DBMS Lab Reference Manual – SQL & PL/SQL

# 1. Overview

This document is a comprehensive reference for DBMS Lab exam preparation. It covers SQL, PL/SQL, constraints, transactions, cursors, procedures, functions, triggers, and common schemas. All examples are Oracle-compatible.

# 2. SQL Command Categories

DDL – Data Definition Language: CREATE, ALTER, DROP, TRUNCATE, RENAME  
DML – Data Manipulation Language: INSERT, UPDATE, DELETE  
DCL – Data Control Language: GRANT, REVOKE  
DQL – Data Query Language: SELECT  
TCL – Transaction Control Language: COMMIT, ROLLBACK, SAVEPOINT

# 3. Common Schemas

## 3.1 Employee–Department Schema

CREATE TABLE Department (  
 dept\_no NUMBER(3) PRIMARY KEY,  
 dept\_name VARCHAR2(20) UNIQUE NOT NULL,  
 location VARCHAR2(20)  
);  
  
CREATE TABLE Employee (  
 emp\_no NUMBER(4) PRIMARY KEY,  
 emp\_name VARCHAR2(30) NOT NULL,  
 gender CHAR(1) CHECK (gender IN ('M','F')),  
 salary NUMBER(10,2) CHECK (salary > 0),  
 hire\_date DATE DEFAULT SYSDATE,  
 dept\_no NUMBER(3),  
 CONSTRAINT fk\_dept FOREIGN KEY (dept\_no)  
 REFERENCES Department(dept\_no)  
);

## 3.2 Bank Schema

CREATE TABLE Branch (  
 branch\_name VARCHAR2(15) PRIMARY KEY,  
 branch\_city VARCHAR2(20),  
 assets NUMBER(12,2)  
);  
  
CREATE TABLE Account (  
 account\_number NUMBER(10) PRIMARY KEY,  
 branch\_name VARCHAR2(15) REFERENCES Branch(branch\_name),  
 balance NUMBER(10,2)  
);  
  
CREATE TABLE Customer (  
 customer\_name VARCHAR2(25) PRIMARY KEY,  
 customer\_street VARCHAR2(25),  
 customer\_city VARCHAR2(20)  
);  
  
CREATE TABLE Depositor (  
 customer\_name VARCHAR2(25) REFERENCES Customer(customer\_name),  
 account\_number NUMBER(10) REFERENCES Account(account\_number),  
 PRIMARY KEY (customer\_name, account\_number)  
);  
  
CREATE TABLE Loan (  
 loan\_number NUMBER(10) PRIMARY KEY,  
 branch\_name VARCHAR2(15) REFERENCES Branch(branch\_name),  
 amount NUMBER(10,2)  
);  
  
CREATE TABLE Borrower (  
 customer\_name VARCHAR2(25) REFERENCES Customer(customer\_name),  
 loan\_number NUMBER(10) REFERENCES Loan(loan\_number),  
 PRIMARY KEY (customer\_name, loan\_number)  
);

## 3.3 University Schema

CREATE TABLE Department (  
 dept\_name VARCHAR2(20) PRIMARY KEY,  
 building VARCHAR2(15),  
 budget NUMBER(10,2)  
);  
  
CREATE TABLE Instructor (  
 id NUMBER(5) PRIMARY KEY,  
 name VARCHAR2(25),  
 dept\_name VARCHAR2(20) REFERENCES Department(dept\_name),  
 salary NUMBER(8,2)  
);  
  
CREATE TABLE Student (  
 id NUMBER(5) PRIMARY KEY,  
 name VARCHAR2(25),  
 dept\_name VARCHAR2(20) REFERENCES Department(dept\_name),  
 tot\_cred NUMBER(3)  
);  
  
CREATE TABLE Course (  
 course\_id VARCHAR2(10) PRIMARY KEY,  
 title VARCHAR2(30),  
 dept\_name VARCHAR2(20) REFERENCES Department(dept\_name),  
 credits NUMBER(2)  
);  
  
CREATE TABLE Takes (  
 id NUMBER(5) REFERENCES Student(id),  
 course\_id VARCHAR2(10) REFERENCES Course(course\_id),  
 semester VARCHAR2(10),  
 year NUMBER(4),  
 grade CHAR(2),  
 PRIMARY KEY (id, course\_id, semester, year)  
);

# 4. SQL Query Examples

-- Select all employees  
SELECT \* FROM Employee;  
  
-- Find employees with salary > 50000  
SELECT emp\_name, salary FROM Employee WHERE salary > 50000;  
  
-- Find departments with average salary > 60000  
SELECT dept\_no, AVG(salary) AS avg\_sal  
FROM Employee  
GROUP BY dept\_no  
HAVING AVG(salary) > 60000;  
  
-- Join example  
SELECT e.emp\_name, d.dept\_name  
FROM Employee e  
JOIN Department d ON e.dept\_no = d.dept\_no;

# 5. PL/SQL Basics

Structure of a PL/SQL block:  
  
SET SERVEROUTPUT ON;  
DECLARE  
 variable\_name datatype;  
BEGIN  
 -- executable statements  
 DBMS\_OUTPUT.PUT\_LINE('Message');  
EXCEPTION  
 WHEN OTHERS THEN  
 DBMS\_OUTPUT.PUT\_LINE('Error');  
END;  
/  
  
Example: Print Hello World  
DECLARE  
 msg VARCHAR2(20) := 'Hello World';  
BEGIN  
 DBMS\_OUTPUT.PUT\_LINE(msg);  
END;  
/

# 6. Control Structures

IF-THEN-ELSE  
IF condition THEN  
 statements;  
ELSIF condition THEN  
 statements;  
ELSE  
 statements;  
END IF;  
  
LOOP  
 statements;  
 EXIT WHEN condition;  
END LOOP;  
  
WHILE condition LOOP  
 statements;  
END LOOP;  
  
FOR i IN 1..5 LOOP  
 DBMS\_OUTPUT.PUT\_LINE(i);  
END LOOP;

# 7. Cursors

Explicit Cursor Example:  
DECLARE  
 CURSOR c1 IS SELECT emp\_name, salary FROM Employee;  
 v\_emp Employee.emp\_name%TYPE;  
 v\_sal Employee.salary%TYPE;  
BEGIN  
 OPEN c1;  
 LOOP  
 FETCH c1 INTO v\_emp, v\_sal;  
 EXIT WHEN c1%NOTFOUND;  
 DBMS\_OUTPUT.PUT\_LINE(v\_emp || ' earns ' || v\_sal);  
 END LOOP;  
 CLOSE c1;  
END;  
/  
  
Cursor FOR Loop:  
FOR rec IN (SELECT \* FROM Employee WHERE salary > 50000) LOOP  
 DBMS\_OUTPUT.PUT\_LINE(rec.emp\_name || ' - ' || rec.salary);  
END LOOP;

# 8. Procedures and Functions

Procedure:  
CREATE OR REPLACE PROCEDURE raise\_salary(p\_id IN NUMBER, p\_amt IN NUMBER) AS  
BEGIN  
 UPDATE Employee SET salary = salary + p\_amt WHERE emp\_no = p\_id;  
END;  
/  
  
Function:  
CREATE OR REPLACE FUNCTION avg\_salary RETURN NUMBER AS  
 v\_avg NUMBER;  
BEGIN  
 SELECT AVG(salary) INTO v\_avg FROM Employee;  
 RETURN v\_avg;  
END;  
/

# 9. Triggers

CREATE TABLE Product (  
 prod\_id VARCHAR2(10) PRIMARY KEY,  
 prod\_name VARCHAR2(30),  
 price NUMBER(10,2)  
);  
  
CREATE TABLE Price\_Log (  
 prod\_id VARCHAR2(10),  
 old\_price NUMBER(10,2),  
 new\_price NUMBER(10,2),  
 change\_date DATE  
);  
  
CREATE OR REPLACE TRIGGER trg\_price\_update  
BEFORE UPDATE OF price ON Product  
FOR EACH ROW  
BEGIN  
 INSERT INTO Price\_Log VALUES(:OLD.prod\_id, :OLD.price, :NEW.price, SYSDATE);  
END;  
/

# 10. Transactions

COMMIT; -- Save changes permanently  
ROLLBACK; -- Undo uncommitted changes  
SAVEPOINT A; -- Mark a transaction point  
ROLLBACK TO A;-- Undo changes after savepoint  
  
Example:  
UPDATE Employee SET salary = salary + 2000 WHERE dept\_no = 10;  
SAVEPOINT S1;  
DELETE FROM Employee WHERE dept\_no = 30;  
ROLLBACK TO S1;  
COMMIT;

# 11. Viva and Quick Reference

- Primary Key: Uniquely identifies a record.  
- Foreign Key: References another table's primary key.  
- View: A virtual table.  
- Cursor: Pointer to query result set in memory.  
- Trigger: Auto executes when a DML event occurs.  
- Procedure: Performs action without returning value.  
- Function: Returns a value.  
- Commit: Save changes.  
- Rollback: Undo changes.