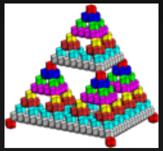
TESTING ORACLE

QA Strategies for Success

Anthony D. Noriega
MSCS, MBA, BSSE, OCP-DBA

Speaker Qualifications



- Over 20 years of IT experience, 18 with Oracle
- Former OCP and Oracle University Instructor.
- Speaker at NYOUG and IOUG events
- MS Computer Science, NJIT, 1993
- PhD CIS candidate, NJIT, 1997 with software engineering research under the supervision of Profs. Dr. Wilhelm Rossak (Technische Universität Wien, TU Vienna) and Dr. David T. Wang (CMU)
- MBA MIS, Montclair State University, 2006
- BS Systems Eng., Universidad del Norte, Colombia, 1987 Consultant at Allied-Signal Aerospace Company’s CAE Center, AT&T, Bowne & Co, Deutsche Bank, Empire Blue-Cross Blue-Shield, FMC, IBM, MCS Canon, Merrill-Lynch, M&M Mars, TD Ameritrade, and Time Warner.

Objectives

- Recount 18 years of Oracle analysis, design and implementation, and software engineering experience
- Emphasize the workflow characterization in order to accomplish better QA control.
- Discuss areas of current and future applications to achieve process improvement.
- Emphasize and inter-related both the related conceptual and practical frameworks.
- Introduce Oracle11g Real Application Testing (RAT)

Conceptual Framework Overview

- Software Engineering Model
  - Waterfall Model
  - Incremental and Iterative Models
  - Spiral Model
- CMM/CMMI CMM:
  - Predictability of recurring success based on maturity.
  - CMMI-Dev focuses on specific areas such as risk, configuration management, high availability, reliability, etc.
- Six Sigma
  - Define, Measure, Analyze, Improve, Control (DMAIC).
- XP Model
  - Unit Testing.

Conceptual Framework Overview

- Prototyping Methodologies
  - Rapid
  - Incremental
  - Exploratory
- Rapid Application Development (RAD) Methodologies
- Rational Unified Process (RUP)
  - Object-Oriented Analysis and Design (UML)
  - Use Cases
- Reengineering Model

Testing Paradigms

- Black Box Testing
- White Box Testing
- Gray Box Testing
- Unit Testing (Extreme Programming, XP)
- Software Refactoring (Regression Testing)
- SADT (Software Analysis and Design Technique)
- Quality Improvement Process (QIP)
- IDEAL.

Software Metrics

- Production Cost Models
  - Construction Cost Model (COCOMO)
  - COCOMO II
  - Based on KLOC or KDSI model
- Commercial off-the-shelf (COTS)
  - Component reusability
  - Pre-built, cheaper and faster (hot pluggable).

Quality Assurance Models

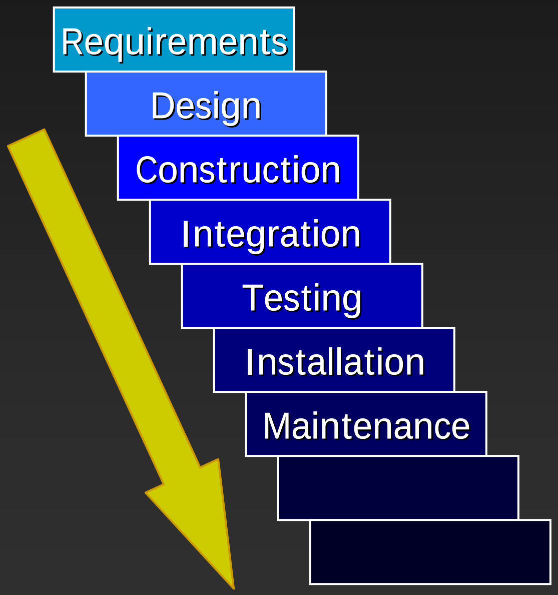
- Total Quality Management (TQM)
- Quality Improvement Process (QIP)

Quality Assurance Stages

- Requirements Engineering
- Design and Implementation
- Development
- System Integration
- User Acceptance Testing (UAT)
- Regression Testing
- Maintenance and Production Control

Empiric Business Framework

The WaterFall Model



Requirements

Design

Construction

Integration

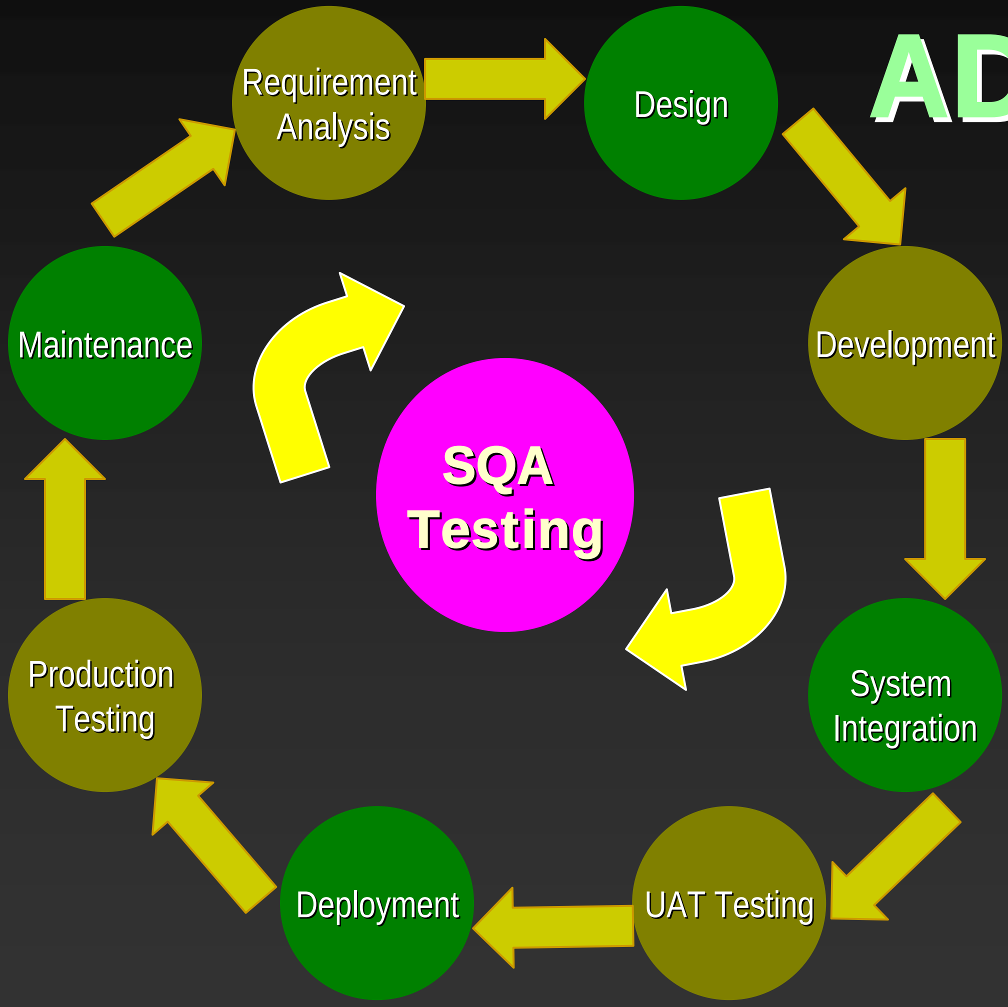
Testing

Installation

Maintenance

SDLC

ADN



Requirement Analysis

Design

Development

System Integration

UAT Testing

Deployment

Production Testing

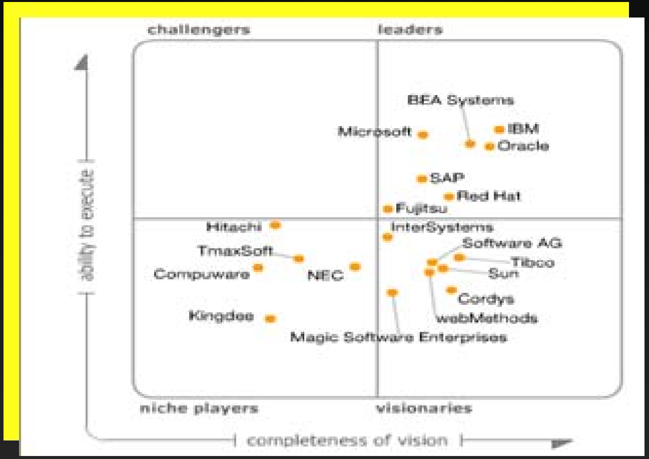
Maintenance

SQA Testing

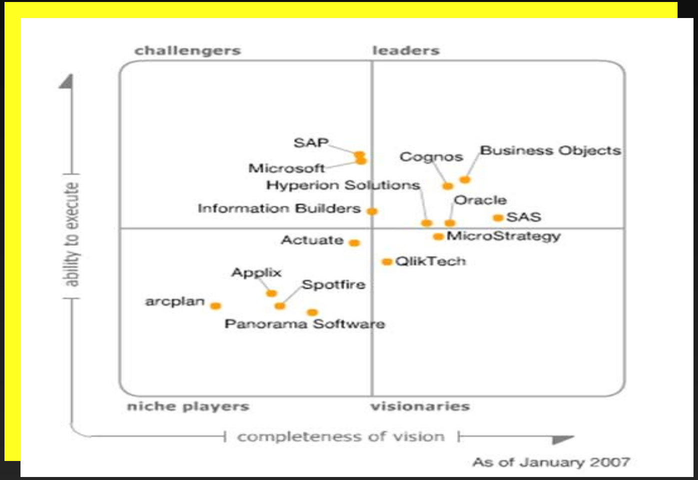
Workflow Characterization

- Relating Database Workflow characterization is finding the cause of resource utilization and related patterns.
- Managing and controlling workflow accordingly is not a task reserved solely to the DBA or the System Administrator.
- Identifying workflow source such as being driven by a Front-end, Back-end, Middleware or Network Entity is an important finding to achieve successful testing and quality assurance in a related database environment.
- Failing to correctly identify and characterize the environment workflow could result in the inadequacy of the master QA plan, and systematic error at all stages of the SDLC or other significant iterative methodology used.

Leadership Counts



Application Server Market

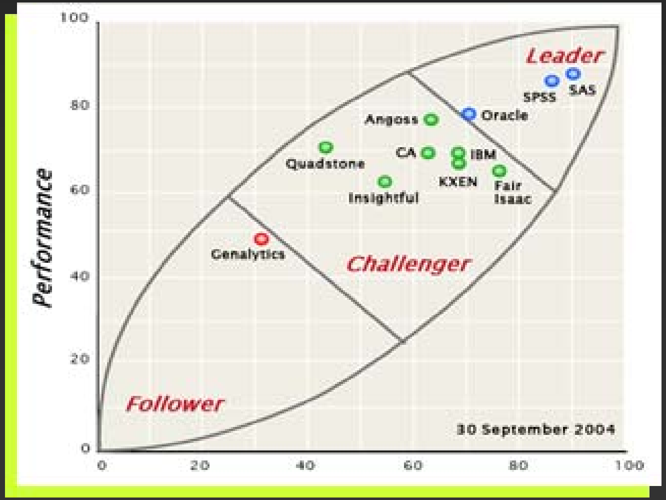


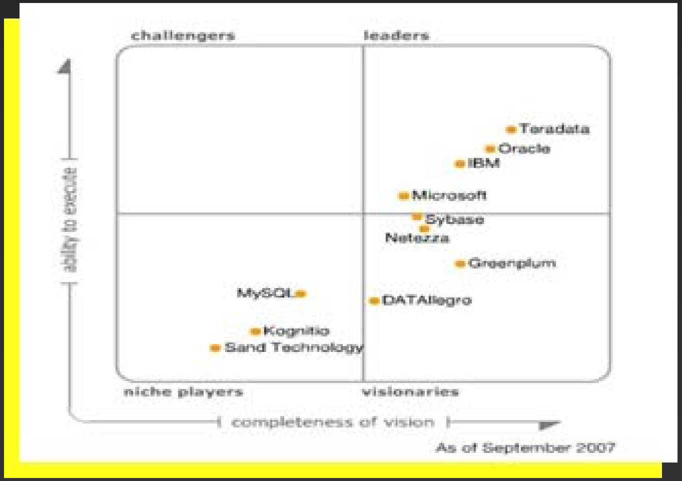
Business Intelligence Market

Oracle in the

Leader’s

Quadrant





Data Mining Market

Database Market

Consulting Case Studies

- **Requirements Engineering**
  - **From Functional to Technical Requirements.**
  - **Setting the Baseline**

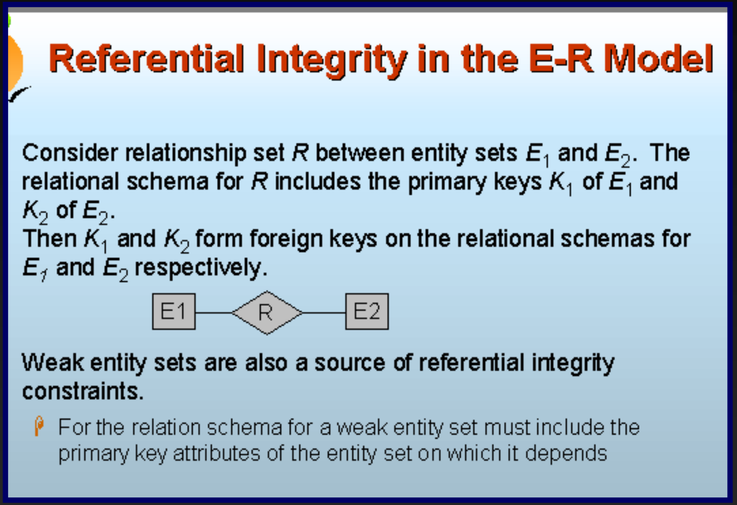
Performing elicitation, gathering and maintaining consistent and up-to-date requirements at all phases is of major concern to attain optimal quality assurance at each stage.

Consulting Case Studies

- Data Model Design and Implementation
  - Data Model tuning could involve:
    - Further normalization or de-normalization
    - Further model decomposition
    - Physical model tuning: indexes, MVs, etc.
- Business-driven tuning or enhancement, e.g., globalization, internationalization, or localization of an existing data model.

Ensuring the data integrity and consistency fully comply with requirements engineering goals, even in scenarios where testing is made via snapshot from production databases.

Testing a Simple Relationship

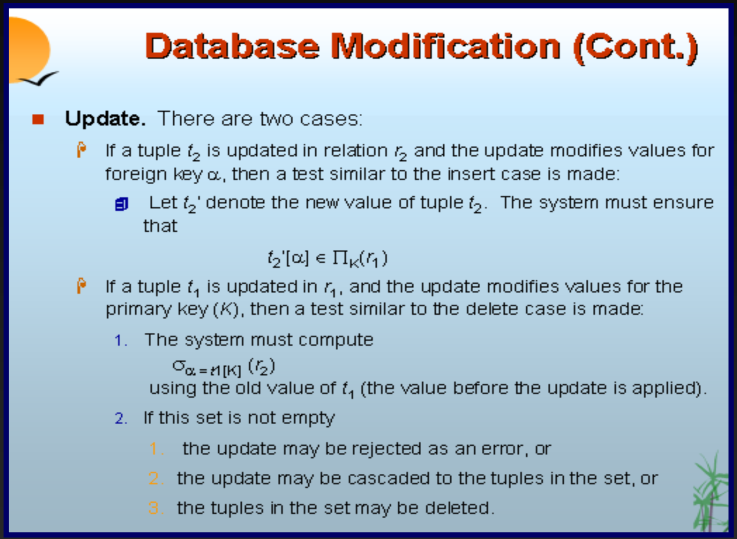


Referential Integrity in the E-R Model

Consider relationship set R between entity sets E_1 and E_2. The relational schema for R includes the primary keys K_1 of E_1 and K_2 of E_2.
Then K_1 and K_2 form foreign keys on the relational schemas for E_1 and E_2 respectively.

Weak entity sets are also a source of referential integrity constraints.

- For the relation schema for a weak entity set must include the primary key attributes of the entity set on which it depends

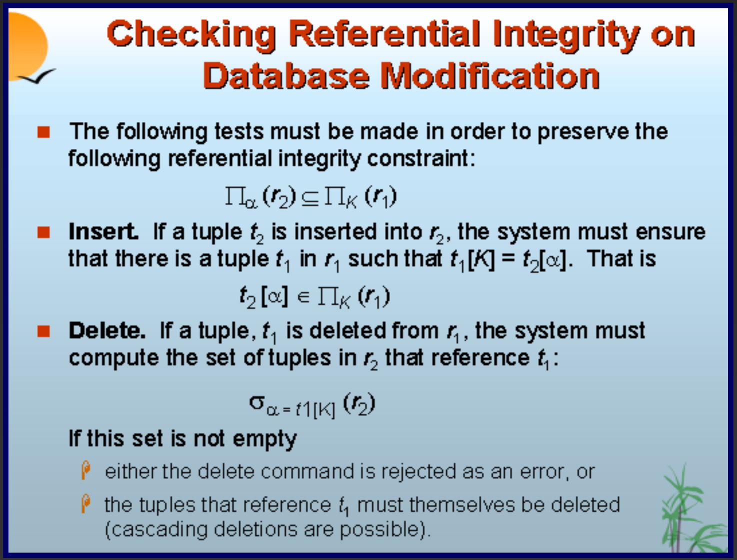


Database Modification (Cont.)

- **Update.** There are two cases:
  - If a tuple t_2 is updated in relation r_2 and the update modifies values for foreign key α, then a test similar to the insert case is made:
    - Let t_2' denote the new value of tuple t_2. The system must ensure that

$$t_2'[\alpha] \in \Pi_K(r_1)$$

  - If a tuple t_1 is updated in r_1, and the update modifies values for the primary key (K), then a test similar to the delete case is made:
    1. The system must compute
       $\sigma_{\alpha = t1[K]} (r_2)$
       using the old value of t_1 (the value before the update is applied).
    2. If this set is not empty
       1. the update may be rejected as an error, or
       2. the update may be cascaded to the tuples in the set, or
       3. the tuples in the set may be deleted.



Checking Referential Integrity on Database Modification

- The following tests must be made in order to preserve the following referential integrity constraint:

$$\Pi_\alpha (r_2) \subseteq \Pi_K (r_1)$$

- **Insert.** If a tuple t_2 is inserted into r_2, the system must ensure that there is a tuple t_1 in r_1 such that $t_1[K] = t_2[\alpha]$. That is

$$t_2 [\alpha] \in \Pi_K (r_1)$$

- **Delete.** If a tuple, t_1 is deleted from r_1, the system must compute the set of tuples in r_2 that reference t_1:

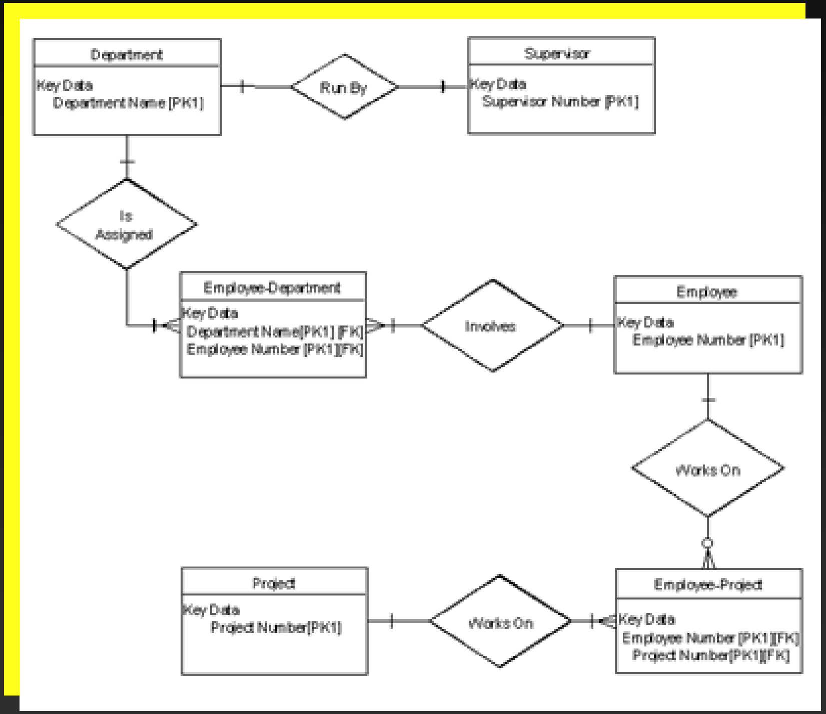
$$\sigma_{\alpha = t1[K]} (r_2)$$

  If this set is not empty
  - either the delete command is rejected as an error, or
  - the tuples that reference t_1 must themselves be deleted (cascading deletions are possible).

Testing a simple relationship is foundational to a successful transactional process. A master-detail relationship is another scenario that could be simplified on good integrity practices.

Consulting Case Studies

Art by Fondini



Source: http://www.infocom.cqu.edu.au/Courses/spr2000/95169/Extra_Examples/ERD.htm

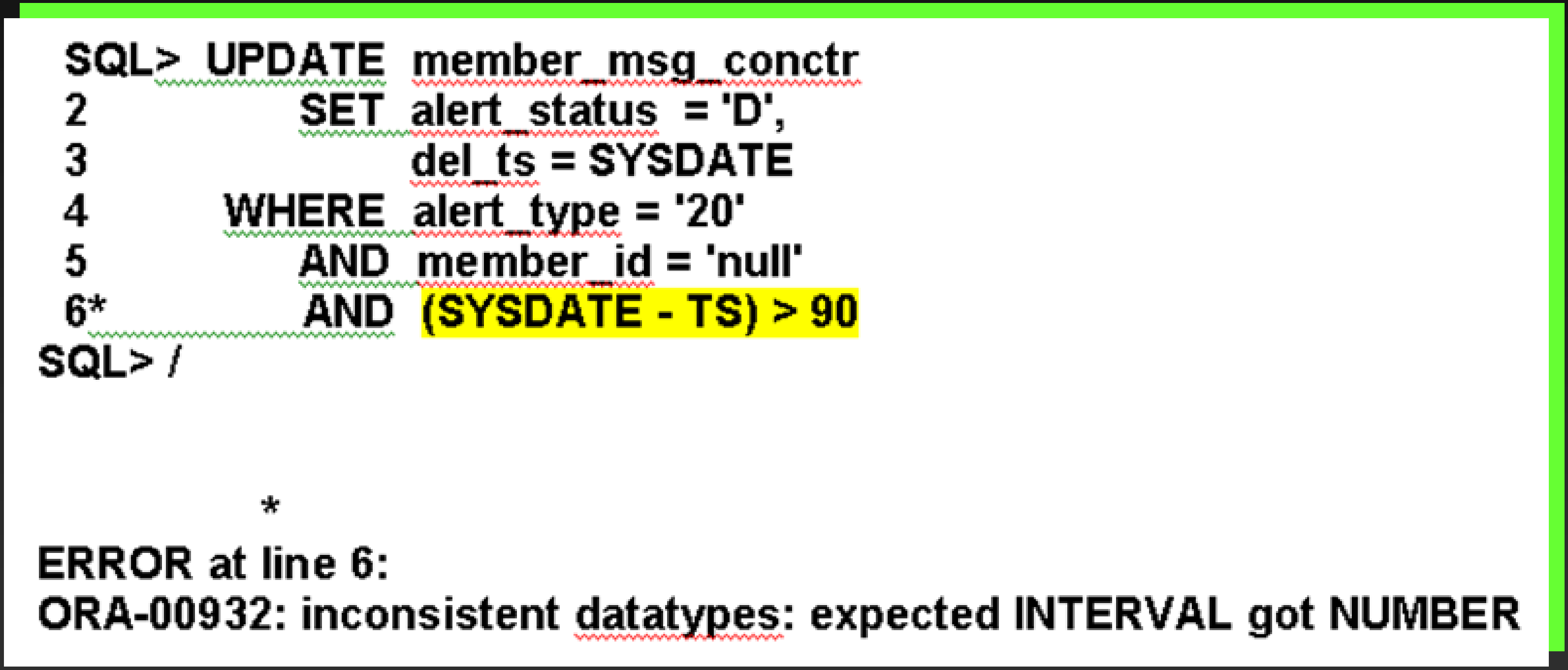
If your original data model is not consistent, your test could go upside down in most stages…or be rather surreal…

Consulting Case Studies

- Development
  - Testing SQL Code
    - Testing for Performance Tuning
    - Testing for Upgrade/Migration
    - Testing for Cloning Tuning
  - Strategies to Test PL/SQL

SQL Testing could be as easy as 123 or cumbersome as having to create a tuning SQL profile or outline after a careful study. PL/SQL extends its programming test mode and its SQL embedding capability thereon.

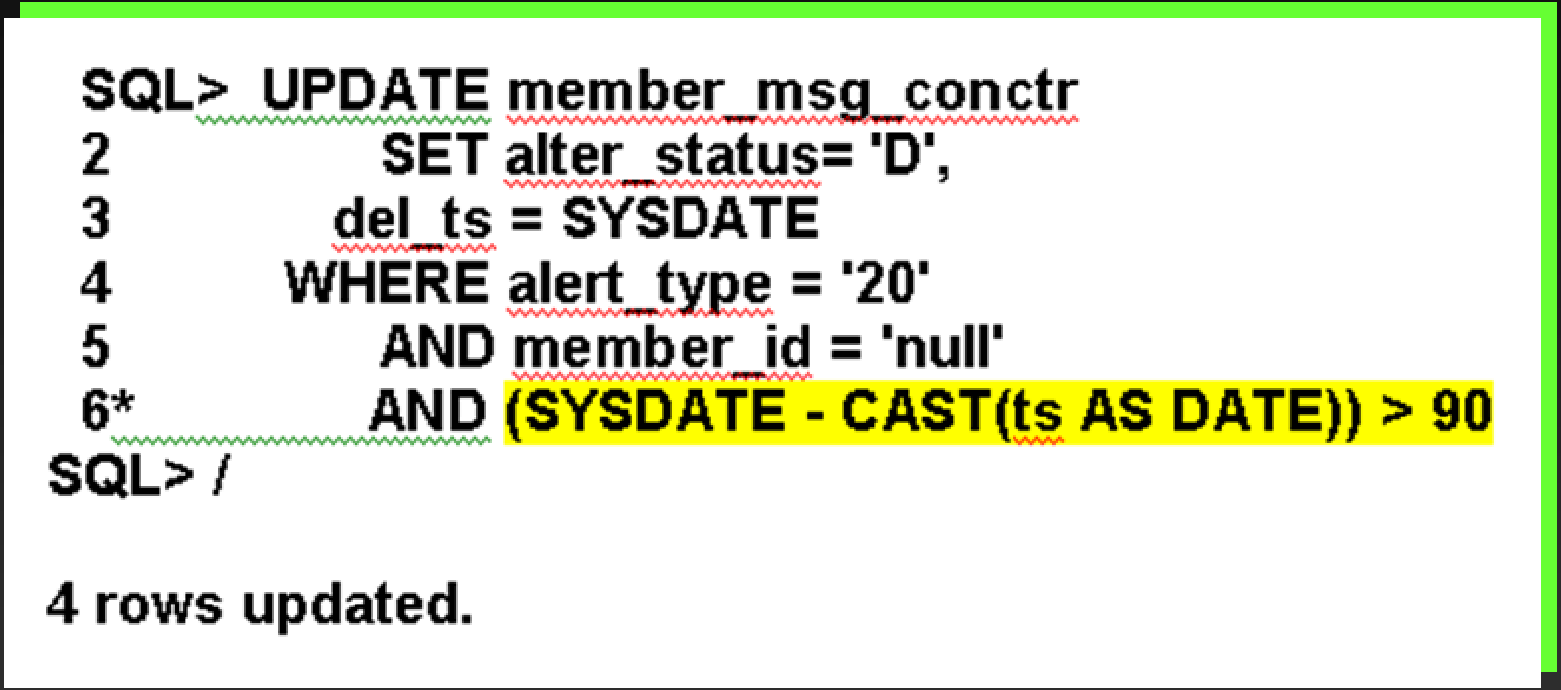
Sample SQL Code (Oracle9i) (8i)



```
  SQL>  UPDATE  member_msg_conctr
  2             SET  alert_status  = 'D',
  3                  del_ts = SYSDATE
  4          WHERE  alert_type = '20'
  5             AND  member_id = 'null'
  6*            AND  (SYSDATE - TS) > 90
SQL> /



          *
ERROR at line 6:
ORA-00932: inconsistent datatypes: expected INTERVAL got NUMBER
```

SQL code prior to upgrade does not work after upgrade, since this code used to have time-based columns using the DATE data type.

Sample SQL Code (Oracle10g)



```
  SQL>  UPDATE member_msg_conctr
  2              SET alter_status= 'D',
  3            del_ts = SYSDATE
  4         WHERE alert_type = '20'
  5              AND member_id = 'null'
  6*            AND (SYSDATE - CAST(ts AS DATE)) > 90
SQL> /

4 rows updated.
```

After upgrade, this code uses time-based columns with the TIMESTAMP data type. Time arithmetic is now subject to different rules. This code was embedded in a JSP. The Java developer could never figure out what was wrong with it, until his team formally consulted the DBA.

Consulting Case Studies

- User Acceptance Testing (UAT)
  - Meeting User Interface and Human Factors Goals
  - Achieving Optimal Functionality and Usability.

Concentrating efforts in achieving synergy with business development leaders and managers is key to Oracle stakeholders, such as DBAs and Architects, and other infrastructure stakeholders, such as system, network, and SAN administrators to congruently attain the project goals and meet the desired milestones in a timely manner.

Consulting Case Studies

- Regression Testing after Integration.
  - Verify and Validate.
  - Testing the old or untouched modules.
  - Testing the new modules.
  - Fixing what does not work anyway.

What was changed in that code that has affected the behavior of my triggers and forms. Why the newly applied patch affecting the consistency of my results. These are issues where system integration testing may not guarantee that the existing module will not be ultimately affected, and therefore regression testing is a must. It has happened many times during my computer consulting career. Comment: The project manager never gets blamed for it! It could have been the consultant that just got involved!

Consulting Case Studies

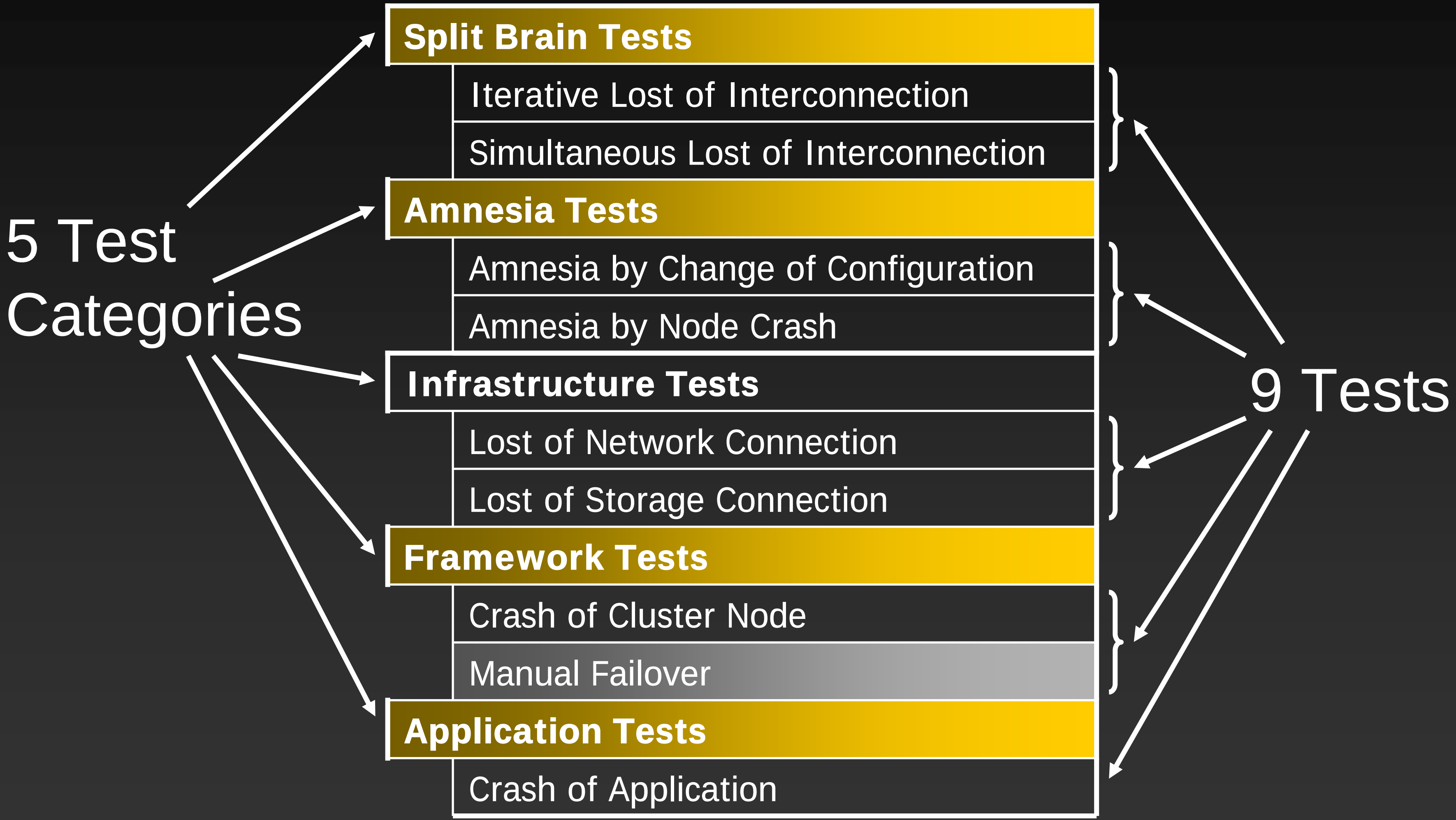
- Production QA and Maintenance
  - Business Continuity
  - Production Support
  - Deploying and Redeploying Upgraded modules
  - On-demand Reverse Engineering

Trusting that your code can rely exclusively on previous stages testing once it goes into production is probably a naive neglect.. The interdependence from your deployed application an other processes such as business continuity (BC) and high availability (HA) backup and recovery (BR), and more critically disaster recovery (DR) is as important as quality assurance in those earlier stages. Whether your use RMAN or a third-party solution, your database availability and your SAN reliability are both mission critical.

Consulting Case Studies

- Grid Testing
  - Infrastructure Testing
  - Agent, Manager and Network Connectivity
  - Tuning Automation, Pooling and Virtualization frameworks.
  - Testing Manageability and Grid Control.
  - Testing Grid Interfaces.
  - Information Retrieval
    - Data Consistency
    - Availability
    - Latency

RAC Testing



5 Test Categories

- **Split Brain Tests**
  - Iterative Lost of Interconnection
  - Simultaneous Lost of Interconnection
- **Amnesia Tests**
  - Amnesia by Change of Configuration
  - Amnesia by Node Crash
- **Infrastructure Tests**
  - Lost of Network Connection
  - Lost of Storage Connection
- **Framework Tests**
  - Crash of Cluster Node
  - Manual Failover
- **Application Tests**
  - Crash of Application

9 Tests

Source: André Feld, Technology Manager, Deutsche Post World Net

Consulting Case Studies

- Quality Assurance
  - Baselines and Timeline Control
  - The accuracy of COCOMO I I and other Project Management methodologies when Software Quality Assurance (SQA) is mission critical.
  - Downtime and Windows of time control.

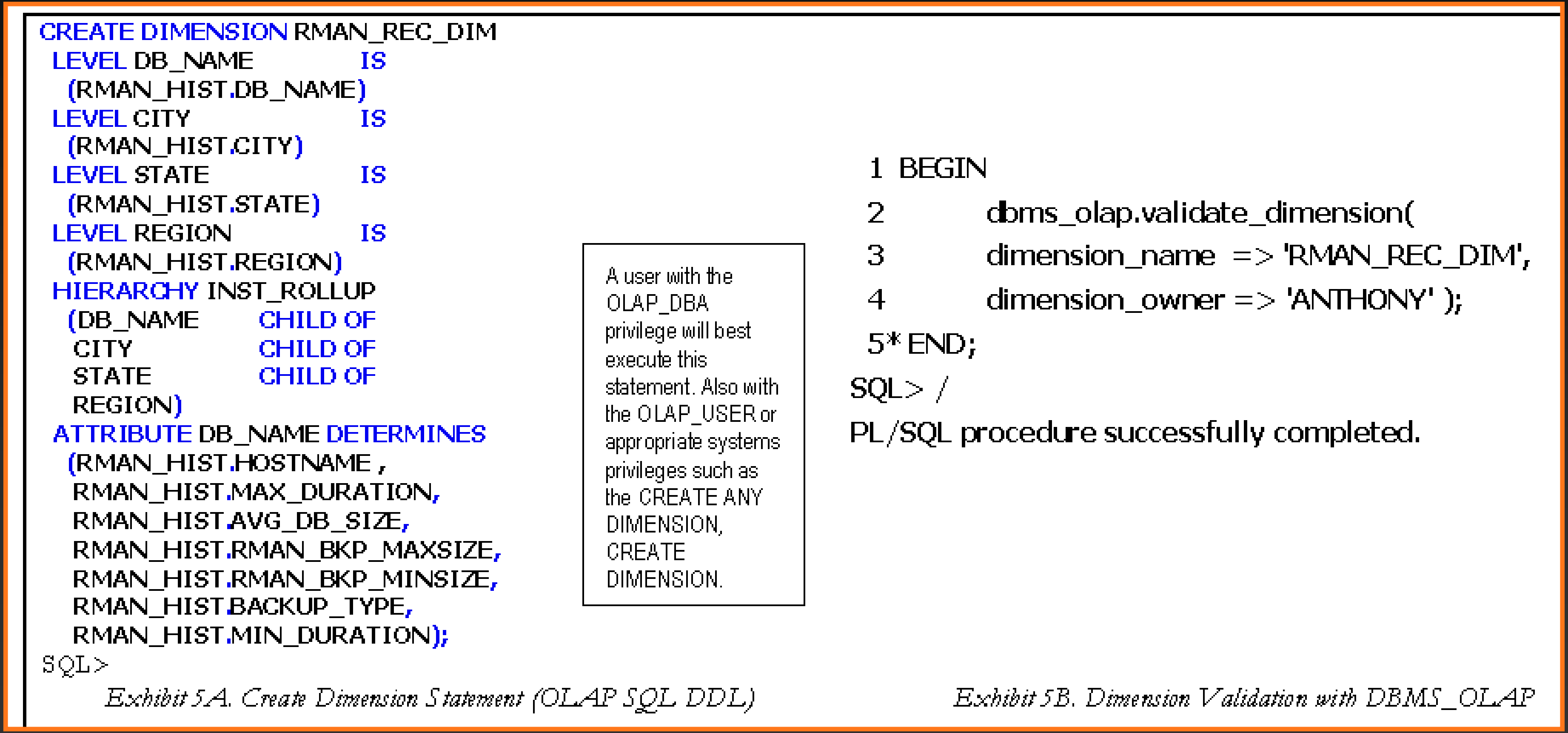
Workflow Characterization

- Front-end driven workflow.
- Back-end driven Workflow.
- Middleware-driven workflow.
  - Streams and Messaging Technologies
  - SOA and Web Services
  - Other middleware

Testing User Interfaces

- Testing Forms
  - Form, object, and block level triggers
  - Interface congruency (Usability)
  - Form process performance (Functionality)
- Testing Reports
  - Accuracy
  - Timeliness
- Testing BI Applications
  - BI Congruency
  - BI Datamart Reporting

Validating BI Objects (MOLAP)



```
CREATE DIMENSION RMAN_REC_DIM
 LEVEL DB_NAME             IS
   (RMAN_HIST.DB_NAME)
 LEVEL CITY                IS
   (RMAN_HIST.CITY)
 LEVEL STATE               IS
   (RMAN_HIST.STATE)
 LEVEL REGION              IS
   (RMAN_HIST.REGION)
 HIERARCHY INST_ROLLUP
   (DB_NAME       CHILD OF
    CITY          CHILD OF
    STATE         CHILD OF
    REGION)
 ATTRIBUTE DB_NAME DETERMINES
   (RMAN_HIST.HOSTNAME ,
    RMAN_HIST.MAX_DURATION,
    RMAN_HIST.AVG_DB_SIZE,
    RMAN_HIST.RMAN_BKP_MAXSIZE,
    RMAN_HIST.RMAN_BKP_MINSIZE,
    RMAN_HIST.BACKUP_TYPE,
    RMAN_HIST.MIN_DURATION);
SQL>
```

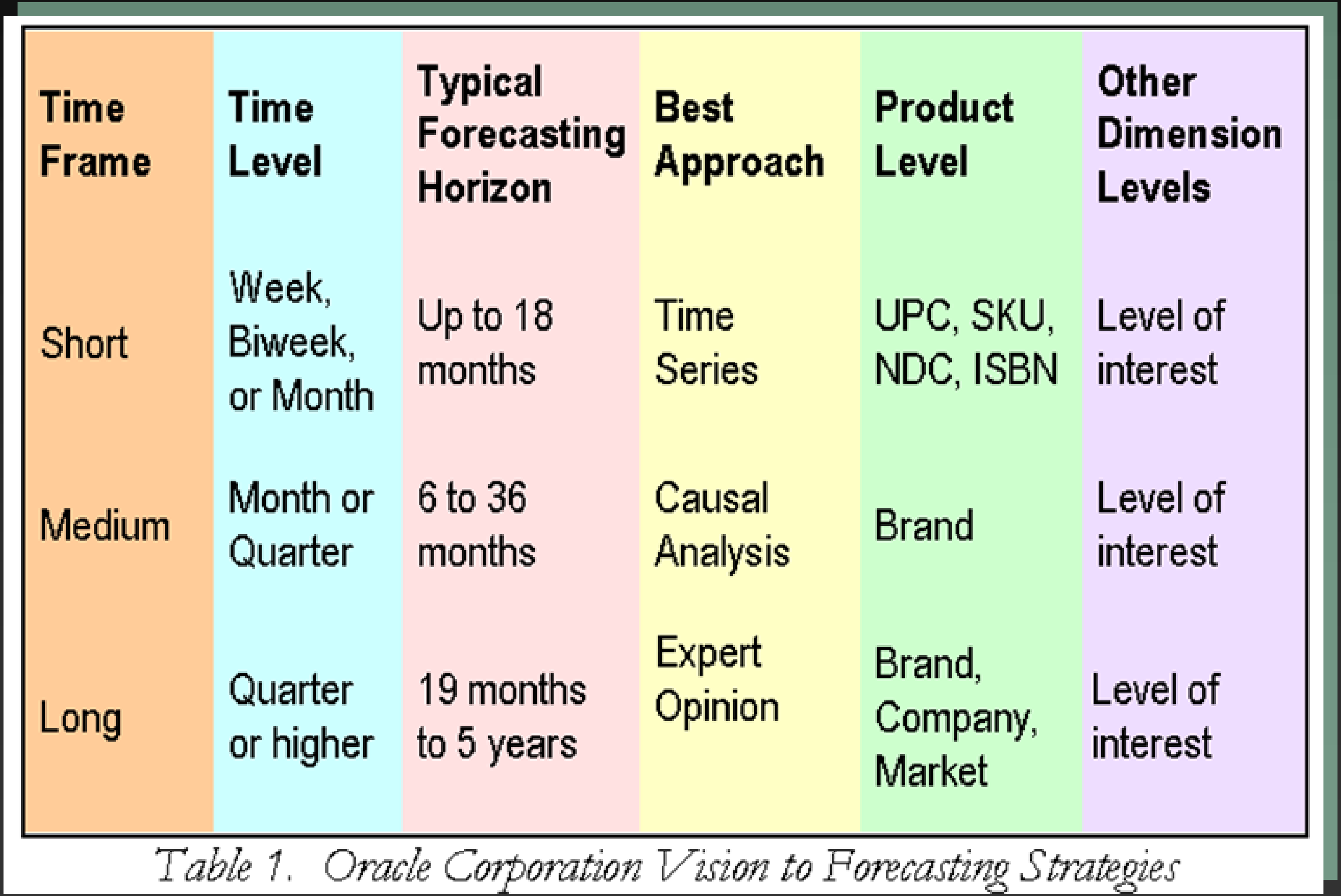
Exhibit 5A. Create Dimension Statement (OLAP SQL DDL)

A user with the OLAP_DBA privilege will best execute this statement. Also with the OLAP_USER or appropriate systems privileges such as the CREATE ANY DIMENSION, CREATE DIMENSION.

```
  1  BEGIN
  2          dbms_olap.validate_dimension(
  3          dimension_name  => 'RMAN_REC_DIM',
  4          dimension_owner => 'ANTHONY' );
  5* END;
SQL> /
PL/SQL procedure successfully completed.
```

Exhibit 5B. Dimension Validation with DBMS_OLAP

Oracle’s BI Test Approach



Time Frame	Time Level	Typical Forecasting Horizon	Best Approach	Product Level	Other Dimension Levels
Short	Week, Biweek, or Month	Up to 18 months	Time Series	UPC, SKU, NDC, ISBN	Level of interest
Medium	Month or Quarter	6 to 36 months	Causal Analysis	Brand	Level of interest
Long	Quarter or higher	19 months to 5 years	Expert Opinion	Brand, Company, Market	Level of interest

Table 1. Oracle Corporation Vision to Forecasting Strategies

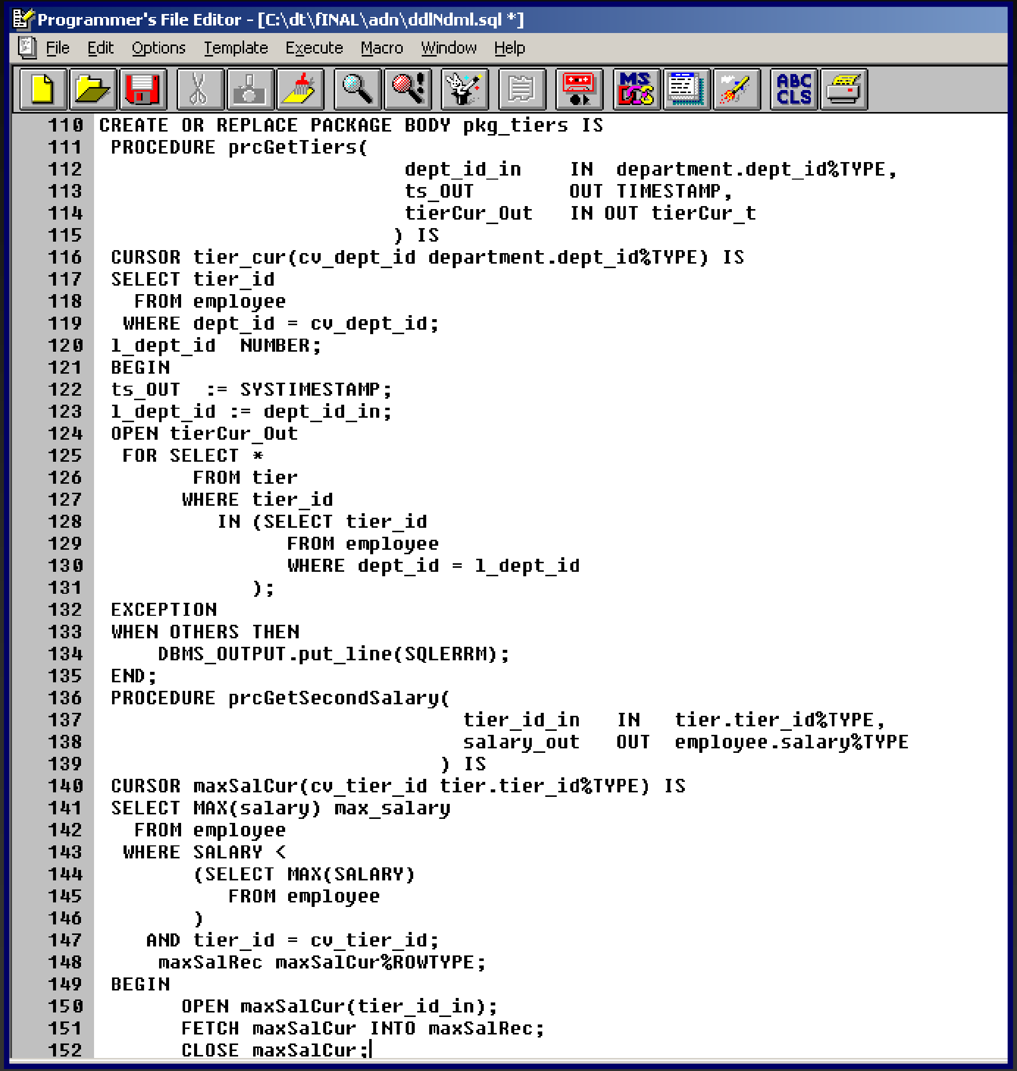
Testing SQL

- SQL Writing Best Practices
- SQL Tuning
- Upgrading and Migrating SQL

Testing PL/SQL

- Summary PL/SQL Writing Best Practices
  - Auditing
  - Maintain valid objects.
  - Test as your objects grow with requirements.
  - Entice team consistency and congruency with a version control paradigm.
- PL/SQL Tuning
  - Auditing
  - Logging
  - Tracing

Testing PL/SQL



```
CREATE OR REPLACE PACKAGE BODY pkg_tiers IS
 PROCEDURE prcGetTiers(
                          dept_id_in    IN  department.dept_id%TYPE,
                          ts_OUT        OUT TIMESTAMP,
                          tierCur_Out   IN OUT tierCur_t
                         ) IS
 CURSOR tier_cur(cv_dept_id department.dept_id%TYPE) IS
 SELECT tier_id
   FROM employee
  WHERE dept_id = cv_dept_id;
 l_dept_id  NUMBER;
 BEGIN
 ts_OUT  := SYSTIMESTAMP;
 l_dept_id := dept_id_in;
 OPEN tierCur_Out
  FOR SELECT *
        FROM tier
       WHERE tier_id
          IN (SELECT tier_id
                FROM employee
                WHERE dept_id = l_dept_id
             );
 EXCEPTION
 WHEN OTHERS THEN
     DBMS_OUTPUT.put_line(SQLERRM);
 END;
 PROCEDURE prcGetSecondSalary(
                               tier_id_in    IN   tier.tier_id%TYPE,
                               salary_out    OUT  employee.salary%TYPE
                             ) IS
 CURSOR maxSalCur(cv_tier_id tier.tier_id%TYPE) IS
 SELECT MAX(salary) max_salary
   FROM employee
  WHERE SALARY <
        (SELECT MAX(SALARY)
           FROM employee
        )
    AND tier_id = cv_tier_id;
     maxSalRec maxSalCur%ROWTYPE;
 BEGIN
       OPEN maxSalCur(tier_id_in);
       FETCH maxSalCur  INTO maxSalRec;
       CLOSE  maxSalCur;
```

An IDE can help your test, but a simple text editor could be more practical in some cases, where unit testing and manual control are more valuable overall.

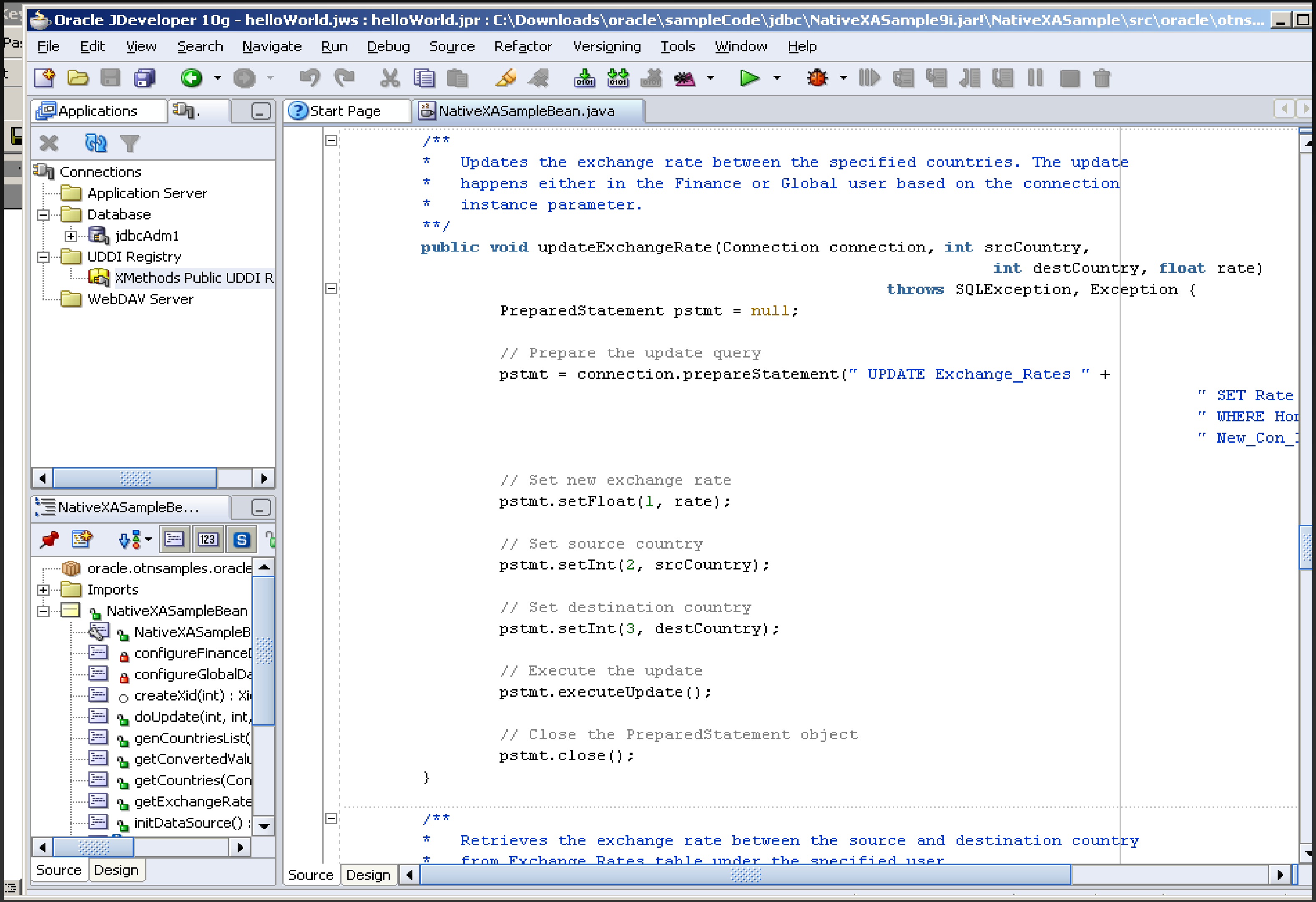
Testing Java

- Testing JSP
  - Strive for an embedded SQL needs to be neat and bind parameters properly.
  - Prefer stored procedures to dynamic SQL.
- Testing Servlets
  - Consider latency issues dealing with middleware architecture, network constraints, and HTML generation.
  - Attain a reliable consistent servlet HTML rendering and load response time.

Testing Java

- Testing EJBs
  - Establish different patterns for persistent and non-persistent EJBs as they map with other Oracle Java application models, for instance, BC4J.
  - Match Hibernate to persistent practices.
- Testing CORBA and IIOP-based Applications
  - Automation can be easy or complex depending on the processes involved.
  - Ensure that you can deal with IIOP networking issues at any time or have a network expert monitoring protocol issues.
- Using IDEs: JDeveloper, Eclipse, etc.

Testing Java



```java
/**
 *   Updates the exchange rate between the specified countries. The update
 *   happens either in the Finance or Global user based on the connection
 *   instance parameter.
 **/
public void updateExchangeRate(Connection connection, int srcCountry,
                                                      int destCountry, float rate)
                                           throws SQLException, Exception {
        PreparedStatement pstmt = null;

        // Prepare the update query
        pstmt = connection.prepareStatement(" UPDATE Exchange_Rates " +
                                                                    " SET Rate
                                                                    " WHERE Hor
                                                                    " New_Con_I



        // Set new exchange rate
        pstmt.setFloat(1, rate);

        // Set source country
        pstmt.setInt(2, srcCountry);

        // Set destination country
        pstmt.setInt(3, destCountry);

        // Execute the update
        pstmt.executeUpdate();

        // Close the PreparedStatement object
        pstmt.close();
}

/**
 *   Retrieves the exchange rate between the source and destination country
 *   from Exchange_Rates table under the specified user
```

Testing Heterogeneous Services

- Microsoft
  - Testing asp.net
  - MSMQ
- Testing Messaging Gateways
  - IBM
  - Tibco
  - Others

Testing Oracle XML

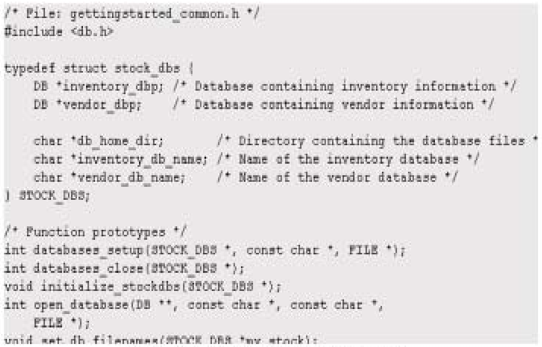
- Testing XML Schemas
- Working with Oracle XML DTDs
- Testing Oracle XML Supplied PL/SQL Packages
- Testing Oracle XML Java Supplied PL/SQL Package

Testing XML in Oracle XML DB projects could probably simplify integration in stages such as verification, validation, prior to full deployment in comparison to using versions of third party vendors.

XML Parsing, embedding, and rendering are processes that require massive and consistent testing at each stage such that they are fully functional with the finalized deployed application.

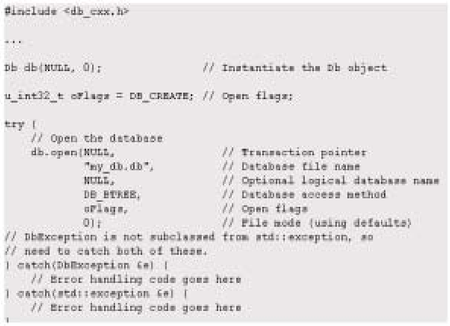
Testing Embedded Code

- Pervasive Testing on independent devices
  - Wireless Network Protocols
  - Network devices
- General Embedded Testing using driver-level programming
- Other embedded SQL using Oracle OCI
- Other embedded Testing using Berkeley DB.



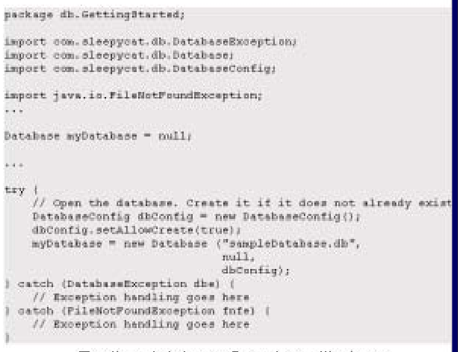
```c
/* File: gettingstarted_common.h */
#include <db.h>

typedef struct stock_dbs {
    DB *inventory_dbp; /* Database containing inventory information */
    DB *vendor_dbp;    /* Database containing vendor information */

    char *db_home_dir;       /* Directory containing the database files *
    char *inventory_db_name; /* Name of the inventory database */
    char *vendor_db_name;    /* Name of the vendor database */
} STOCK_DBS;

/* Function prototypes */
int databases_setup(STOCK_DBS *, const char *, FILE *);
int databases_close(STOCK_DBS *);
void initialize_stockdbs(STOCK_DBS *);
int open_database(DB **, const char *, const char *,
    FILE *);
void set_db_filenames(STOCK_DBS *my_stock);
```

Testing database Opening with C



```cpp
#include <db_cxx.h>

...

Db db(NULL, 0);               // Instantiate the Db object

u_int32_t oFlags = DB_CREATE; // Open flags;

try {
    // Open the database
    db.open(NULL,                // Transaction pointer
            "my_db.db",          // Database file name
            NULL,                // Optional logical database name
            DB_BTREE,            // Database access method
            oFlags,              // Open flags
            0);                  // File mode (using defaults)
// DbException is not subclassed from std::exception, so
// need to catch both of these.
} catch(DbException &e) {
    // Error handling code goes here
} catch(std::exception &e) {
    // Error handling code goes here
}
```

Testing database Opening with C++



```java
package db.GettingStarted;

import com.sleepycat.db.DatabaseException;
import com.sleepycat.db.Database;
import com.sleepycat.db.DatabaseConfig;

import java.io.FileNotFoundException;
...

Database myDatabase = null;

...

try {
    // Open the database. Create it if it does not already exist
    DatabaseConfig dbConfig = new DatabaseConfig();
    dbConfig.setAllowCreate(true);
    myDatabase = new Database ("sampleDatabase.db",
                               null,
                               dbConfig);
} catch (DatabaseException dbe) {
    // Exception handling goes here
} catch (FileNotFoundException fnfe) {
    // Exception handling goes here
}
```

Testing database Opening with Java

Data Warehouse Testing

- Extract Transform Load Testing
  - Utilities
  - Manual
  - Custom
- Tools and Functionality
  - Oracle Warehouse Builder
  - Informatica, Microstrategy, Others.
- Outcome Congruency

Unified Perspective

- SQL Functions and API
- Generic SQL DML
- OLAP DML API
  - Functions
  - Statements
- J2EE
- XML
- SOA

Oracle11g Real Application Testing (RAT) Blueprint

- Applying Database Replay
  - Allows to comfortably capture actual production workloads at the database level and replay them on your test system environment.
  - Provide complete testing on the impact of system changes is then possible, including critical concurrency characterization.

Oracle11g Real Application Testing (RAT) Blueprint

- SQL Performance Analyzer consistently identifies:
  - Structured query language (SQL)
  - Execution plan changes
  - Performance regressions
  - Problems can then be fixed using SQL.
- Tuning Advisor either:
  - Reverts to the original execution plans or
  - Performs and attains further tuning.

Convergence and Collaboration

- The Project Manager should focus on team collaboration to overcome the overall convergence of IT resources.
- Excessive lack of control on convergence could lead to data and network grid disruption whose outcome could be poor reliability and availability.

Testing to attain Compliance

- HIPAA
  - Privacy
  - Security
- Sarbannes-Oxley (SOX)
  - Rules and regulations
  - Financial data consistency
  - Protocol Settlement Verification and Validation
- MasterCard Privacy Act
  - Privacy

Concluding Remarks

- Testing Oracle IS NOT easy, but rather a complex process at any stage. Good planning could lead to simplified solutions.
- Success is likewise a systematic process: If you may mistakes in the beginning you are likely to pay for the cost of those errors later on.
- Planning well with a flexible timeline and a cost model could allow you to do your utmost in any project and work around any constraints to reach milestones timely.

Concluding Remarks

Testing is complex

Success is systematic

Planning well is good!