

Bond & Pollard Ltd

Oracle Demo Application

Installation Guide

Ian Bond
15 April 2025

Table of Contents

Introduction	3
Oracle Database Express (XE)	4
Installing Oracle XE	4
Managing Oracle	5
Starting the Oracle Services	5
Starting and Stopping the Database	7
Net Services Listener	8
Developer Tools	10
SQL*Plus	10
SQL Developer	12
The Demo Application	13
Database Security	13
Demo Schema	14
Users / Schemas	14
Applications	14
Packages	14
Tables	15
Source Code Templates	16
Directories	16
Installing the Demo Application	17
Getting Started	24
CSV File Import Demo	26
Application Development	33
Systems Development Life Cycle	34
Planning	34
Analysis	34
Design	35
Development	35
Testing	35
Implementation	36
Maintenance	36
Documentation	37
Release Management	38
Deployment Environments	38
Source Control Management	38

Modular Database Applications.....	39
Client Server Architecture.....	39
Database Design.....	40
Normalization.....	40
Indexes and Performance	46
Surrogate vs Natural Keys	46
Constraints	48
Coding Standards	49
Golden Rules for Software Development	49
Naming Conventions.....	50
Coding Style.....	52
PL/SQL Programming Tips.....	53
Packages.....	54
Functions.....	54
Data Typing	55
Performance	55

Introduction

The Demo Application has been developed as a working model of how to create a simple Oracle Database Application.

The application comprises:

- A single schema based on Oracle Education's training database, known as the Scott schema.
- A setup.exe installer that generates then runs installation scripts that create the schema, load seed data and compile the packages.
- Additional tables and related objects to extend the functionality of the basic schema.
- A directory structure containing the sample application program source code, templates, installation and admin scripts, SQL reports and documentation.
- A simple application to import data into, and export data from the database using CSV files.
- PL/SQL Packages demonstrating useful functions.
- Source code templates.

The following documentation is included:

- A guide to creating an application development environment.
- A simple set of coding standards.
- A template technical specification document.
- Functional specifications.
- Technical specifications.
- A user guide for the data import application.

Oracle Database Express (XE)

Installing Oracle XE

The first task is to install Oracle Database Express, and SQL Developer, both of which can be downloaded from Oracle's website.

Please refer to the [Database Express Installation Guide](#).

1. Login to your Oracle account on the Web.
2. Download Oracle XE for Windows.
3. Unzip the installation file.
4. Run setup.exe
5. Make a note of the connection strings.

Tip: Store usernames and passwords securely in an encrypted database, such as Bitwarden.

Managing Oracle

Starting the Oracle Services

The Oracle Database Services must be started prior to accessing the database and should start automatically when you start your computer.

The first pluggable database XEPDB1 opens automatically when the Oracle Database Service is started. Other pluggable databases remain closed by default and must be opened manually or set to open automatically.

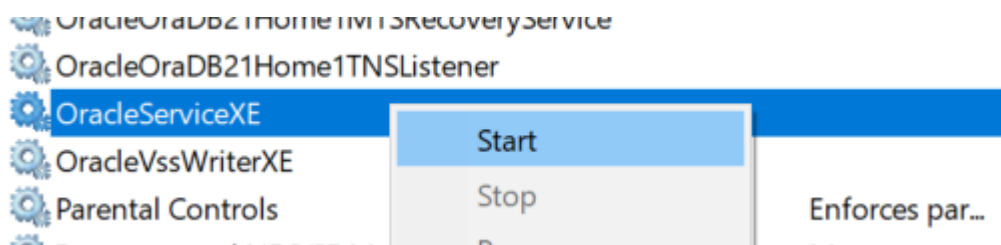
To manage the Oracle Services, run the Windows Services app.



Search for the Oracle services, which should look like the example below:

OracleJobSchedulerXE		Disabled	NT SERVICE...
OracleOraDB21Home1MTSRecoveryService	Running	Automatic	NT SERVICE...
OracleOraDB21Home1TNSListener	Running	Automatic	NT SERVICE...
OracleServiceXE	Running	Automatic	NT SERVICE...
OracleVssWriterXE		Automatic	NT SERVICE...

To start the Oracle Service, right-click on **OracleServiceXE**, and select Start.



To set the start-up properties of a service, right-click, select properties and select Automatic, Manual or Disabled from the Startup type list.

OracleServiceXE Properties (Local Computer) X

General Log On Recovery Dependencies

Service name: OracleServiceXE

Display name: OracleServiceXE

Description:

Path to executable:
"d:\oracle\bin\ora_11g\bin\ORACLE.EXE" XE

Startup type: Automatic

Automatic
Automatic (Delayed Start)
Automatic
Manual
Disabled
Stopped

Service status: Stopped

Start Stop Pause Resume

You can specify the start parameters that apply when you start the service from here.

Start parameters:

OK Cancel Apply

Starting and Stopping the Database

*Starting the Database via SQL*Plus*

Run SQL*Plus from the command prompt, connect as SYSDBA:

```
C:\> SQLPLUS / AS SYSDBA
```

To start the database:

```
SQL> STARTUP
```

The first pluggable database should open automatically. To open all pluggable databases:

```
SQL>ALTER PLUGGABLE DATABASE ALL OPEN
```

*Shutdown the Database via SQL*Plus*

You should always shutdown the Oracle database before shutting down your computer, to prevent data corruption due to the file system suddenly being taken away from the running database.

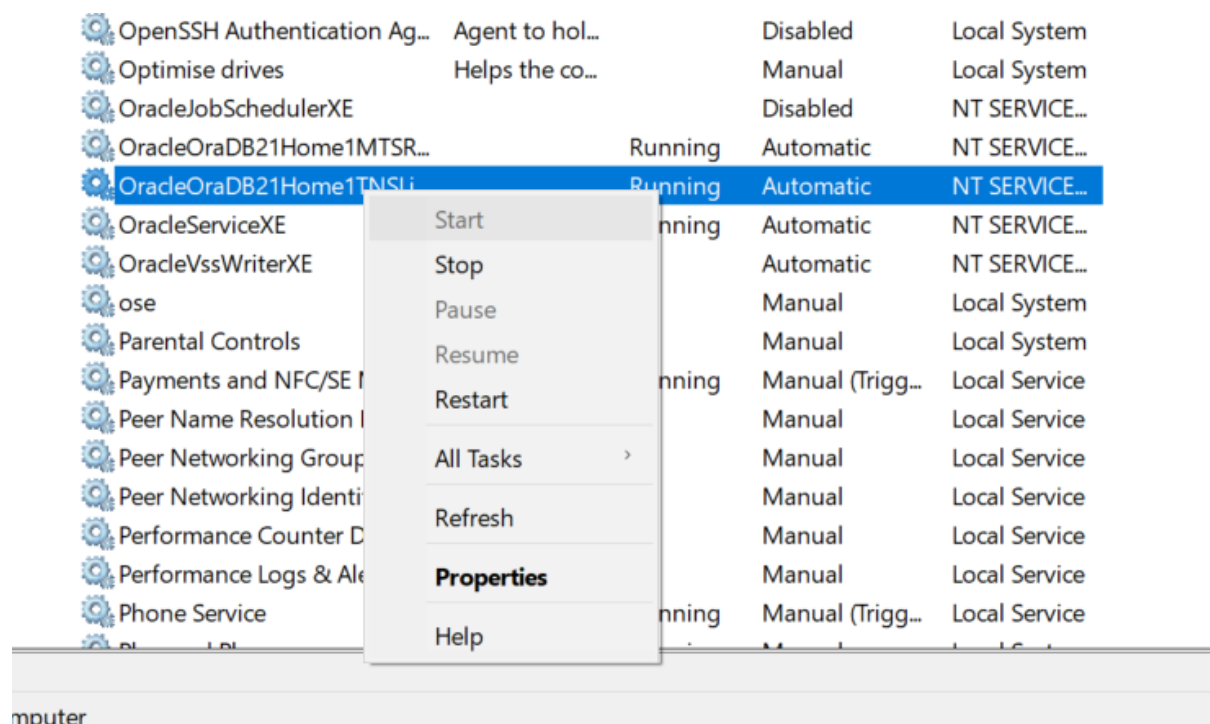
To shut down the database:

```
SQL>SHUTDOWN IMMEDIATE
```


Net Services Listener

The listener processes on a server detect incoming requests from clients for connection, by default on port 1521, and manage network-traffic once clients have connected to an Oracle database. The listener uses a configuration-file named **listener.ora** to help keep track of names, protocols, services and hosts.

You can start the listener using the Microsoft services app, or from the command prompt:



To check the listener status from the command prompt:

C:\> LSNRCTL STATUS

```
LSNRCTL for 64-bit Windows: Version 21.0.0.0.0 - Production on 23-JUL-2022 12:33:06

Copyright (c) 1991, 2021, Oracle. All rights reserved.

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=LAPTOP-F9IQQLLC.lan)(PORT=1521)))
STATUS of the LISTENER
-----
Alias                     LISTENER
Version                  TNSLSNR for 64-bit Windows: Version 21.0.0.0.0 - Production
Start Date               13-JUL-2022 23:41:48
Uptime                   9 days 12 hr. 51 min. 17 sec
Trace Level              off
Security                 ON: Local OS Authentication
SNMP                     OFF
Default Service          XE
Listener Parameter File  D:\oracle\homes\OraDB21Home1\network\admin\listener.ora
Listener Log File        D:\oracle\diag\tnslsnr\LAPTOP-F9IQQLLC\listener\alert\log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=LAPTOP-F9IQQLLC)(PORT=1521)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(PIPENAME=\\.\pipe\EXTPROC1521ipc)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=127.0.0.1)(PORT=5500))(Security=(my_wallet_directory=D
:\ORACLE\admin\XE\xdb_wallet))(Presentation=HTTP)(Session=RAW))
Services Summary...
Service "71c33b1ba5b341c9bcb3f58ce1c993dc" has 1 instance(s).
  Instance "xe", status READY, has 2 handler(s) for this service...
Service "CLRExtProc" has 1 instance(s).
  Instance "CLRExtProc", status UNKNOWN, has 1 handler(s) for this service...
Service "XE" has 1 instance(s).
  Instance "xe", status READY, has 2 handler(s) for this service...
Service "XEXDB" has 1 instance(s).
  Instance "xe", status READY, has 1 handler(s) for this service...
Service "xepdb1" has 1 instance(s).
  Instance "xe", status READY, has 2 handler(s) for this service...
The command completed successfully
```

Developer Tools

SQL*Plus

The SQL*Plus tool runs from the Windows command prompt, with the following format:

Note that square brackets [] indicate optional parameters, angle brackets < > are required.

C:\> SQLPLUS <USERNAME>/<PASSWORD>@//<HOSTNAME>[:PORT]/<DBNAME> [AS SYSDBA]

<username>	Database username, e.g. SYS
<password>	Password you specified during installation
<hostname>	Name of the database server, or its IP address. To reference the current computer, you can use <i>localhost</i> .
[:port]	Must be specified if the listener is not configured to use the default port 1521
<dbname>	XE for the container XEPDB1 for the first pluggable database
[AS SYSDBA]	This is required for the SYS user, otherwise you get the error "SP2-0157: unable to CONNECT to ORACLE after 3 attempts, exiting SQL*Plus"

If you are a database administrator running SQL*Plus on the database server, you can connect to the container database directly by running the following command.

C:\> SQLPLUS / AS SYSDBA

Connect to container database XE

C:\> SQLPLUS SYS/<PASSWORD>@//LOCALHOST:1521/XE AS SYSDBA

C:\> SQLPLUS SYS/<PASSWORD>@//LOCALHOST/XE AS SYSDBA

C:\> SQLPLUS SYS/<PASSWORD>@//192.168.1.225:1521/XE AS SYSDBA

Connect to first pluggable database XEPDB1

C:\> SQLPLUS SYS/<PASSWORD>@//LOCALHOST:1521/XEPDB1 AS SYSDBA

C:\> SQLPLUS SYS/<PASSWORD>@//LOCALHOST/XEPDB1 AS SYSDBA

C:\> SQLPLUS SYS/<PASSWORD>@//192.168.1.225:1521/XEPDB1 AS SYSDBA

Check which database you are connected to:

SQL> SHOW CON_NAME

```
C:\Users\ianbo>sqlplus / as sysdba

SQL*Plus: Release 21.0.0.0.0 - Production on Sat Jul 23 13:05:17 2022
Version 21.3.0.0.0

Copyright (c) 1982, 2021, Oracle. All rights reserved.

Connected to:
Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0

SQL> show con_name

CON_NAME
-----
CDB$ROOT
SQL> _
```

To switch to the first pluggable database:

SQL> ALTER SESSION SET CONTAINER=XEPDB1;

```
SQL> ALTER SESSION SET CONTAINER=XEPDB1;

Session altered.

SQL> show con_name

CON_NAME
-----
XEPDB1
SQL> _
```

To switch to the container database:

SQL> ALTER SESSION SET CONTAINER=CDB\$ROOT;

```
SQL> ALTER SESSION SET CONTAINER=cdb$root;

Session altered.

SQL> show con_name

CON_NAME
-----
CDB$ROOT
SQL> _
```

SQL Developer

Download the SQL Developer tool from the Oracle website.

Create a connection for the SYS user, with role SYSDBA.

Name:

Database Type:

User Info Proxy User

Authentication Type:

Username: Role:

Password: ☒ Save Password

Connection Type:

Details Advanced

Hostname:

Port:

☐ SID

☒ Service name:

Tip: Restrict access to the SYS user and store the password securely in an encrypted database.

The Demo Application

Database Security

The worst, and most commonly occurring security mistake is to have all your database objects in one schema and then give all your users and applications the schema's password. This is an extremely dangerous thing to do. Anybody could connect to your schema and start dropping tables or changing the table structures.

Instead:

- Create a schema that owns all the database objects, the Owing Schema, that no users or applications ever connect to.
- Prevent users from connecting to the Owing Schema. Lock the account and turn off authentication so no clues are given that the account exists. If someone tries to logon, they will get an invalid username/password error instead of a message saying the account is locked, which would give away its existence, and importance.
- Create a secondary schema for the users to connect to the database with, that has limited privileges, and the necessary grants to access objects in the Owing Schema.

This simple approach will prevent users from truncating and dropping tables. If you are giving privileges such as delete and update, there is a risk that people could alter the data, but at least you have prevented structural changes to the database.

Demo Schema

The demo application database schema is based on Oracle Education's Scott schema, with some additional features and sample PL/SQL applications.

Users / Schemas

User Name	Description
appsdemo	This is the owning schema, which contains all the demo database objects. No users or applications must ever connect to the database as this user, as it would be a major security risk.
demo_connect	This user has the minimal privileges, and grants to the appsdemo schema, necessary to run the demo applications.

Applications

Application	Description
CSV Data Import	Import data from CSV files into the database. A simple demo has been provided, along with a sales order import that has more complex validation.
CSV Data Export	Export data to CSV file. A simple demo, and a sales order export have been provided.

Packages

The following packages have been provided.

Package Name	Description
EXPORT	Export data (orders) to CSV files.
IMPORT	Import data from CSV files with validation and error handling. Includes order import and a simple demo.
ORDERRP	Rules package for Order related functions, for example <i>currentprice</i> returns the price that is currently in effect for a product. This saves repeating code and makes maintenance easier.
PLSQL_CONSTANTS	Define non-table related data types and constants such as directory names, delimiter characters.
UTIL_ADMIN	Admin functions such as standardised error handling.

Package Name	Description
UTIL_DATE	Date manipulation functions. Date of Easter and related holidays. Is date a working day? Last working day of month.
UTIL_FILE	Load data from an external CSV file into a staging table.
UTIL_NUMERIC	Number manipulation functions. Base conversion, factorial, sort numbers, convert integer to an alphabetic code.
UTIL_STRING	String handling functions. Extract fields from a delimited string (used by CSV import function), sort strings, convert escaped characters to formatting characters (\n becomes New Line ASCII character 10).

Tables

Oracle Demo

Table Name	Description
BONUS	Employee salary, commission
CUSTOMER	Customer table
DEPT	Departments
DUMMY	Demo table of numbers used for SQL exercises
EMP	Employees
ITEM	Item (order lines) table
ORD	Order table (order lines in associated ITEM table)
PRICE	Product prices. Minimum, Current price effective start and end date.
PRODUCT	Product table, foreign key reference by ITEM
SALGRADE	Employee job grade salary ranges

Demo Application

Table Name	Description
APPLOG	Application message log: messages with a severity, timestamp, user and program name.
APPSEVERITY	Application message severity: Info, Warning, Error
COUNTRY	Maintain a list of valid country codes.
COUNTRY_HOLIDAY	Maintain holiday dates by country and year. Used by the date functions to calculate working days: last working day of month, number of working days between two dates etc.
DEMO	Simple table with a date and text column, used to demonstrate data import/export functions.
IMPORTCSV	Staging table for CSV data to be imported into the database. Function <i>util_file.load_csv</i> loads CSV data into this table. Data fields are in a delimited string "field1",field2,field3,"field4" etc.
IMPORTERROR	When importing CSV data, it will be validated and all errors logged here. Separate from the application log to make it easier to report and manage.

Source Code Templates

Source Type	Template
DOS Batch file	..\templates\dos_script.bat
PL/SQL Package Body	..\templates\pkg_template_body.pkb
PL/SQL Package Specification	..\templates\pkg_template_spec.pks
SQL script	..\templates\sql_template.sql

Directories

Directory Name	Contents
admin	Database admin scripts. Start database. Create users. Check for invalid objects.
bin	Binaries. Executable images (compiled programs)
com	DOS batch scripts.
config	Configuration files. You must edit these to specify the database service name, application owner (schema), and directory paths.
data	Data (import and export files)
documentation	Systems documentation, specifications, user guides
install	Scripts to automatically install the database schema, load seed data and compile packages
log	Program logs
out	User report output
plsql	Package source code
sql	SQL scripts: reports, scripts to execute package code
src	Program source code, e.g., C source files
templates	Code templates
test	Scripts and SQL to test programs, automated test scripts

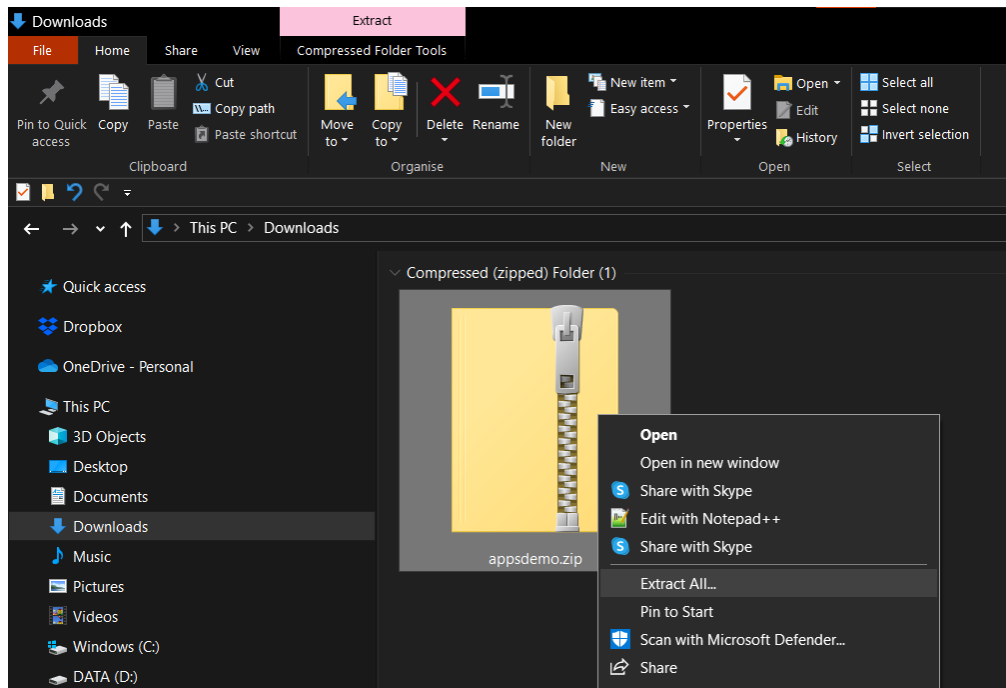
Installing the Demo Application

Download the archive file containing the demo application from the Bond & Pollard website:

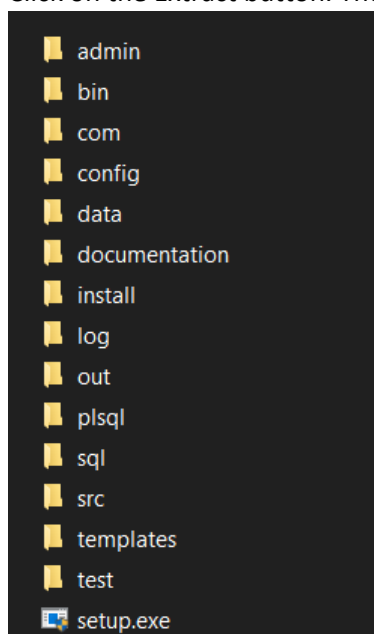
<http://www.bondpollard.co.uk/documents/appsdemo.zip>

The appsdemo.zip file will be downloaded to your PC downloads folder.

Right-click on appsdemo.zip and select Extract All.



Click on the Extract button. The following directories will be created under appsdemo:



Go to the appsdemo folder and double-click on **setup.exe** to run the installer.

A User Account Control windows will be displayed asking you to confirm you want to run the program. Click the YES button.



```
D:\users\ianbo\Downloads\appsdemo\appsdemo\setup.exe
*****
* ORACLE DEMO APPLICATION SETUP *
* (c) Bond & Pollard Ltd 2025 *
*****
This program will install the Oracle demo application.

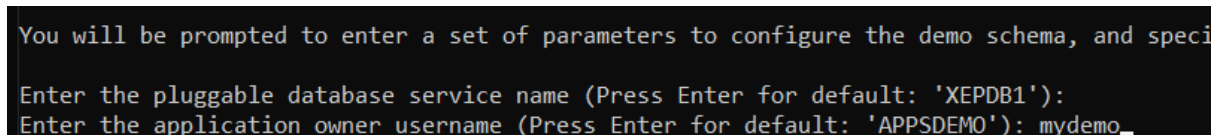
Prerequisites:
You must install Oracle Express first.

You will be prompted to enter a set of parameters to configure the demo schema, and specify the installation location.

Enter the pluggable database service name (Press Enter for default: 'XEPDB1'): _
```

Press ENTER to accept the default for the database service XEPDB1, the default first pluggable database.

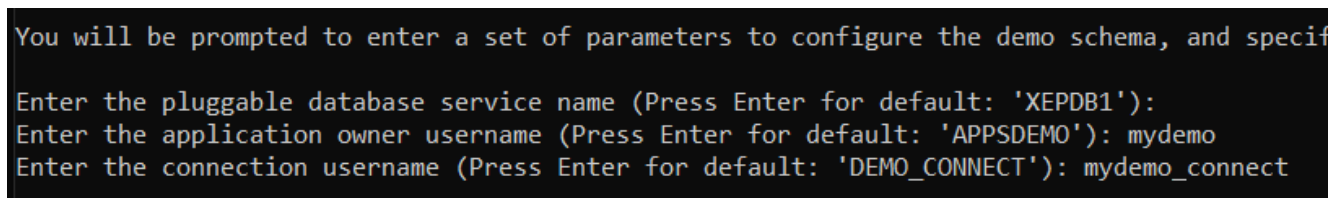
Enter *mydemo* as the name of the application owner. The application owner is the user who owns all the database schema objects such as tables. Access must be strictly controlled to this user.



```
You will be prompted to enter a set of parameters to configure the demo schema, and specify the installation location.

Enter the pluggable database service name (Press Enter for default: 'XEPDB1'):
Enter the application owner username (Press Enter for default: 'APPSDEMO'): mydemo_
```

Enter *mydemo_connect* as the name of the connection user. This user has limited privileges and will be used to run the application.



```
You will be prompted to enter a set of parameters to configure the demo schema, and specify the installation location.

Enter the pluggable database service name (Press Enter for default: 'XEPDB1'):
Enter the application owner username (Press Enter for default: 'APPSDEMO'): mydemo
Enter the connection username (Press Enter for default: 'DEMO_CONNECT'): mydemo_connect
```

Press *enter* to accept the default database listener port 1521:

Next you will be prompted for APP_HOME directory. This directory contains all the SQL scripts and programs.

The directory must exist, so create it before continuing.

We will specify the directory **d:\test_app**

```
Enter the pluggable database service name (Press Enter for default: 'XEPDB1'):  
Enter the application owner username (Press Enter for default: 'APPSDEMO'): mydemo  
Enter the connection username (Press Enter for default: 'DEMO_CONNECT'): mydemo_connect  
Enter the listener port (Press Enter for default: '1521'):  
  
Specify the application home directory APP_HOME. Include the drive, separate each directory w  
Do not end with a \ unless specifying just the drive root e.g. D:\  
You may include spaces and ampersands & in the APP_HOME path.  
Do not include quotes "  
Enter the app installation directory path (Press Enter for default: 'C:\'): d:\test_app
```

Next you will be prompted for the APP_DATA directory. This directory contains external files such as CSV files used for data import to the database.

The directory must exist, so create it before continuing.

IMPORTANT: This directory path must not contain spaces or special characters such as &.

We will specify the directory **d:\test_user_data**

```
Specify the data home directory DATA_HOME. Include the drive, separate each director  
Do not end with a \ unless specifying just the drive root e.g. D:\  
Spaces are not allowed in this path.  
Do not include ampersands &.  
Do not include quotes ".  
Enter the data directory path (Press Enter for default: 'C:\'): d:\test_user_data
```

The setup parameters will be displayed for you to check:

```
Specify the data home directory DATA_HOME. Include the drive, separa  
Do not end with a \ unless specifying just the drive root e.g. D:\  
Spaces are not allowed in this path.  
Do not include ampersands &.  
Do not include quotes ".  
Enter the data directory path (Press Enter for default: 'C:\'): d:\t  
SETUP PARAMETERS  
=====  
Source files extracted to: D:\users\ianbo\Downloads\appsdemo\appsdemo  
DBSERVICE = XEPDB1  
Listener port = 1521  
Database connection string = //localhost:1521/XEPDB1  
APP_OWNER = mydemo  
CONNECT_USER = mydemo_connect  
APP_HOME directory = d:\test_app\XEPDB1\mydemo  
SQL_APP_HOME directory = d:\\test_app\\XEPDB1\\mydemo  
DATA_HOME directory = d:\test_user_data\XEPDB1\mydemo\data  
SQL_DATA_HOME directory = d:\\test_user_data\\XEPDB1\\mydemo\\data  
  
Do you want to continue with the installation (Y or N)?_
```

Make a note of the names of APP_OWNER and CONNECT_USER. You will need these to connect to the database.

You will be prompted to confirm that you want to continue with the installation. Enter Y.

Progress bars will be displayed as the APP_HOME and APP_DATA directories are created.

```
Do you want to continue with the installation (Y or N)?y
Creating APP_HOME. Copying files from D:\users\ianbo\Downloads\appsdemo\appsdemo to d:\te

[#####] 100%

APP_HOME created.

Creating DATA_HOME. Copying files from D:\users\ianbo\Downloads\appsdemo\appsdemo\data to
a...

[#####] 100%

DATA_HOME created.

Creating d:\test_app\XEPDB1\mydemo\config\set_env.sql...
SQL script generated: d:\test_app\XEPDB1\mydemo\config\set_env.sql
SQL script set_env.sql created.
Creating d:\test_app\XEPDB1\mydemo\config\set_env.bat...
Script generated: d:\test_app\XEPDB1\mydemo\config\set_env.bat
Script set_env.bat created.
Creating d:\test_app\XEPDB1\mydemo\install\auto_install.sql...
SQL script generated: d:\test_app\XEPDB1\mydemo\install\auto_install.sql
SQL Script auto_install.sql created.
Creating database objects.
Executing: sqlplus / as sysdba @"d:\test_app\XEPDB1\mydemo\install\auto_install.sql"
```

The following scripts are created automatically using the parameters you specified:

- Set_env.sql – sets parameters for sql scripts that run from the command line.
- Set_env.bat – sets parameters required for batch programs that run at the command prompt.
- Auto_install.sql – this SQL script runs automatically to create the database schema objects, load seed data into the tables, and compile the PL/SQL packages.

SQL*Plus runs the `auto_install.sql` script to create the database objects. You will be prompted to enter the password for the application owner *mydemo*. I have chosen *tiger* the name of Bruce Scott's cat.

TIP: Record the app owner and password in a secure encrypted app such as Bitwarden.

```
Executing: sqlplus / as sysdba @"d:\test_app\XEPDB1\mydemo\install\auto

SQL*Plus: Release 21.0.0.0.0 - Production on Tue Mar 11 20:04:30 2025
Version 21.3.0.0.0

Copyright (c) 1982, 2021, Oracle. All rights reserved.

Connected to:
Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0

Enter the password for mydemo: tiger_
```

Next you will be asked to specify a password for the connection user *mydemo_connect*.

TIP: Record the connect username and password in a secure encrypted app such as Bitwarden.

```
Connected to:
Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0

Enter the password for mydemo: tiger
Enter the password for the database connection user mydemo_connect: mydemo25
Enter SYS password: _
```

You will be prompted for the SYS password that you created earlier when you installed Oracle.

Enter the SYS password and press enter. The script will then proceed to create the database objects, load data into tables, and compile packages.

A summary will be displayed.

```
INSTALLATION SUMMARY:
=====
Database Service   : XEPDB1
Application Owner  : mydemo
Connection User    : mydemo_connect
Application Home    : d:\test_app\XEPDB1\mydemo
Data Home          : d:\test_user_data\XEPDB1\mydemo\data
Setup Log          : install.log

Installation complete. Press RETURN to exit.
```

A log file called *install.log* is created in the download directory which you can check for errors.

Press *enter* to exit from the installer.

A shortcut named as the app owner will be created on the desktop. You can click on this to start the database and open the command line, with the environment configured for your Oracle app.



The installation program should have done the following:

- Create directories:
 - APP_HOME
 - DATA_HOME
- Create configuration scripts:
 - Set_env.sql
 - Set_env.bat
 - Auto_install.sql
- Install schema:
 - Create Owner Schema
 - Create database connection user
 - Create database directories
 - Create sequences
 - Create tables and indexes
 - Grant table access privileges to connection user
 - Create synonyms for tables
- Seed data:
 - Load data into tables
- Compile PL/SQL packages:
 - Compile packages
 - Grant execute privileges to connection user
 - Create synonyms for packages
- Lock schema:
 - Lock the Owner Schema, and set no authentication to keep it secure

Launch SQL Developer and create a connection for *mydemo_connect*:

Username: mydemo_connect

Password: (the password you specified during installation)

Service name: check the radio button, and enter XEPDB1.

New / Select Database Connection

Connection Na...	Connection De...
appsdemo	appsdemo@//l...
bond_i DEVEL...	bond_i@//loca...
demo_connect	demo_connect...
ian	ian@//localho...
invoices	invoices@//loc...
Liz	liz@//localhost...
scott	scott@//localh...
SYS as SYSDBA	sys@//localho...
tommy	tommy@//loca...

Name: mydemo_connect

Database Type: Oracle

User Info: Proxy User

Authentication Type: Default

Username: mydemo_connect

Role: default

Password:

Save Password: ☒

Connection Type: Basic

Details: Advanced

Hostname: localhost

Port: 1521

☐ SID: xe

☒ Service name: XEPDB1

Status: Success

Buttons: Help, Save, Clear, Test, Connect, Cancel

Click Test to check the connection works, then click save.

Getting Started

Click on the desktop icon to start Oracle and get to the command prompt.



```
mydemo

D:\test_app\XEPDB1\mydemo\com>ECHO OFF
Starting Oracle environment for database service XEPDB1 application mydemo

SQL*Plus: Release 21.0.0.0.0 - Production on Tue Mar 11 21:26:00 2025
Version 21.3.0.0.0

Copyright (c) 1982, 2021, Oracle. All rights reserved.

Connected to:
Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0

Starting the Oracle Database Service XEPDB1 for application mydemo
ORA-01081: cannot start already-running ORACLE - shut it down first
Disconnected from Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0
XEPDB1\mydemo>
```

The command prompt is set as **XEPDB1\mydemo>**

To run SQL*Plus enter the command:

sqlplus mydemo_connect/password@//localhost/xepdb1

```
C:\% mydemo - sqlplus mydemo_connect/mydemo25@//localhost/xepdb1
Starting Oracle environment for database service XEPDB1 application mydemo

SQL*Plus: Release 21.0.0.0.0 - Production on Tue Mar 11 22:26:14 2025
Version 21.3.0.0.0

Copyright (c) 1982, 2021, Oracle. All rights reserved.

Connected to:
Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0

Starting the Oracle Database Service XEPDB1 for application mydemo
ORA-01081: cannot start already-running ORACLE - shut it down first
Disconnected from Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0
XEPDB1\mydemo>sqlplus mydemo_connect/mydemo25@//localhost/xepdb1

SQL*Plus: Release 21.0.0.0.0 - Production on Tue Mar 11 22:26:53 2025
Version 21.3.0.0.0

Copyright (c) 1982, 2021, Oracle. All rights reserved.

Last Successful login time: Tue Mar 11 2025 22:24:26 +00:00

Connected to:
Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0

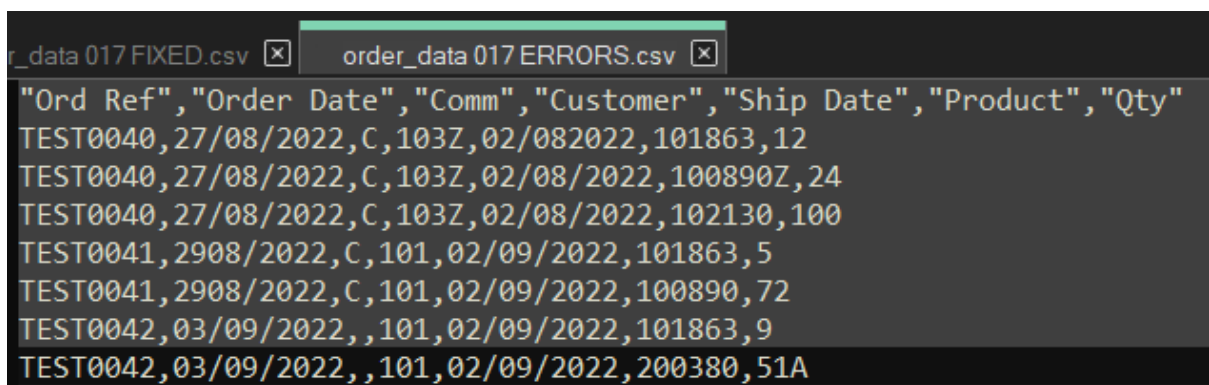
SQL> _
```

CSV File Import Demo

In this demonstration we will:

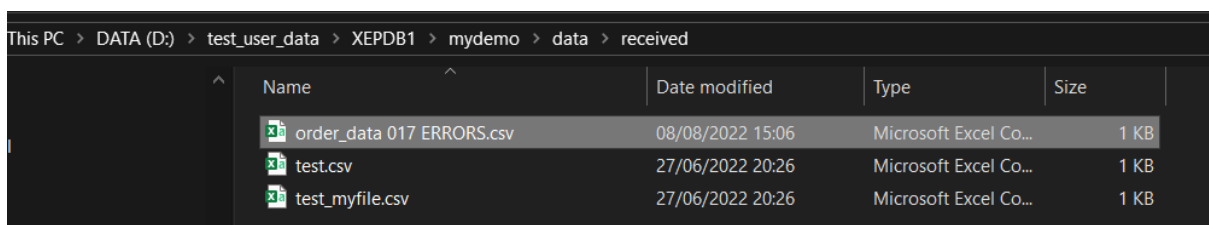
1. Attempt to import order data from a CSV file which contains several errors.
2. Run a report to find out what the errors were.
3. Fix the errors in the CSV file.
4. Run the import process again with the corrected CSV file.
5. Run an order report to view the newly imported orders.




For this demonstration we will use the test order file **order_data 017 ERRORS.csv** which is in the *data\pending* directory. Note that this file contains a number of errors, so the import will fail.



```
"Ord Ref","Order Date","Comm","Customer","Ship Date","Product","Qty"
TEST0040,27/08/2022,C,103Z,02/082022,101863,12
TEST0040,27/08/2022,C,103Z,02/08/2022,100890Z,24
TEST0040,27/08/2022,C,103Z,02/08/2022,102130,100
TEST0041,2908/2022,C,101,02/09/2022,101863,5
TEST0041,2908/2022,C,101,02/09/2022,100890,72
TEST0042,03/09/2022,,101,02/09/2022,101863,9
TEST0042,03/09/2022,,101,02/09/2022,200380,51A
```

Copy the CSV file from the data directory Pending to Received.



This PC > DATA (D:) > test_user_data > XEPDB1 > mydemo > data > received				
Name	Date modified	Type	Size	
 order_data 017 ERRORS.csv	08/08/2022 15:06	Microsoft Excel Co...	1 KB	
 test.csv	27/06/2022 20:26	Microsoft Excel Co...	1 KB	
 test_myfile.csv	27/06/2022 20:26	Microsoft Excel Co...	1 KB	

Click on the desktop shortcut.



Run the *import_order* batch program. Enter the following commands at the prompt:

```
XEPDB1\mydemo>cd com  
XEPDB1\mydemo>import_order_
```

You will be prompted to enter the password for *mydemo_connect*.

Enter the password that you specified during installation.

```
C:\mydemo  
SQL*Plus: Release 21.0.0.0.0 - Production on Tue Mar 11 22:45:30 2025  
Version 21.3.0.0.0  
  
Copyright (c) 1982, 2021, Oracle. All rights reserved.  
  
Last Successful login time: Tue Mar 11 2025 22:43:43 +00:00  
  
Connected to:  
Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production  
Version 21.3.0.0.0  
  
old 2: v_filename VARCHAR2(100) := '&1';  
new 2: v_filename VARCHAR2(100) := 'order_data 017 ERRORS.csv';  
Information from program at 11-MAR-25 22:03:30.464000000 Message: Order Data  
Import from file: order_data 017 ERRORS.csv SQLERRM is (Log mode is S)  
Error from program IMPORT.ORD_IMP at 11-MAR-25 22:03:30.599000000 Message:  
Invalid data importing file order_data 017 ERRORS.csv SQLERRM is ORA-20001:  
(Log mode is B)  
Error from program IMPORT_ORDER.SQL at 11-MAR-25 22:03:30.606000000 Message:  
Error importing file order_data 017 ERRORS.csv SQLERRM is ORA-20099: Order  
import failed. View errors in IMPORTERROR for file order_data 017 ERRORS.csv  
(Log mode is B)  
  
PL/SQL procedure successfully completed.  
  
Disconnected from Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production  
Version 21.3.0.0.0  
  
XEPDB1\mydemo>
```

The program reported that there were errors importing the file *order_data 017 ERRORS.csv*.

To investigate further, open SQL Developer, connect to *mydemo_connect*, then open and run the SQL script ***applog***.

			Bond and Pollard Limited =====			
Application Log				Page: 1		
Message	Logged At	Username	SQLERRM	Program	Severity	RECID
Invalid data importing file or der_data 017 ERRORS.csv	11/03/25 22:03:30.59 5000	MYDEMO	ORA-20001:	IMPORT.ORD_I MP	Error	1
Error importing file order_dat a 017 ERRORS.csv	11/03/25 22:03:30.60 5000	MYDEMO	ORA-20099: Order import failed . View errors in IMPORTERROR f .SQL or file order_data 017 ERRORS. csv	IMPORT_ORDER	Error	2

Note the error messages reported for the order file we tried to import.

To view the detailed error messages, run the report *import_errors.sql*.

Note that Key Value refers to the Order Reference field in the CSV file. The CSV data is shown against each error message so you can identify the row in the CSV file that needs to be fixed.

Rec ID	Filename	Key Value	Data	Message	SQLERRM	Date	User
1	order_data 017 S.csv	ERROR TEST0040	TEST0040,27/08/2022, C,103Z,02/082022,101 863,12	Customer ID 103Z invalid		11-MAR-25 22:03:30.5 44000	MYDEMO
2		TEST0040	TEST0040,27/08/2022, Ship Date 02/08/2022,101 863,12	2022 must be on or later than the order date 27/08/2022		11-MAR-25 22:03:30.5 47000	MYDEMO
3		TEST0040	TEST0040,27/08/2022, Customer ID 103Z,10 0890Z,24	invalid		11-MAR-25 22:03:30.5 51000	MYDEMO
4		TEST0040	TEST0040,27/08/2022, Ship Date 02/08/2022,10 0890Z,24	must be on or later than the order date 27/08/2022		11-MAR-25 22:03:30.5 51000	MYDEMO
5		TEST0040	TEST0040,27/08/2022, Product ID 100890Z,10 0890Z,24	invalid		11-MAR-25 22:03:30.5 51000	MYDEMO
6		TEST0040	TEST0040,27/08/2022, Customer ID 103Z,10 2130,100	invalid		11-MAR-25 22:03:30.5 52000	MYDEMO
7		TEST0040	TEST0040,27/08/2022, Ship Date 02/08/2022,10 2130,100	must be on or later than the order date 27/08/2022		11-MAR-25 22:03:30.5 52000	MYDEMO
8		TEST0042	TEST0042,03/09/2022, Ship Date 02/09/2022,1018 63,9	must be on or later than the order date 03/09/2022		11-MAR-25 22:03:30.5 54000	MYDEMO
9		TEST0042	TEST0042,03/09/2022, Ship Date 02/09/2022,101,02/09/2022,2003 80,51A	must be on or later than the order date 03/09/2022		11-MAR-25 22:03:30.5 54000	MYDEMO
10		TEST0042	TEST0042,03/09/2022, Qty 51A,101,02/09/2022,2003 80,51A	invalid		11-MAR-25 22:03:30.5 70000	MYDEMO

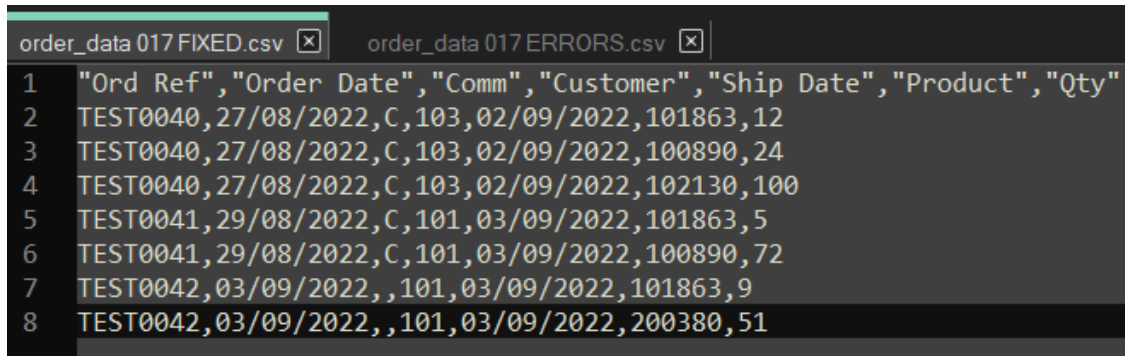
The following errors have been reported:

- TEST0040: Customer ID 103Z is invalid – the Z on the end should not be there, so the customer cannot be found.
- TEST0040: Ship Date 02/08/2022 is earlier than the order date. To correct this error, we need to change it to a date on or after the order date 27/08/2022.
- TEST0040: Product 100890Z invalid – the Z on the end should not be there, so the product cannot be found.
- TEST0042: Ship Date 02/09/2022 is earlier than the order date. To correct this error, we need to change it to a date on or after the order date 03/09/2022.
- TEST0042: Qty 51A invalid number. Just remove the A from the end.

To fix the errors:

1. Find the CSV file in the error directory DATA_IN\error.
2. Edit the file and correct the above errors.
3. Move the file to the Received directory. For this demonstration, we will also rename the file to **order_data 017 FIXED.csv**
4. Run the *import_order* program again, and check for errors.

Editing the file to correct the reported errors:



```
order_data 017 FIXED.csv [X] | order_data 017 ERRORS.csv [X]
1 "Ord Ref","Order Date","Comm","Customer","Ship Date","Product","Qty"
2 TEST0040,27/08/2022,C,103,02/09/2022,101863,12
3 TEST0040,27/08/2022,C,103,02/09/2022,100890,24
4 TEST0040,27/08/2022,C,103,02/09/2022,102130,100
5 TEST0041,29/08/2022,C,101,03/09/2022,101863,5
6 TEST0041,29/08/2022,C,101,03/09/2022,100890,72
7 TEST0042,03/09/2022,,101,03/09/2022,101863,9
8 TEST0042,03/09/2022,,101,03/09/2022,200380,51
```

Run the *import_order* process again.

Enter the password for *mydemo_connect* when prompted.

```
Enter the password for mydemo_connect: mydemo25
CSV FILE FOUND: d:\test_user_data\XEPDB1\mydemo\data\RECEIVED\order_data 017 FIXED.csv
  1 file(s) copied.

SQL*Plus: Release 21.0.0.0.0 - Production on Tue Mar 11 23:33:09 2025
Version 21.3.0.0.0

Copyright (c) 1982, 2021, Oracle. All rights reserved.

Last Successful login time: Tue Mar 11 2025 22:45:30 +00:00

Connected to:
Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0

old  2:  v_filename VARCHAR2(100) := '&1';
new  2:  v_filename VARCHAR2(100) := 'order_data 017 FIXED.csv';
Information from program at 11-MAR-25 23:03:11.299000000 Message: Order Data
Import from file: order_data 017 FIXED.csv SQLERRM is (Log mode is S)
Information from program at 11-MAR-25 23:03:11.908000000 Message: Success!
SQLERRM is (Log mode is S)

PL/SQL procedure successfully completed.

Disconnected from Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0
```

Note that this time we see the message **“Success!”**.

Run the *import_errors.sql* report again. This time, there should be no errors reported for the orders in the CSV file.

```
columns cleared
breaks cleared
computes cleared
no rows selected
```


Finally, run the orders report, *orders_param.sql*. Specify the range of orders to report as TEST0040 to TEST0042:

Bond and Pollard Limited									
=====									
Sales Order Report							Page: 1		
Order ID	Ord Ref	Date Ordered	To Ship	Comm	Total	Cust Name	Rep	Item	Product Description
Price Qty		Item Total							
622	TEST0040	27-AUG-22	02-SEP-22	C	1,882.00	103 JUST TENNIS	WARD	1	101863 SP JUNIOR RACKET
12.50	12	150.00							
622	TEST0040	27-AUG-22	02-SEP-22	C	1,882.00	103 JUST TENNIS	WARD	2	100890 ACE TENNIS NET
58.00	24	1,392.00							
622	TEST0040	27-AUG-22	02-SEP-22	C	1,882.00	103 JUST TENNIS	WARD	3	102130 RH: "GUIDE TO TENNIS
								"	3.40 100
340.00									
623	TEST0041	29-AUG-22	03-SEP-22	C	4,238.50	101 TKB SPORT SHOP	WARD	1	101863 SP JUNIOR RACKET
12.50	5	62.50							
623	TEST0041	29-AUG-22	03-SEP-22	C	4,238.50	101 TKB SPORT SHOP	WARD	2	100890 ACE TENNIS NET
58.00	72	4,176.00							
624	TEST0042	03-SEP-22	03-SEP-22		316.50	101 TKB SPORT SHOP	WARD	1	101863 SP JUNIOR RACKET
12.50	9	112.50							
624	TEST0042	03-SEP-22	03-SEP-22		316.50	101 TKB SPORT SHOP	WARD	2	200380 SB VITA SNACK-6 PACK
4.00	51	204.00							
7 rows selected.									

The orders in the CSV file have now been successfully loaded into the database.

Application Development

You should consider the following before starting to develop your applications:

Systems Design

- Choose a systems development methodology, for example the Oracle Unified Method.

Data Security

- Lock the owning schema and provide connection only users with limited privileges.
- Passwords to be encrypted, never hardcoded in scripts.
- General Data Protection Regulation GDPR.
- Segregation of duties.
- Backup & Disaster Recovery.

Change Management

- Managing changes:
 - Support Ticket / Change Request / Asset management (Samanage, SolarWinds Service Desk).
- Documenting changes to your database and application.

Software Development

- Create separate database environments for [development, testing and production](#).
- Source Control Management: Subversion, Git.
- Determine how you will [document your applications](#). At the very least you should document all of your [package public functions and procedures](#). Include comments in the package specification that describe how to use each procedure and function.
- Database design
 - [Normalization](#).
 - [Indexes and Performance](#).
 - [Surrogate or Natural keys](#).
 - [Constraints](#) to enforce business rules and data integrity.
- [Coding standards](#), based on software engineering best practice.
 - [Naming conventions](#) for your database objects and applications.
- Building [packages/libraries](#) of commonly used functions and procedures:
 - Business functions
 - Rules Packages
 - Constants
 - Error handling

Systems Development Life Cycle

The Systems Development Life Cycle (SDLC) is the process by which information systems are created or modified.

The SDLC has the following stages:

1. Planning
2. Analysis
3. Design
4. Development
5. Testing
6. Implementation
7. Maintenance

It is best to take an iterative approach to the SDLC. Do some analysis, then some design and implementation, and feed-back what you learn into the analysis stage, and so on.

Planning

Study the feasibility of creating or changing a system to meet the needs of the business. Consider the benefits of the new system versus the resources, time and cost involved.

Whilst the proposals may be technically possible, you must consider the business case, and identify potential problems. For example, it may be feasible to configure a warehouse management system to provide lot control, bringing benefits to the customer, however increased administration costs may not have been considered in the contract, leading to financial losses.

Analysis

Gather the requirements of the business, and users of the system. Consider the functionality of the new system. Analyse the requirements to select a solution that best fits the business needs.

- Research and describe the operation of the current business system
- Understand the problems of the current system, and the reasons why a new system is required
- Define and document the requirements of the new system.

Design

Create a detailed functional specification for the system. Work with the users to define their requirements and identify all the information that needs to be processed. Consider the hardware, networking and software requirements.

- Decide on a design strategy
- Pick an appropriate solution
- Produce a detailed specification
 - Data Model / Entity Relationships Diagrams
 - Process Model / Data Flow Diagrams
- Consider a strategy for support of the system once it has been implemented

Development

Design and build the database using the data model. Write technical specifications for the processes. Develop the software application programs.

- Database creation and population
- Technical Specifications
- Program Design
- Programming

Testing

Test the system to ensure it meets the defined requirements, is free from errors, robust and performs to the required standard.

The testing will be carried out by quality assurance professionals, and the end users. When the end users are satisfied with the system it can be signed off as ready to move into production, or go live.

Implementation

The new database and software applications are moved into a live “production” environment. A plan must be in place to revert to the old system if problems are encountered during implementation.

- Preparation of user documentation
- User training
- Implement system support strategy

Maintenance

The system will require support and upgrades during its lifetime. Faults may need to be fixed, or new requirements will be identified, necessitating changes to the system.

Documentation

You will need to create, update and manage the documentation for your applications.

Oracle use the following document templates for application extensions:

- [MD050 Application Extension Functional Design](#)
- [MD070 Application Extension Technical Design](#)

You will also need the following documents:

- Planning / Feasibility Study
- Requirements Analysis
- Data Model: Entity Relationship Diagrams
- Process Model: Data Flow Diagrams
- Coding Standards
- PL/SQL Libraries
- Application Programming Interfaces
- User Guide
- Test Plans
- Release Control (with instructions for how to deploy the applications into production)

Release Management

Software applications are developed, tested and deployed into product via a structured release management process. This process allows for changes to be rolled back if there are any problems, to maintain the integrity of the live system.

Deployment Environments

The following environments are used:

- DEV is the development environment where all new software is created and where changes are made to existing applications
- TEST is the environment where all the testing, including user acceptance tests, takes place.
- PROD is the live Production environment

Additional environments such as SANDBOX may be created for testing patches, upgrades and other major changes to the system.

Source Control Management

Source Control Management tools allow you track and manage changes to software.

The source code is stored in a repository, which retains a history of each version as changes are made. Developers check out code to work on, and check the new version of the source code back into the repository when it is ready to be deployed into the production system.

Modular Database Applications

The following abstract is taken from Bryn Llewellyn's White Paper: [White Papers\why use plsql whitepaper 10.pdf](#)

Large software systems must be built from modules. A module hides its implementation behind an interface that exposes its functionality. This is computer science's most famous principle.

For applications that use an Oracle Database, the database is, of course, one of the modules. The implementation details are the tables and the SQL statements that manipulate them. These are hidden behind a PL/SQL interface. This is the Thick Database paradigm: *select, insert, update, delete, merge, commit, and rollback* are issued only from database PL/SQL. Developers and end-users of applications built this way are happy with their correctness, maintainability, security, and performance. But when developers follow the noPLSQL paradigm, their applications have problems in each of these areas and end-users suffer.

Do not build your business rules into individual programs or database triggers, as this will lead to duplication, inconsistency, and make maintenance extremely difficult and time consuming. Instead, build your business processes into database packages. The packaged functions and procedures will perform all the necessary SQL actions described above. Call the packaged functions from database triggers and application programs as required.

For example, let us say you have a function that checks a customer's credit status, and returns their account balance. You may want to perform this check in several places: before accepting a new order, when printing a customer account report, and so on. Rather than having the function coded separately in different programs, it is far better to do it once in a single packaged function that can be called whenever necessary.

Client Server Architecture

Hide the implementation from the client applications

Build your business logic into packages of procedures and functions that reside in the database. Your client programs will call these stored procedures whenever they need to retrieve, insert or modify data – the client programs themselves should not communicate directly with the database.

Database Design

Normalization

Normalization is the process of organizing a database to remove redundant, duplicated data, and to group related items together to allow efficient storage, retrieval and modification of data.

Edgar F Codd invented the Relational Model for databases in the early 1970s, and together with Raymond F Boyce defined Boyce-Codd Normal Form BCNF.

The following example describes the process of converting non-relational order data stored in a spreadsheet into a set of normalized relational database tables.

The following table represents how non-normalized order data is stored in a spreadsheet.

Order ID	Order Date	Cust ID	Name	Item ID	Product ID	Description	Price	Qty	Total
601	01-MAY-86	106	Shape Up	1	200376	SB ENERGY BAR-6 PACK	2.40	1	2.40
604	15-JUN-86	106	Shape Up	1	100890	ACE TENNIS NET	58.00	3	174.00
				2	100861	ACE TENNIS RACKET II	42.00	2	84.00
				3	100860	ACE TENNIS RACKET I	44.00	10	440.00
610	07-JAN-87	101	TKB Sport Shop	1	100860	ACE TENNIS RACKET I	35.00	1	35.00
				2	100870	ACE TENNIS BALLS 3-PACK	2.80	3	8.40
				3	100890	ACE TENNIS NET	58.00	1	58.00

Note:

- An order can have one or more associated products.
- The columns containing product information are repeated.
- The product columns have been wrapped to fit on the page, but imagine them being all on a single row for each order.

First Normal Form

A relation is in first normal form if it conforms to the following rule:

- **Each row, or record, in your database table must be uniquely identified by a Primary Key, with no repeating groups of fields.**

In our example, the Product ID, Description, Price, Qty and Total columns are repeated several times for each order.

There are several problems with this organization of the data:

- A customer may order 1, 2, 10 or more products. Your database table would need multiple sets of columns to store the product information for each order.
- You will be restricted as to how many products a customer can order at one time depending on how many columns you created in your table.
- You could potentially waste a lot of space in your database if you provide columns for 10 products per order, but customers typically order one or two items each time.

To make the database comply with First Normal Form we must split the repeating product information into separate rows, each uniquely identified by a key comprising one or more columns.

ORDID (Primary Key part 1)	ORDERDATE	CUSTID	NAME	ITEMID (Primary Key part 2)	PRODID	DESCRIPTION	ACTUALPRICE	QTY	TOTAL
601	01-MAY-86	106	Shape Up	1	200376	SB ENERGY BAR-6 PACK	2.40	1	2.40
604	15-JUN-86	106	Shape Up	1	100890	ACE TENNIS NET	58.00	3	174.00
604	15-JUN-86	106	Shape Up	2	100861	ACE TENNIS RACKET II	42.00	2	84.00
604	15-JUN-86	106	Shape Up	3	100860	ACE TENNIS RACKET I	44.00	10	440.00
610	07-JAN-87	101	TKB Sport Shop	1	100860	ACE TENNIS RACKET I	35.00	1	35.00
610	07-JAN-87	101	TKB Sport Shop	2	100870	ACE TENNIS BALLS 3- PACK	2.80	3	8.40
610	07-JAN-87	101	TKB Sport Shop	3	100890	ACE TENNIS NET	58.00	1	58.00

- The columns have been renamed to follow our database object naming rules.
- Each row can be identified by a unique Primary Key comprising the *ORDID* plus *ITEMID*.

Second Normal Form

A relation is in second normal form if it conforms to the following rules:

- **It is in first normal form.**
- **Each column, or field, in the record must depend on the whole Primary Key and not just part of it.**

The columns ORDERDATE, CUSTID and NAME are duplicated in several rows, as they are dependent on only part of the Primary Key, ORDID. To resolve this problem, we need to create a new table *ORD* that will hold a single row for each unique order.

ORD

ORDID (Primary Key)	ORDERDATE	CUSTID	NAME
601	01-MAY-86	106	Shape Up
604	15-JUN-86	106	Shape Up
610	07-JAN-87	101	TKB Sport Shop

There are still a number of problems with this new table:

- The customers' names are repeated on multiple rows.
- If a customer's name changes, and not all rows are updated, there will be inconsistencies in the database.

The remaining columns will be placed in a new table called ITEM.

ITEM

ORDID (Primary Key Part 1)	ITEMID (Primary Key Part 2)	PRODID	DESCRIPTION	ACTUALPRICE	QTY	TOTAL
601	1	200376	SB ENERGY BAR-6 PACK	2.40	1	2.40
604	1	100890	ACE TENNIS NET	58.00	3	174.00
604	2	100861	ACE TENNIS RACKET II	42.00	2	84.00
604	3	100860	ACE TENNIS RACKET I	44.00	10	440.00
610	1	100860	ACE TENNIS RACKET I	35.00	1	35.00
610	2	100870	ACE TENNIS BALLS 3-PACK	2.80	3	8.40
610	3	100890	ACE TENNIS NET	58.00	1	58.00

- Each row on item is uniquely identified by a primary key consisting of ORDID plus ITEMID.
- ORDID is a Foreign Key, referencing the ORD table ORDID column.
- The product descriptions are repeated in multiple rows.

Third Normal Form

A relation is in third normal form if it conforms to the following rules:

- **The database must conform to the second normal form rules.**
- **No column must depend on any other column except the Primary Key.**

If a column is uniquely identified *through one or more other columns in addition to the primary key*, this is known as transitive dependence, and breaks the rules of third normal form.

The name of the customer associated with each order depends directly on the CUSTID, not on the Primary Key, ORDID. We need to create a new table to hold the customer information, which will eliminate the duplication of customer names.

CUSTOMER

CUSTID (Primary Key)	NAME
101	TKB Sport Shop
106	Shape Up

The ORD table now contains the following.

ORD

ORDID (Primary Key)	ORDERDATE	CUSTID
601	01-MAY-86	106
604	15-JUN-86	106
610	07-JAN-87	101

CUSTID on the ORD table is said to be a Foreign Key, linking to CUSTID on the CUSTOMER table.

The product information that is not wholly related to the primary key of Item must be moved to a new table named PRODUCT, to eliminate duplicates of description.

PRODUCT

PRODID (Primary Key)	DESCRIPTION
100860	ACE TENNIS RACKET I
100861	ACE TENNIS RACKET II
100870	ACE TENNIS BALLS 3-PACK
100890	ACE TENNIS NET
200376	SB ENERGY BAR-6 PACK

The ITEM table now contains the following columns.

ITEM

ORDID (Primary Key Part 1)	ITEMID (Primary Key Part 2)	PRODID	ACTUALPRICE	QTY
604	1	100890	58.00	3
604	2	100861	42.00	2
604	3	100860	44.00	10
610	1	100860	35.00	1
610	2	100870	2.80	3
610	3	100890	58.00	1

The column PRODID on Item is a foreign key, relating to the primary key of the PRODUCT table.

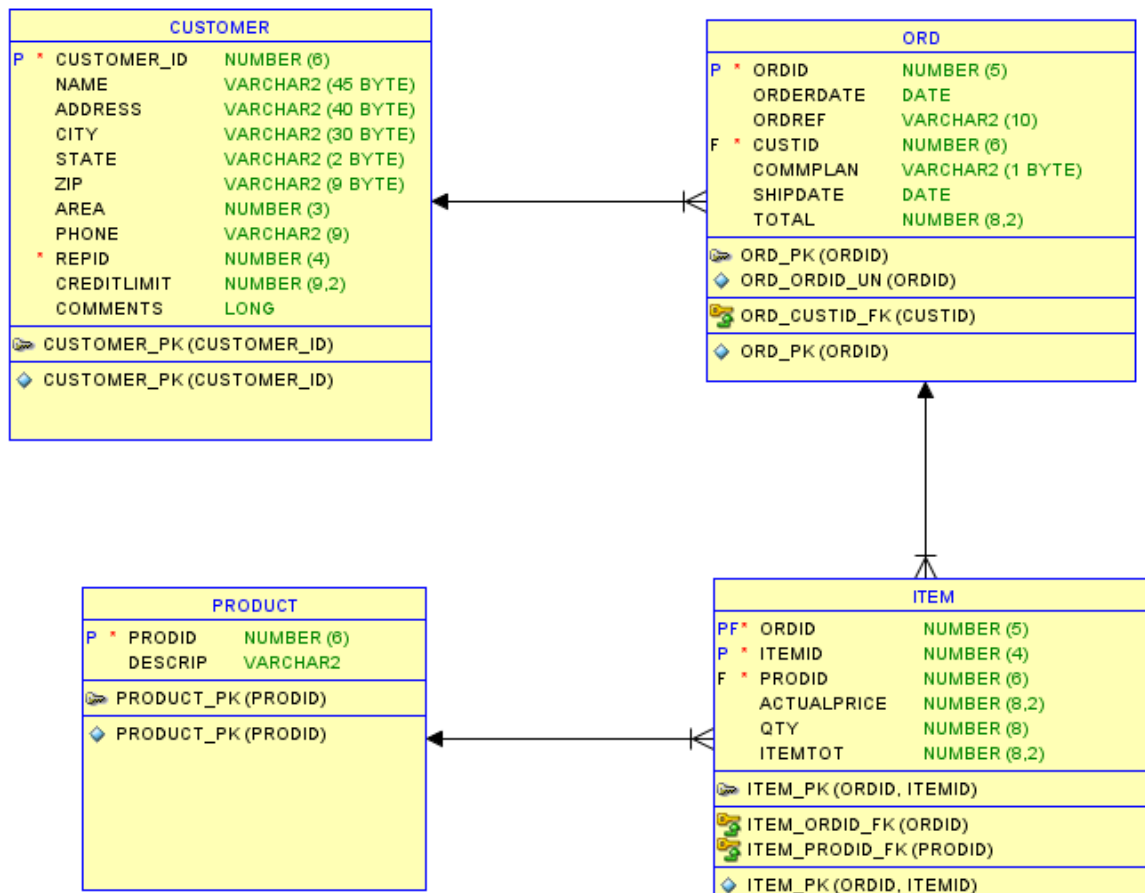
Calculated Values

The Total column in Item table is derived by multiplying QTY by the ACTUALPRICE.

If you included a total column on the order item table, you would need to recalculate and store its value every time the price or quantity was changed.

This is not a normalization problem, but you can remove the total column to make your table smaller and reduce the risk of data inconsistencies.

Data Model Diagram



Note:

- Columns have been added to show how the database can be developed further.
- Primary Keys are labelled 'P'.
- Foreign Keys are labelled 'F'.
- Indexes have been added to speed up queries.

Indexes and Performance

Avoid slow data retrieval by creating indexes for key table columns. If you don't have an index the database will scan the full table, causing serious performance issues.

Using the above example, consider a production database where the ORD table holds data for several thousand orders. If you create an index for the ORDID column then queries that need to retrieve a specific order can locate it via the index which is many times quicker than scanning the entire table looking for a match.

Surrogate vs Natural Keys

Each table should have a primary key that uniquely identifies each row. A primary key can consist of one or more columns, containing either natural or surrogate values.

Note that you should never rely on the table's ROWID as a key, as its value may change or be deleted.

Surrogate keys are preferred over natural keys. A surrogate key is a number generated by a sequence, and as such is guaranteed to be unique and will never change. Surrogate keys are numeric and can usually be stored more efficiently than natural keys, with string values.

```
CREATE TABLE mytable (  
  mytableID INTEGER GENERATED ALWAYS AS IDENTITY,  
  description VARCHAR2(50),  
  PRIMARY KEY (mytableID) );
```

A natural key has a real-world value, such as a National Insurance number, or a person's last name. You cannot absolutely guarantee that a natural key will be unique, or that it will never change. If you have a very simple, small table, you may prefer to use a natural key for simplicity if its primary key values are static and unique. For example, a table of countries may have a primary key Country_Code with values such as "F" for France, "UK" for the United Kingdom etc.

A few reasons you may prefer to use a surrogate key:

- You notice column you wish to use as a key has ascribed meanings, e.g., "2022-ABC-0001". Somebody could decide to change the structure of this value in the future.

- Your key would have multiple segments. Again, it could be decided to add more segments to the key in future, plus your SQL join statements are more complicated.
- There is a risk of duplicates, for example Last_Name would be a bad choice for a unique primary key.

The following example illustrates the use of a surrogate key.

ORD

Column Name	Data Type	Description
ORDID	NUMBER	Primary key (unique generated sequence number) NUMBER GENERATED ALWAYS AS IDENTITY
ORDREF	VARCHAR2(10)	Free form reference e.g. FRED-A001
ORDERDATE	DATE	
CUSTID	NUMBER	Foreign key references CUSTOMER.CUSTID

ITEM

Column Name	Data Type	Description
ORDERITEMID	NUMBER	Primary key (unique generated sequence number) NUMBER GENERATED ALWAYS AS IDENTITY
ORDID	NUMBER	Foreign key references ORD.ORDID
ITEMID	NUMBER	Sequential line number 1,2,3 etc.
PRODID	NUMBER	Foreign key references PRODUCT.PRODID
ACTUALPRICE	NUMBER	
QTY	NUMBER	

- ORDREF is not suitable for use as a key, as it contains user defined values that could easily be duplicated, or need to change.
- The primary key on each table consists of a single column containing a sequentially generated number.
- The Primary Key ORDERITEMID uniquely identifies each row on ITEM, and is a sequentially generated number with no direct relationship to the ORD table.
- The Foreign Key ORDID links the rows in ITEM with their corresponding ORD row.
- If you need a foreign key to ITEM on another table, you can use the ORDERITEMID column, instead of having to add two separate columns ORDID and ITEMID.

Constraints

Use Database Constraints to enforce business rules for table columns.

Database constraints are stored in the data dictionary rather than in application code, and provide a simple to code, centralized method for enforcing business rules.

All applications that access the database must follow the rules defined in the data dictionary. If you build the rules into each application program you need to do a lot more work, the complexity is increased and inconsistencies and errors are more likely.

Examples:

Constraint Type	Description
PRIMARY KEY	A column or group of columns that uniquely identifies a row in the table. No column in the primary key may be null, unlike a UNIQUE constraint. It may be preferable to use a surrogate key value instead of natural keys such as a name. Not mandatory.
FOREIGN KEY	Create a relationship between two tables. For example, ORDID on ITEM references ORDID on ORD. Prevents a row from being created on ITEM without a corresponding ORD (orphaned row). Improve performance when reading data. Insert/modify/delete slowed down, but maintains data integrity. You may omit foreign keys on audit or logging tables where you never want to delete the data in cascade.
UNIQUE	The data stored in the specified column, or group of columns, must be unique for the rows in the table. Columns may have null values.
NOT NULL	The column must contain a non-null value.
CHECK	The column must not contain a value that violates the specified condition, for example SALARY > 0

Coding Standards

Golden Rules for Software Development

1. Modular programming: separate the functionality of your programs into independent modules, or packages containing functions and procedures.
2. Each module should hide its implementation behind an interface that exposes its functionality. This is a key principle of software engineering.
3. Code for
 - a. Correctness
 - b. Maintainability
 - c. Security
 - d. Performance
4. Never repeat code. Write the code for each business function once, and call it from database triggers and application programs.
5. Do not hardcode literal values.
6. No GOTOs or unconditional branching. Every loop should have one way in, and one way out

Naming Conventions

The following rules apply to database objects (tables, views, procedures) and PL/SQL variables:

- Maximum length 30 characters
- First character must be a letter
- Valid characters: letter, numeral, \$, _, #
- PL/SQL is not case sensitive to identifiers

Note that the object names are stored in the database in uppercase, and are not case sensitive unless you surround them with double quotes.

File Names

PL/SQL Package Specification	<packagename>.pls
PL/SQL Package Body	<packagename>.b.pls
PL/SQL library	<libraryname>.pll .plx .pld

Database Objects

Table Names	Singular description for example PRODUCT. For tables that resolve many-to-many relationships combine the names of each table
Views	<tablename>_<criteria>_V
Index Primary	<tablename>_IDX
Index Other	<tablename>_<column>_IDX
Constraint (Primary Key)	<tablename>_PK
Constraint (Foreign Key)	<table from>_<table to>_FK
Constraint (Not Null)	<tablename>_<column>_NN
Constraint (Check)	<tablename>_<column>_CHK
Constraint (Unique)	<tablename>_<column>_UN
Sequences	<tablename>_<column>_SEQ
Triggers	<tablename>_T[n]

PL/SQL Variables and Identifiers

Prefixes

l_	Local variables
g_	Global variables
v_	Variable
c_	Constant
p_	Parameter
t_	User defined type
tb_	PL/SQL table
r_	PL/SQL record

Suffixes

_cur	Cursor
_IN	Parameter Input

Coding Style

Make sure that your code is easily readable, and that its intended purpose is clear. Creating simple, elegant code is an art form, but it is well worth doing, as it makes future maintenance so much easier. Your code should practically document itself, but that is not to say that you should do away with comments and documentation altogether!

- Include useful comments that clearly explain what the program does, but only where necessary.
- Do not include comments for lines of code that are self-explanatory.
- Indent your code to make it readable, using spaces. Never use tabs as they may vary in width in different environments, messing up the formatting.

PL/SQL Programming Tips

1. Do not put your code inside database triggers, instead call packaged functions and procedures from the triggers.
2. Thick database paradigm: SQL that manipulates data (select, insert, update, delete, merge, commit, rollback) should only be issued from a PL/SQL packaged function or procedure that resides in the database. Do not put this SQL code inside your application programs.
3. Create an error/exception handling package, and call this from your programs instead of hardcoding error messages in multiple places.
4. Avoid explicitly declared datatypes for your variables, instead use %TYPE to reference database columns.
5. For derived values, declare your own application types (SUBTYPE) in a Rules Package.
6. Define types that are not derived from table columns in a package, for example: *plsql_constants* so that they're all in one place, and easy to change if required.
7. Avoid using *commit* as it is hardcoding, and compromises flexibility around testing
8. Implicit cursors may be faster than explicit cursors
9. Bulk Collect should be used in preference to cursor for loops
10. Use TOAD (Tool for Oracle Application Developers) PL/SQL Code Expert Review features. Available in version 8 onward.
11. Follow Oracle's application development and customization standards

Packages

By creating a specific object for each business process or function, you can maintain the logic in one place and re-use it many times. This is easier to maintain and saves a great deal of effort.

- Collect related procedures and functions together.
- Restrict public access to application logic.

Functions

Do not hardcode business rules into your code – hide the implementation by creating a function to perform the necessary process.

Example:

Hardcoded rule to derive an employee's full name.

```
SELECT employee.first_name || ' ' || employee.last_name
      INTO l_full_name
...

```

Instead, create a function to derive the full name.

```
SELECT employee_rp.fullname(first_name, last_name)
      INTO l_full_name
...

```

Here we call the function *employee_rp.fullname*, which takes *first_name* and *last_name* as parameters and returns the full name, formatted as required. The function is stored in a package named *employee_rp*, which is the rules package for handling employee data.

You would just need to alter the *employee_rp.fullname* function to implement business changes, rather than seeking out, and amending each instance of code in many different programs.

Data Typing

Do not hard-code data-types in your programmes that will potentially cause future problems.

For example, a variable to hold a person's name, set to a fixed length of characters. What if the associated column in the database is altered in the future and its new length exceeds your variable's declared length?

Instead, base your variables on database columns wherever possible so that your programs stay in line with the underlying database structure.

```
V_employee_name    varchar2(100);           -- WRONG
V_employee_name    emp.employee_name%TYPE; -- BETTER!
```

Performance

Bulk Collect versus Cursor For Loops

Do not use a Cursor For Loop for a single row query.

With Oracle 8i and above, replace cursor FOR loops with the much faster BULK COLLECT query.

NB: From Oracle version 9i onward you can bulk collect into row type structures instead of individual tables for each field, but you cannot currently reference the individual fields within the FORALL process.

Bulk Collect avoids context switch between SQL engine and PL/SQL engine that embedded SQL statements cause – this is slow.

Take great care with this – Bulk Collect sacrifices memory for speed. All your collected data is stored in memory, and you then use FORALL to process the data. If you have tables with millions of rows Bulk Collect could cause severe memory problems.

To avoid this problem, code the Bulk Collect to work in manageable batches of rows, for example groups of 200.

Cursors

Implicit cursors are often faster than explicit cursors.

```
SELECT employee.last_name  
      INTO l_last_name  
FROM employees  
WHERE employee_id = emp_id_in;
```

Instead of:

```
DECLARE  
  
CURSOR c_emp IS  
      SELECT last_name  
      FROM employees  
      WHERE employee_id = emp_id_in;  
  
Rec_emp c_emp%ROWTYPE;  
BEGIN  
      OPEN c_emp;  
      FETCH c_emp INTO rec_emp;  
      IF c_emp%FOUND THEN  
          ...  
END
```