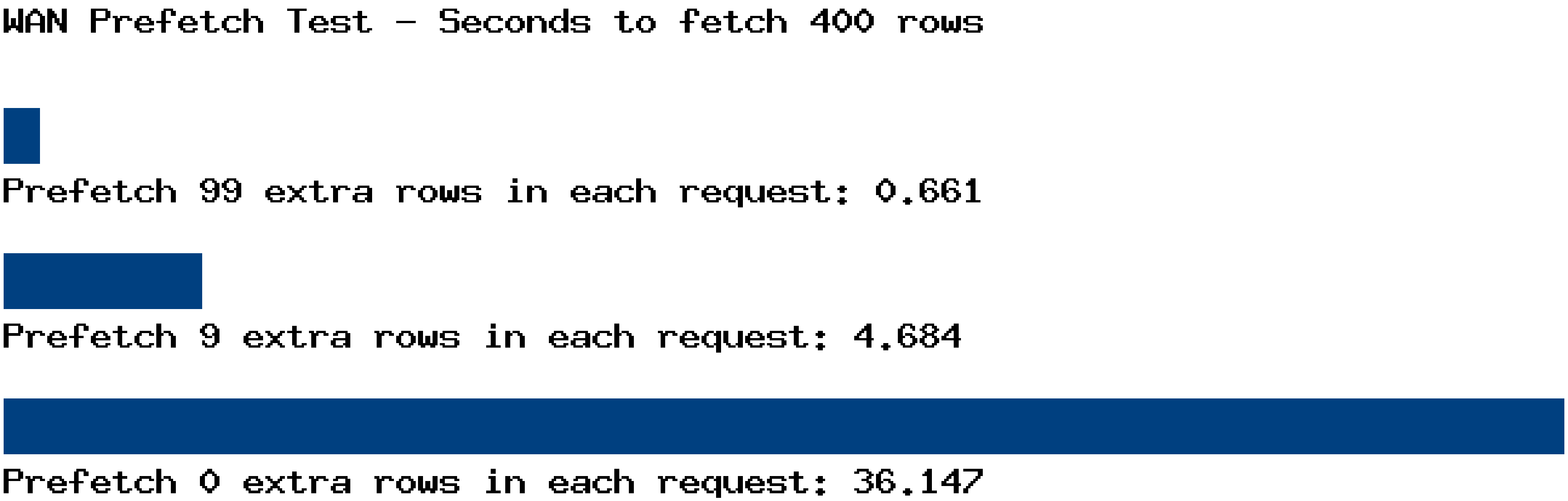
```
$r = oci_fetch_array(...);
var_dump($r);
// array('1100', 'Venice')
```



**No DB access for next fetch**

Reduces round trips

# Prefetching Reduces Query Times



Your results may vary

# Prefetching is Enabled by Default

- Enabled by default `oci8.default_prefetch = 100` rows
  - Was 10 rows in OCI8 1.2
- Tuning goal: Reduce round trips
  - but transfer reasonable chunks, not huge sets
- Can tune per statement:

```
$s = oci_parse($c, 'select city from locations');
oci_set_prefetch($s, 87);
oci_execute($s);
while (($row = oci_fetch_array($s, OCI_ASSOC)) != false)
  foreach ($row as $item)
    print $item;
```

# Menu

- Appetizer
- Connection Techniques
- Statement Techniques
- ►Caching Techniques
- Oracle 11gR2 and PHP OCI8 1.4

# Resultset Caching

# Oracle 11g Client & Server Result Caches

- Results of queries can be cached
  - Server and client (aka PHP) have caches
  - Recommended for small lookup tables
  - Client cache is per-process

- **Caches automatically invalidated by server data changes**

- Feature can be configured globally or per client
  - DB parameter: `CLIENT_RESULT_CACHE_SIZE`
    Per-client in sqlnet.ora: `OCI_RESULT_CACHE_MAX_SIZE`
  - Has a configurable 'lag' time
    - If no roundtrip within defined time, cache assumed stale

# No DB Access When Client Cache Used

- With Oracle 11gR1 use Query Annotation hint

```
select /*+ result_cache */ * from cjcrc
```

- V$RESULT_CACHE_* views show cache usage
- Test shows reduced DB access when client caching enabled:

```
SQL> select parse_calls, executions, sql_text
     from v$sql where sql_text like '%cjcrc%';

PARSE_CALLS EXECUTIONS SQL_TEXT
----------- ---------- -----------------------------------
          2        100 select * from cjcrc
          2          2 select /*+ result_cache */ * from cjcrc
```

# Result Caching Timing Test

```
$c = oci_pconnect('hr', 'hrpwd', 'localhost/orcl');
$tbls = array('locations', 'departments', 'countries');
foreach ($tbls as $t) {
    $s = oci_parse($c, "select /*+ result_cache */ * from $t");
    oci_execute($s, OCI_DEFAULT);
    while ($row = oci_fetch_array($s, OCI_ASSOC)) {
        foreach ($row as $item) {echo $item."\n";}}}
```

```
$ siege -c 20 -t 30S  http://localhost/clientcache.php
```

Without result cache:  select * from $t
Transaction rate: 32.02 trans/sec

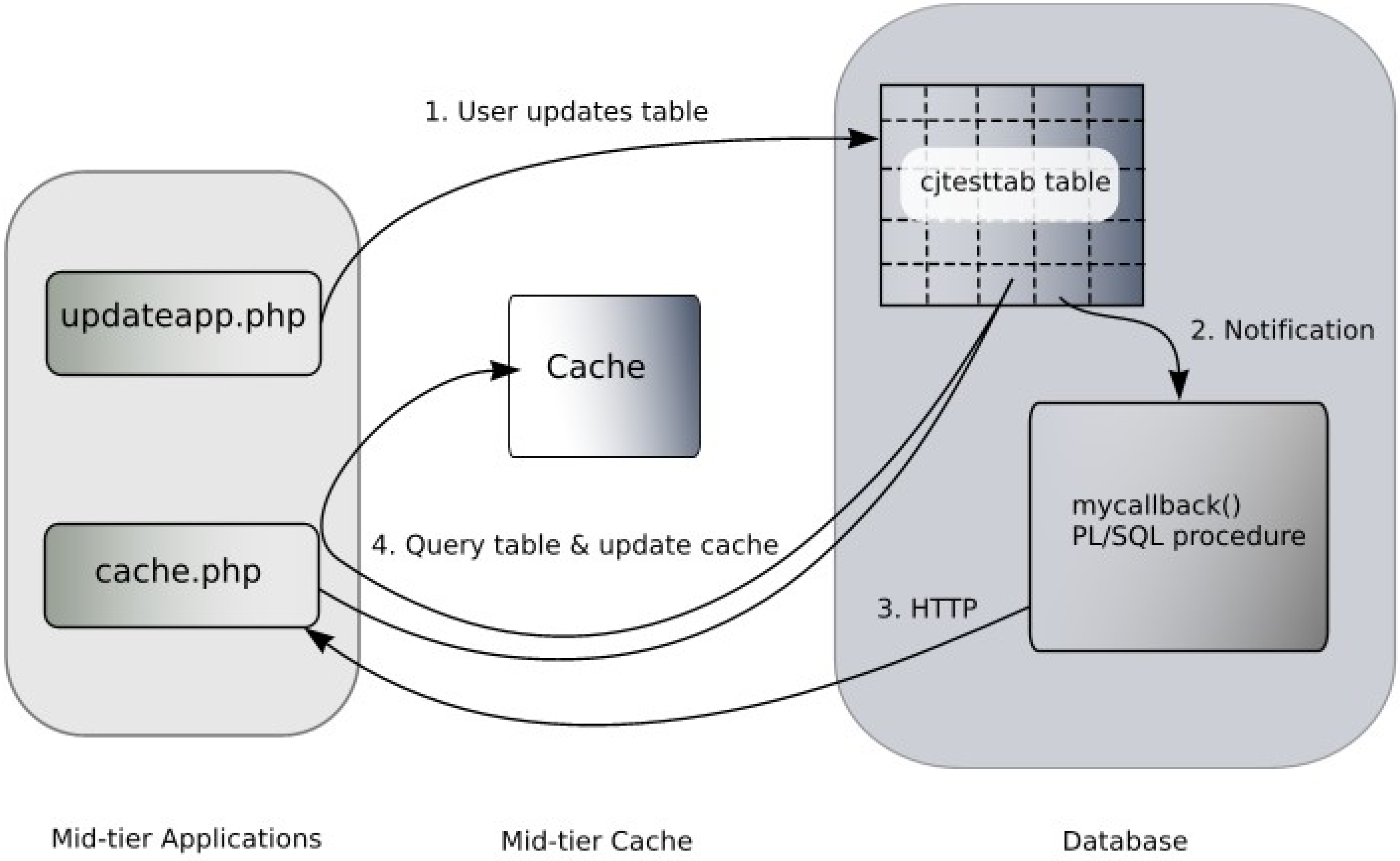
With result cache cache:  select /*+ result_cache */ * from $t
Transaction rate: 36.79 trans/sec

Result Caching was approx. 15% better

# Mid-tier Cache Invalidation by Continuous Query Notification

# Continuous Query Notification Example

Problem: update cache only when the resultset for a query changes



# Example: Cache Depends On A Table

```
$ sqlplus cj/cj
create table cjtesttab (
  group_id number,
  name     varchar2(20)
);
insert into cjtesttab values (1, 'alison');
insert into cjtesttab values (2, 'pearly');
insert into cjtesttab values (2, 'wenji');
```

# The PHP Cache-Updating Code

```
// cqn.php
. . .
$tab = my_filter($_GET[tabname);
$s = oci_parse($c, "select * from ".$tab);
oci_execute($s);
oci_fetch_all($s, $data);
$memcache->set('key', serialize($data));
```

# Create 'mycallback' PL/SQL Procedure

```
create or replace procedure mycallback (
          ntfnds in cq_notification$_descriptor) is
  req  utl_http.req;
  resp utl_http.resp;
begin
  if (ntfnds.event_type = dbms_cq_notification.event_querychange) then
    req := utl_http.begin_request(
         'http://mycomp.us.oracle.com/~cjones/cqn.php&tabname='
         || ntfnds.query_desc_array(1).table_desc_array(1).table_name);
    resp := utl_http.get_response(req);
    utl_http.end_response(resp);
  end if;
end;
/
```

# Register 'mycallback' for a Query

```
declare
  reginfo   cq_notification$_reg_info;
  v_cursor  sys_refcursor;
  regid     number;
begin
  reginfo := cq_notification$_reg_info (
    'mycallback',                     -- PL/SQL callback function
    dbms_cq_notification.qos_query,   -- result-set notification flag
    0, 0, 0);
  regid := dbms_cq_notification.new_reg_start(reginfo);
    open v_cursor for select name from cjtesttab where group_id = 2;
    close v_cursor;
  dbms_cq_notification.reg_end;
end;
/
```

# Example Recap

- Table `cjtesttab`
- PHP script `http://.../cqn.php` to update the cache
- PL/SQL callback procedure `mycallback()`
- Registered query

```
select name from cjtesttab where group_id = 2;
```

  - Aim: refresh mid-tier cache when the query results change

# Example – Continuous Query In Action

- Update the table (aiming at changing the result set):

  ```
  update cjtesttab set name = 'c' where group_id = 2;
  commit;
  ```

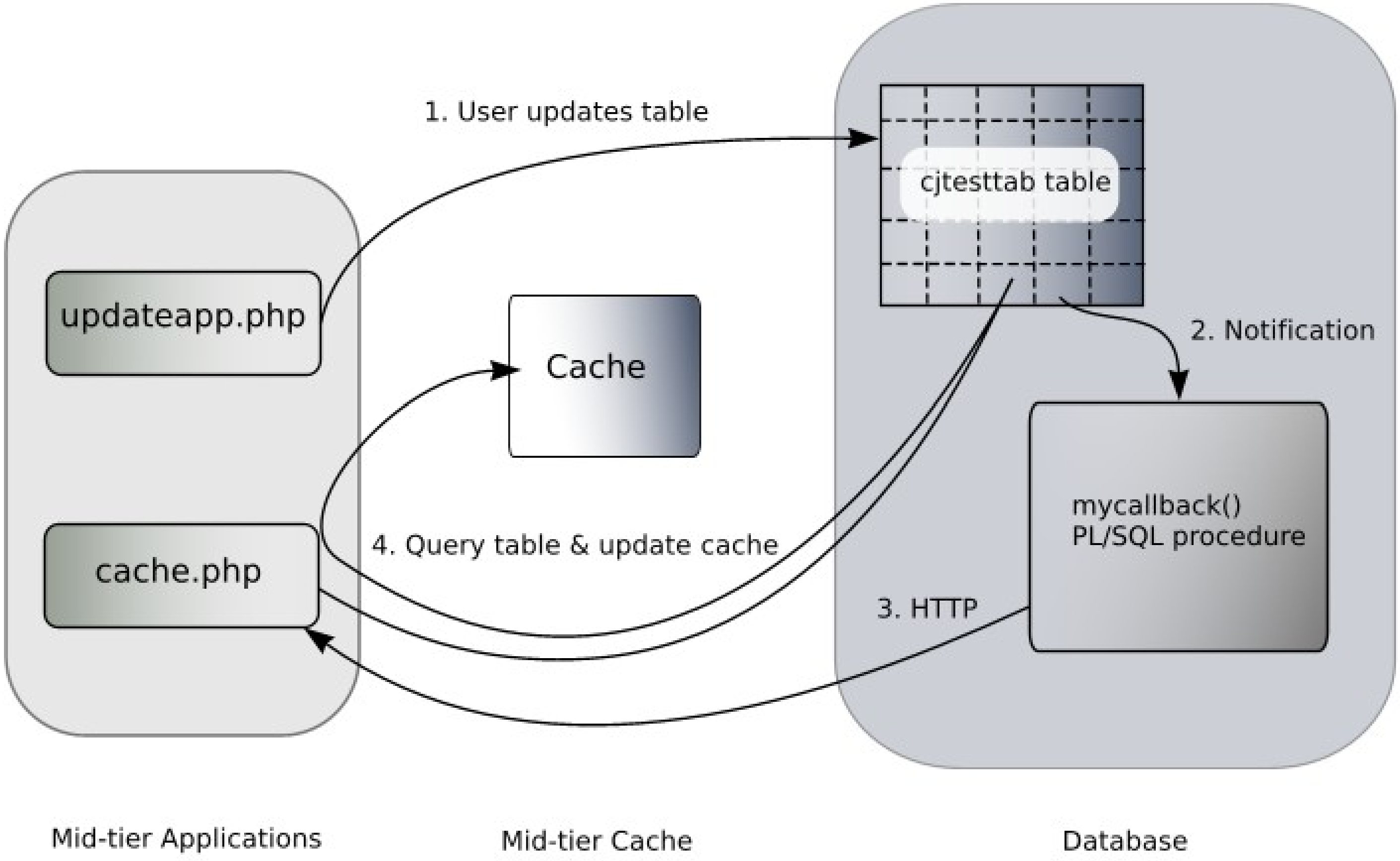
  - Result: Cache is refreshed

- Update a different **`group_id`** (aiming at not changing the result set):

  ```
  update cjtesttab set name = 'x' where group_id = 1;
  commit;
  ```

  - No notification is generated

# Continuous Query Notification Example



# Menu

- Appetizer
- Connection Techniques
- Statement Techniques
- Caching Techniques
- ►Oracle 11gR2 and PHP OCI8 1.4

# Oracle Database 11gR2

- Oracle 11gR2 was released September 2009
- Many new features building on 11gR1 innovations
  - Pre-fetching supported in more places
  - Continuous Query Notification for Views
  - Client Query Result Cache (CQRC) now has table and view annotation
  - CQRC supported with DRCP
  - Improved views for DRCP connection pooling
- Edition Based Redefinition
- RAC One Node option
- . . .

# Oracle 11gR2 Client & Server Result Caches

- In Oracle 11gR1, developer adds hint to table query:

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

- In Oracle 11gR2 DBA can choose tables or view to be cached:

```
create table sales (...) result_cache

alter table last_name result_cache

create view v2 as
   select /*+ result cache */ col1, coln from t1
```

No need to change PHP application

ORACLE

# REF CURSOR Prefetching

- New with Oracle 11.2
  - Works with 11.2 client libraries to older DBs
- Enabled by default or set explicitly

# REF CURSOR Prefetching

```
/* create or replace
procedure myproc(p1 out sys_refcursor) as
begin
  open p1 for select * from tab;
end; */

$s = oci_parse($c, "begin myproc(:rc); end;");
$rc = oci_new_cursor($c);
oci_bind_by_name($s, ':rc', $rc, -1, OCI_B_CURSOR);
oci_execute($s);
oci_set_prefetch($rc, 200);
oci_execute($rc);
oci_fetch_all($rc, $res);
```

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.

ORACLE

# OCI8 1.4 Overview

- PECL OCI8 1.4 Alpha released on October 6
  - Same code was merged to PHP 5.3 & 6
  - Will **not** be in PHP 5.3.1
- OC8 1.4 builds and runs with Oracle 9.2 - 11.2
  - OCI8 1.3 builds and runs with Oracle 9.2 - 11.2
- Has Oracle “Attribute” support
  - Improves tracing, logging, auditing and user security
- Some bug fixes

# OCI8 1.4: Tracing and Monitoring

- Driver type is always set to “PHP OCI8”
  - needs 11.2 client libraries
  - DBA can see in V$SESSION_CONNECT_INFO
- New functions

```
oci_set_client_info($c, “My App”)
oci_set_module_name($c, “Home Page”)
oci_set_action($c, “Friend Lookup”)
```

  - Oracle 10g+
  - User chosen strings
  - Takes effect on next “roundtrip”
  - See in various views e.g. V$SQLAREA, V$SESSION
  - Not reset for persistent connections at script end

# OCI8 1.4: Client Identifier

- New function:

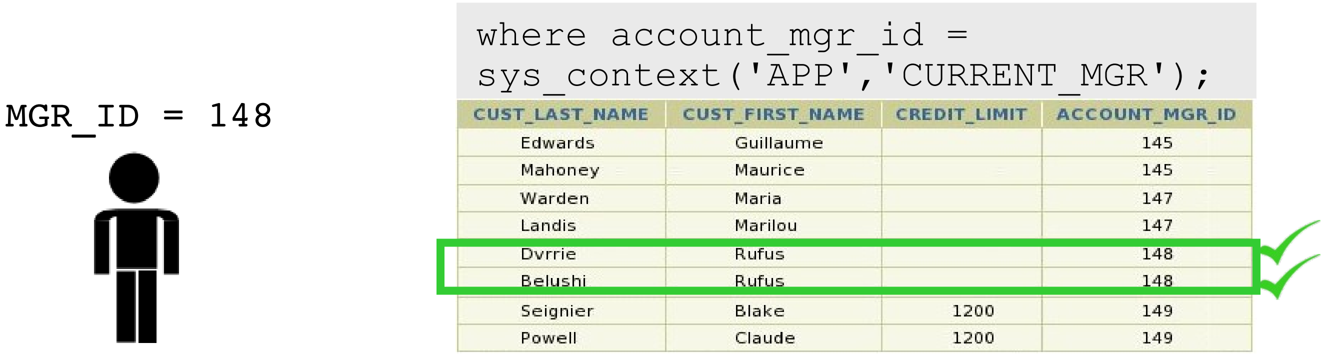
  ```
  oci_set_client_identifer($c, “ABC”)
  ```

  - Oracle 9.2+
  - Takes effect on next “roundtrip”
  - See in various views e.g. V$SESSION
  - Not reset for persistent connections at script end

```
session_start();
$appuser = my_get_app_user($_SESSION['username']);
$c = oci_pconnect('myuser', 'mypasswd, 'MYDB');
oci_set_client_identifer($c, $appuser)
...
```

# Client Identifier and Virtual Private DB

- Client Identifier
  - Propagate middle tier user identity to the backend database
  - Value can be used in access control decisions
  - Value is recorded in audit trail
  - Used with DBMS_MONITOR.CLIENT_ID_TRACE_ENABLE
- Used for Virtual Private Databases
  - Row Level Security: DBMS_RLS package
  - Uses Application Context i.e., WHERE Clause dynamically generated



MGR_ID = 148

```
where account_mgr_id =
sys_context('APP','CURRENT_MGR');
```

| CUST_LAST_NAME | CUST_FIRST_NAME | CREDIT_LIMIT | ACCOUNT_MGR_ID |
|---|---|---|---|
| Edwards | Guillaume | | 145 |
| Mahoney | Maurice | | 145 |
| Warden | Maria | | 147 |
| Landis | Marilou | | 147 |
| Dvrrie | Rufus | | 148 |
| Belushi | Rufus | | 148 |
| Seignier | Blake | 1200 | 149 |
| Powell | Claude | 1200 | 149 |

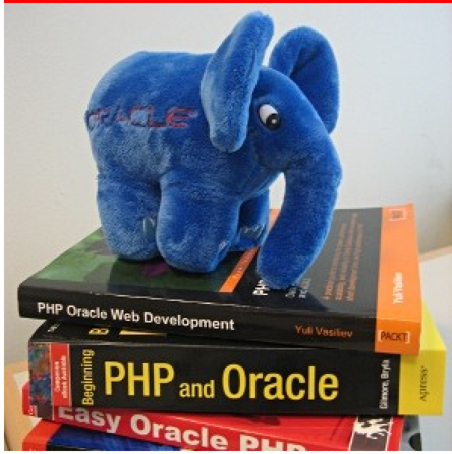
# OCI8 1.4: Extras

- New OCI_NO_AUTO_COMMIT flag
  - Alias of OCI_DEFAULT (which isn't the default option)

```
oci_execute($s, OCI_NO_AUTO_COMMIT)
```

- Allow row prefetch value to be 0
  - Avoids row “loss” if passing REF CURSORS back and forth between PHP and PL/SQL
- New oci_set_edition() procedure

# Upgrading Live Sites

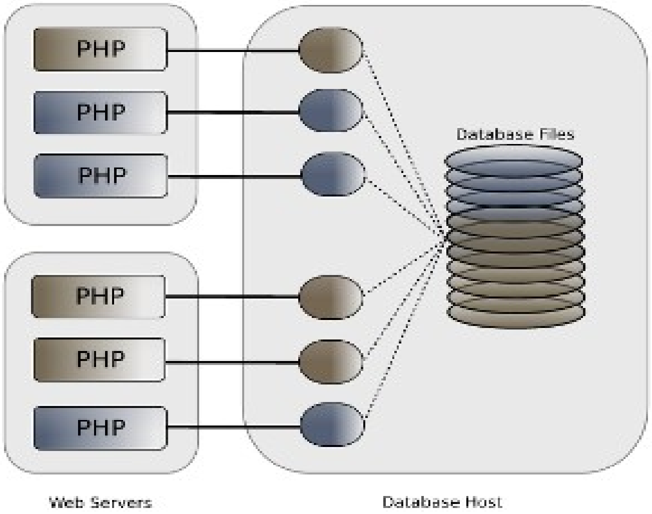


# Edition Based Redefinition

- *EBR allows application upgrade and testing **while still in production use***
- Oracle 11gR introduces EBR and the:
  - Edition
  - Editioning view
  - Crossedition trigger
- EBR allows multiple versions of DB objects to be used concurrently

# EBR and PHP Applications

- Use EBR in conjunction with PHP code changes
  - Load balancer/web server needs to call correct version of PHP scripts
- Use EBR for
  - A/B testing
  - Application migration and release testing



# Application Version 1: DB Objects

**Requirement:** Show number of days vacation for employees

```
SQL> create table myemp (name varchar2(10),
                         hoursworked number);
SQL> insert into myemp values ('alison', 200);

SQL> create function vacationdaysleft(p_name in varchar2)
return number as
  vdl number;
begin
  select floor(hoursworked / 40) into vdl
    from myemp where name = p_name;
  return vdl;
end;
```

# Application Version 1: PHP Code

```
// vacation1.php

oci_set_edition('ora$base');  // OCI8 1.4
$c = oci_connect('cj','welcome','localhost/orcl');

$s = oci_parse($c,
       “begin :vdl := vacationdaysleft('alison'); end;”);
oci_bind_by_name($s, “:vdl”, $vdl, 15);
oci_execute($s);
echo “alison has “ . $vdl . ” days vacation left\n”;
```

```
$ php vacation1.php
alison has 5 days vacation left
```

# EBR Recap: Application Version 1

- One **`vacation1.php`** script
- One **`vacationdaysleft`** stored function
- One **`myemp`** table
- One set of users accessing the application

# Application Version 2: DB Schema

New Requirement: Alter the vacation rate calculation

```
SQL> create edition e2;
SQL> alter session set edition = e2;

SQL> create function vacationdaysleft(p_name in varchar2)
   return number as
   vdl number;
 begin
  select floor(hoursworked / 20) into vdl
    from myemp where name = p_name;
  return vdl;
end;
```

# Application Version 2: PHP Code

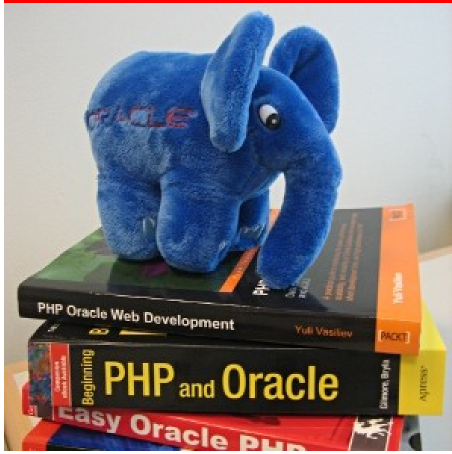
```
// vacation2.php

oci_set_edition('e2');  // OCI8 1.4
$c = oci_connect('cj','welcome','localhost/orcl');

$s = oci_parse($c,
       “begin :vdl := vacationdaysleft('alison'); end;”);
oci_bind_by_name($s, “:vdl”, $vdl, 15);
oci_execute($s);
echo “alison now has “ . $vdl . ” days vacation left\n”;
```

# Recap: Edition Based Redefinition

- Two scripts: **vacation1.php** and **vacation2.php**
- Two **vacationdaysleft** stored functions in same schema
- One **myemp** table
- Two sets of web users running different versions **concurrently**

```
$ php vacation1.php
alison has 5 days vacation left
$ php vacation2.php
alison now has 10 days vacation left
```

- When migration completed, use `DROP EDITION`
- Use `oci_set_edition()` not `ALTER SESSION` in PHP



# Oracle Database and PHP Roadmap

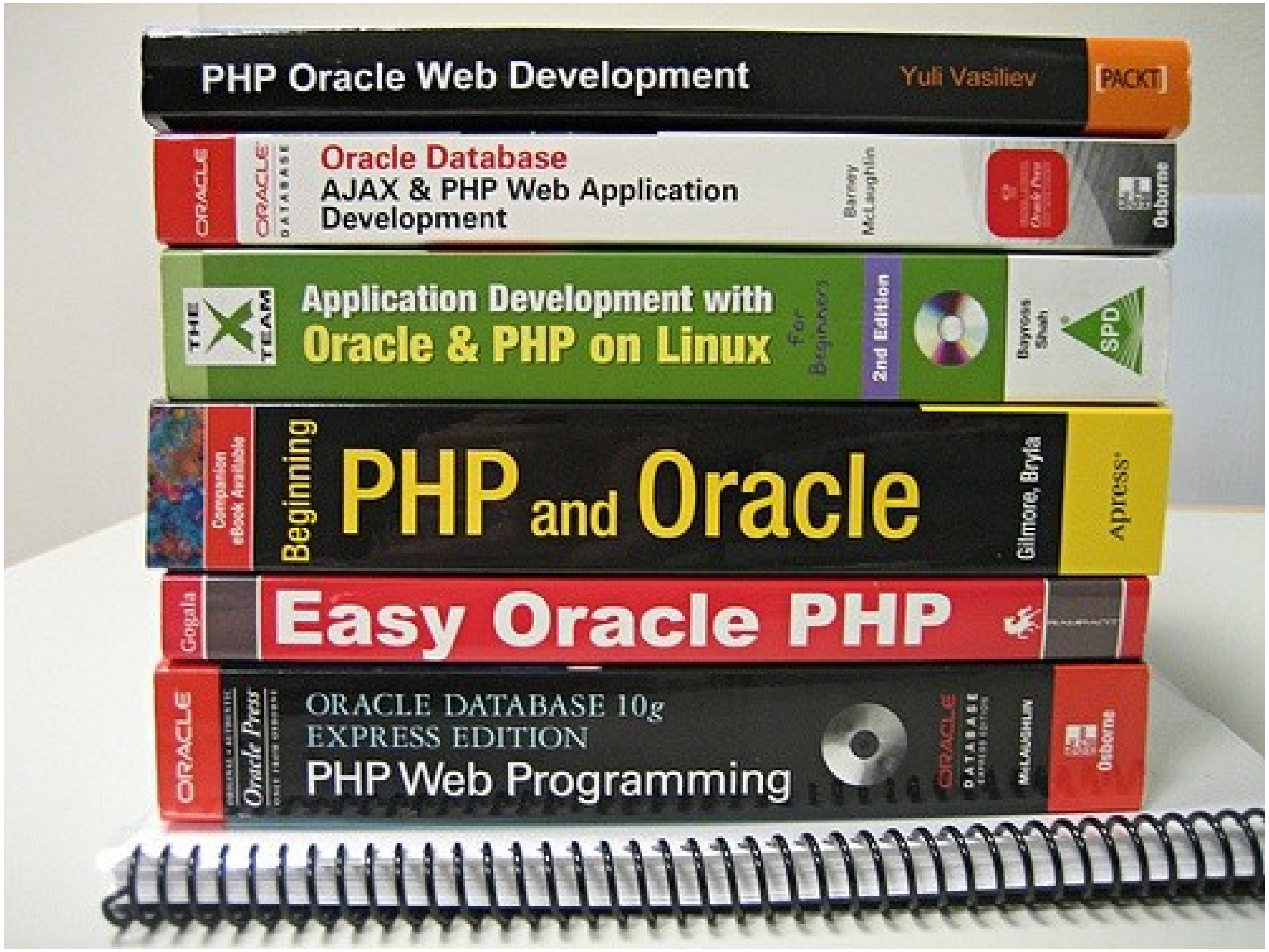
# Oracle Database and PHP Roadmap

- PDO_OCI
  - There is a disconnect between PDO_xxx users and maintainers
  - PDO_xxx drivers are not well maintained

# Oracle Database and PHP Roadmap

- PHP OCI8 integration with
  - TimesTen In Memory Database
    - A fast in memory, persistent DB
  - TimesTen In Memory Database Cache
    - Cache for Oracle Database
  - No need for separate caching logic

# Some PHP & Oracle Books



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.

# Oracle Resources

Oracle Techology Network (OTN)

- PHP Developer Center: **otn.oracle.com/php**
- Free book: *Underground PHP and Oracle Manual*
- Whitepapers, Articles, FAQs, links to blogs
- PHP Extension, PHP RPMs

Information

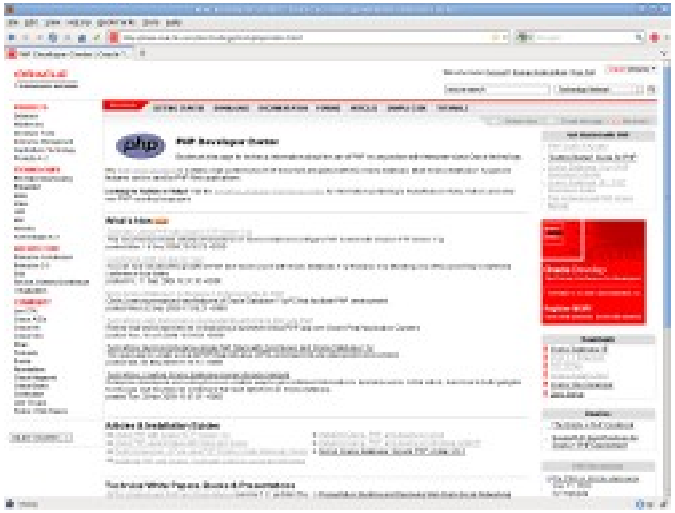
- christopher.jones@oracle.com
- blogs.oracle.com/opal
- kuassi.mensah@oracle.com
- db360.blogspot.com

SQL and PL/SQL Questions

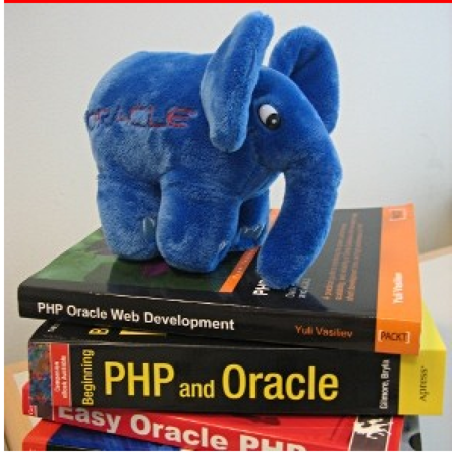
- asktom.oracle.com

ISVs and hardware vendors

- oraclepartnernetwork.oracle.com



ORACLE®



# Extra Slides

# DBMS_XA: Transactions Across Requests

- Oracle 11gR1 Feature
  - Can we use it on the web? Upgrading thick client applications?
- Example from http://tinyurl.com/dbmsxaex

HTTP Request #1:

```
rc  := DBMS_XA.XA_START(DBMS_XA_XID(123), DBMS_XA.TMNOFLAGS);
UPDATE employees SET salary=salary*1.1 WHERE employee_id = 100;
rc  := DBMS_XA.XA_END(DBMS_XA_XID(123), DBMS_XA.TMSUSPEND);
```

HTTP Request #2:

```
rc  := DBMS_XA.XA_START(DBMS_XA_XID(123), DBMS_XA.TMRESUME);
SELECT salary INTO s FROM employees WHERE employee_id = 100;
rc  := DBMS_XA.XA_END(DBMS_XA_XID(123), DBMS_XA.TMSUCCESS);
```

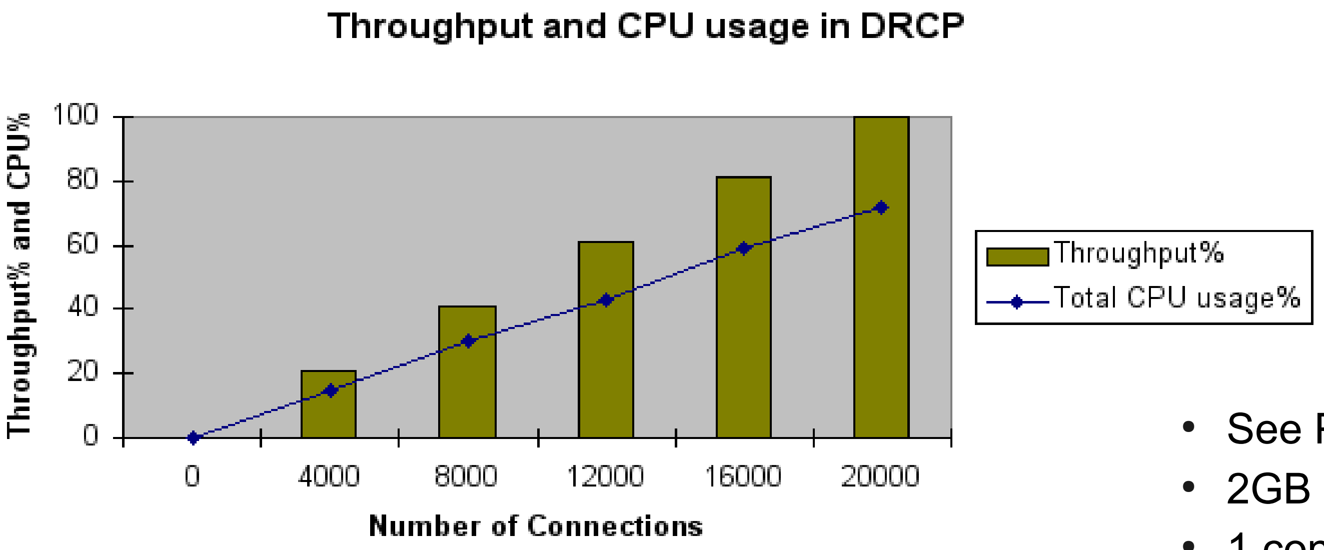
HTTP Request #3:

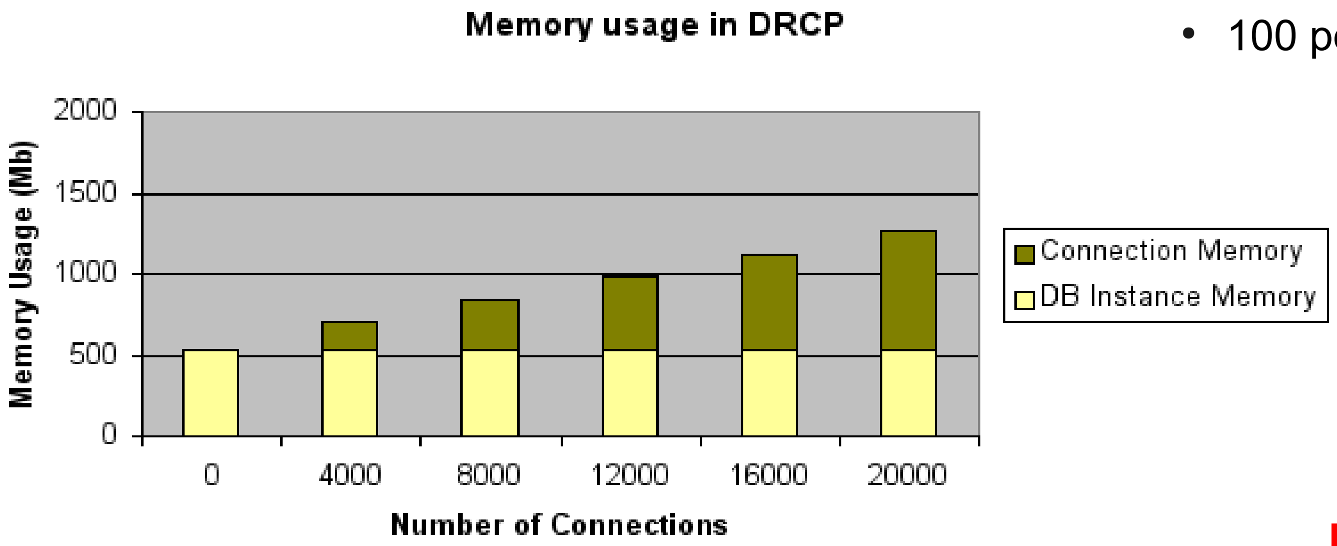
```
rc  := DBMS_XA.XA_COMMIT(DBMS_XA_XID(123), TRUE);
```

# Hey Look! Free Stuff

- Oracle Instant Client
  - Easy to install
  - Client libraries for C, C++, Java, .Net access to a remote DB
- Oracle Database Express Edition (aka “XE”)
  - Same code base as full database
  - Windows & Linux 32 bit
- SQL Developer
  - Thick client SQL and PL/SQL development tool
  - Connects to MySQL too
- Application Express (“Apex”)
  - Web based Application Development tool
  - Try it at http://apex.oracle.com/

# PHP DRCP Benchmark





- See PHP DRCP Whitepaper
- 2GB RAM
- 1 connection broker
- 100 pooled servers