**Department of AIT-Computer Science & Engineering**

SUBJECT: **Advanced Database Management**

**(20CSP-436)**

**B.E. VII Semester**

**(Branch: CSE-AIML)**



**LAB MANUAL**

**Prepared By: Er. Lata Gupta**

**Chandigarh University Gharuan, Mohali**

# Syllabus

|  |  |  |  |
| --- | --- | --- | --- |
| **Institute/Department** | UNIVERSITY INSTITUTEOF ENGINEERING (UIE) | **Program** | **Bachelor of Engineering in Computer Science and Engineering (Hons.) IBM - artificial intelligence & machine learning(cs210)** |
| **Master Subject CoordinatorName:** | Vishwa Deepak | **Master Subject Coordinator E-Code**: | **E12867** |
| **Course Name** | Advanced Database Management Lab | **Course Code** | **CSP-436** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Lectu re | Tutorial | Practical | Self Study | Credit | Subject Type |
| 0 | 0 | 2 | 0 | 1.0 | P |

|  |  |  |  |
| --- | --- | --- | --- |
| Course Type | Course Category | Mode of Assessment | Mode of Delivery |
| Program Core | Graded (GR) | Practical Examination (PRAC) | Practical (PRAC) |

|  |  |
| --- | --- |
| Mission of the University | Providing world class infrastructure, renowned academicians and ideal environment for Research, Innovation, Consultancy and Entrepreneurship relevant to the society.  Offering programs & courses in consonance with National policies for nation building and meeting global challenges.  Designing Curriculum to match international standards, needs of Industry, civil society and for inculcation of traits of Creative Thinking and Critical Analysis as well as Human and Ethical values.  Ensuring students delight by meeting their aspirations through blended learning, corporate mentoring, professional grooming, flexiblecurriculum and healthy atmosphere based on co- curricular and extra-curricular activities.  Creating a scientific, transparent and objective examination/evaluation system to ensure an ideal certification.  Establishing strategic relationships with leading National and International corporates and universities for academic as well as researchcollaborations.  Contributing for creation of healthy, vibrant and sustainable society by involving in Institutional Social Responsibility (ISR) activities likerural development, welfare of senior citizens, women empowerment, community service, health and hygiene awareness and environmentalprotection |
| Vision of the University | To be globally recognized as a Centre of Excellence for Research, Innovation, Entrepreneurship and disseminating knowledge by providing inspirational learning to produce professional leaders for serving the society. |

|  |  |
| --- | --- |
| Mission of the Department | M1: To provide relevant, rigorous and contemporary curriculum and aligned assessment system to ensure effective learningoutcomes for engineering technologies.  M2: To provide platform for industry engagement aimed at providing hands-on training on advanced technological and businessskills to our students.  M3: To provide opportunities for collaborative, interdisciplinary and cutting-edge research aimed at developing solutions to real lifeproblems  M4: To imbibe quest for innovation, continuous learning and zeal to pursue excellence through hard work and problem-solvingapproach  M5: To foster skills of leadership, management, communication, team spirit and strong professional ethics in all academic andsocietal endeavors of our students |
| Vision of the Department | To be recognized as a center of excellence for Computer Science & Engineering education and research, through effective teaching practices, hands-on training on cutting edge computing technologies and excellence in innovation, for creating globallyaware competent professionals with strong work ethics whom would be proficient in implementing modern technology solutions  and shall have entrepreneurial zeal to solve problems of organizations and society at large. |

|  |  |
| --- | --- |
| **Program Educational Objectives(PEOs)** | |
| PEO 1 | To be able to explore areas of research, technology application & innovation and make a positive impact in different types ofinstitutional settings such as corporate entities, government bodies,  NGOs, inter-government organizations, & start-ups. |
| PEO 2 | To be able to design, and implement technology and computing solutions to the organizational problems, effectively deploy knowledge of engineering principles, demonstrate critical thinking skills&make the intellectual connections between quantitativeand qualitative tools, theories and context to solve the organizational problems |
| PEO 3 | To be able to work with, lead & engage big and small teams comprising diverse people in terms of gender, nationality, region,language, culture & beliefs. To understand stated and unstated  differences of views, beliefs & customs in diverse & inter disciplinary team settings |
| PEO 4 | To be able to continuously learn and update one’s knowledge, engage in lifelong learning habits and acquire latest knowledge toperform in current work settings |
| PEO 5 | To continuously strive for justice, ethics, equality, honesty, and integrity both in personal and professional pursuits. Able tounderstand and conduct in a way that is responsible and respectful. |

|  |  |
| --- | --- |
| **Program Specific OutComes(PSOs)** | |
| PSO1 | The graduating student will be able to make valuable contributions in design, development, and production of computer science and related engineering applications in the areas of Artificial intelligence and Machine learning |
| PSO2 | The graduating student will be able to use and deploy the latest software tools and technologies related to Artificial intelligence and Machine learning. |
| PSO3 | The graduating student will be able to practice as an engineer / researcher in the evolving field of AI and ML and its allied application domains by employing project development skills learnt throughout the program |

|  |  |
| --- | --- |
| **Program OutComes(POs)** | |
| PO1 | Engineering Knowledge: Apply knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems. |
| PO2 | Problem Analysis: Identify, formulate, research literature and analyze complex engineering problems reaching Substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences. |
| PO3 | Design/ Development of Solutions: Design solutions for complex engineering problems and design system components or processes that meet specified needs with appropriate consideration for public health  and safety, cultural, societal and environmental considerations |
| PO4 | Conduct investigations of complex problems using research-based knowledge and research methods including design of experiments, analysis and interpretation of data and synthesis of information to provide valid conclusions. |
| PO5 | Modern Tool Usage: Create, select and apply appropriate techniques, resources and modern engineering and IT tools Including prediction and modeling to complex engineering activities with an understanding of the limitations |
| PO6 | The Engineer and Society: Apply reasoning informed by contextual knowledge to assess societal, health, safety, legal and Cultural issues and the consequent responsibilities relevant to professional engineering practice. |
| PO7 | Environment and Sustainability: Understand the impact of professional engineering solutions in societal and Environmental contexts and demonstrate knowledge of and need for sustainable development. |
| PO8 | Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of engineering practice. |
| PO9 | Individual and Team Work: Function effectively as an individual, and as a member or leader in diverse teams and in multi-disciplinary settings. |
| PO10 | Communication: Communicate effectively on complex engineering activities with the engineering  community and withsociety at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations and give and receive clear instructions. |
| PO11 | Project Management and Finance: Demonstrate knowledge and understanding of engineering and managementprinciples and apply these to one's own work, as a member and leader in a team, to  manage projects and in multidisciplinary environments |
| PO12 | Life-long Learning: Recognize the need for and have the preparation and ability to Engage in independent and life- long learning in the broadest context of technological Change. |

**Course Objective**

|  |  |
| --- | --- |
| **Course Objective** | |
| 1 | Develop understanding the advancement in SQL |
| 2 | Demonstrate methods to apply SQL using programming construct PL/SQL |
| 3 | Use and application of normalization techniques and implement the concept of triggers. |

**Course Outcome**

|  |  |
| --- | --- |
| **Course Outcome** | |
| **CO1** | Execute and apply advanced level SQL queries |
| **CO2** | Create views of data and Implement transaction control using locks. |
| **CO3** | Apply the concept of PL/SQL programming for control structures and cursors |
| **CO4** | Apply the concept of PL/SQL programming for performing operations on packages and triggers |
| **CO5** | Analyze and apply advanced normalization technique and redundant functional dependencies |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Course Outcome** | **PO1** | **PO2** | **PO3** | **PO4** | **PO5** | **PO6** | **PO7** | **PO8** | **PO9** | **PO10** | **PO11** | **PO12** | **PSO1** | **PSO2** | **PSO3** |
| **CO1** | 1 | 1 | 3 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| **CO2** | 1 | 1 | 3 | 3 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| **CO3** | 1 | 1 | 3 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| **CO4** | 1 | 1 | 3 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| **CO5** | 1 | 1 | 2 | 2 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |

# Table of Contents

|  |  |  |
| --- | --- | --- |
| **S.NO.** | **Experiment** | Mapped CO |
| **Unit-I** | | |
| 1 | Experiment 1.1 To study and understand TCL commands in SQL. | CO1 |
| 2 | Experiment 1.2 To implement DCL commands in SQL. | CO1 |
| 3 | Experiment 1.3 To analyze and create locks and different types of locks. | CO2 |
| 4 | Experiment 1.4 To create and perform queries on sequences, synonyms and views. | CO2 |
| **Unit 2** | | |
| 5 | Experiment 2.1 To Implement PL/SQL programming using Control Structures. | CO3 |
| 6 | Experiment 2.2 To Implement Pl/SQL programming using Cursors. | CO3 |
| 7 | Experiment 2.3 To Implement Pl/SQL programming using exception handling. | CO3,CO4 |
| **Unit 3** | | |
| 8 | Experiment 3.1 Perform various operations on Packages and Triggers. | CO4 |
| 9 | Experiment 3.2 Prepare a Case Study explaining the need for converting tables to fourth and fifth Normal forms. | CO5 |
| 10 | Experiment 3.3 Prepare a Case Study on Functional dependency with examples for redundant functional dependencies. | CO5 |

**Experiment No. 1**

### Mapped Course Outcomes- CO1

**CO1:** Execute and apply advanced level SQL queries.

**AIM:** To study and understand TCL commands in SQL.

**Objective:** Create a database for Company, add employee details into it as a part of transaction. Analyse the role of Commit, Rollback and Save point.

**Apparatus Required:**

A computer system with Oracle 10g installed.

**Theory related to experiment:**

TCL stands for **Transaction control language**.

A single unit of work in a database is formed after the consecutive execution of commands is known as a transaction.

There are certain commands present in SQL known as TCL commands that help the user manage the transactions that take place in a database.

**COMMIT, ROLLBACK** and **SAVEPOINT** are the most commonly used TCL commands in SQL.

### COMMIT:

COMMIT command in SQL is used to save all the transaction-related changes permanently to the disk. Whenever DDL commands such as INSERT, UPDATE and DELETE are used, the changes made by these commands are permanent only after closing the current session. So before closing the session, one can easily roll back the changes made by the DDL commands. Hence, if we want the changes to be saved permanently to the disk without closing the session, we will use the commit command.

### Syntax:

COMMIT;

**Program Code:**

***CREATE TABLE*** *t\_Company(ID* ***INT****, Company\_Name* ***VARCHAR****(40), Number\_Of\_Empl oyees* ***INT****, Number\_Of\_Project\_Managers* ***INT****, EmailID* ***VARCHAR****(40));*

*BEGIN* ***TRANSACTION****;*

***INSERT INTO*** *t\_Company(ID, Company\_Name, Number\_Of\_Employees, Number\_Of\_Proj ect\_Managers, EmailID)* ***VALUES****(1, "XYZ Public Company", 1000, 12, "*[*xyz15@gmail.com*](mailto:xyz15@gmail.com) *"), (2, "ABCDPublic Company", 800, 35,* [*"abc*](mailto:abcd25@gmail.com)*d*[*25@gmail.com*](mailto:abcd25@gmail.com)*"), (3, "Agra Public Company*

*", 1200, 10,* [*"agra101@gmail.com")*](mailto:agra101@gmail.com)*, (4, "imperial Company", 1110, 40,*[*"imperial@gmail.co*](mailto:imperial@gmail.co) *m”);*

***COMMIT****;*

*Select \* from t\_Company;*

***Output :***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***ID*** | ***Company\_Name*** | ***Number\_of***  ***\_Employees*** | ***Number\_of\_Project\_Ma nagers*** | ***EmailID*** |
| ***1*** | *XYZ Public Comp any* | *1000* | *80* | [*xyz15@gmail.com*](mailto:xyz15@gmail.com) *")* |
| ***2*** | *ABCDPublic Co mpany* | *800* | *35* | [*abc*](mailto:abcd25@gmail.co)[*d25@gmail.co*](mailto:d25@gmail.co) *m* |
| ***3*** | *Agra Public Com pany* | *1200* | *30* | [*agra101@gmail.c*](mailto:agra101@gmail.c) *om* |
| ***4*** | *imperial Compan y* | *1110* | *40* | [*imperial@gmail.c*](mailto:imperial@gmail.c) *om* |

*Database Table will be created using name t\_Company and data will be inserted. Commit Command Will save the data permanently.*

**SAVEPOINT operation:**

A SAVEPOINT is a point in a transaction when you can roll the transaction back to a certain point without rolling back the entire transaction.

### Syntax:

SAVEPOINT SAVEPOINT\_NAME;

### ROLLBACK operation:

The ROLLBACK command is the transactional command used to undo transactions that have not already been saved to the database. This command can only be used to undo transactions since the last COMMIT or ROLLBACK command was issued.

### Syntax:

ROLLBACK;

**Program Code:**

*BEGIN* ***TRANSACTION****;*

*SAVEPOINT Insertion;*

***UPDATE*** *t\_Company* ***SET*** *Number\_Of\_Employees = 9050* ***WHERE*** *ID = 5; SAVEPOINT Updation;*

***ROLLBACK TO*** *Insertion;*

***SELECT*** *\*****FROM*** *t\_Company;*

***Output:***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***ID*** | ***Company\_Name*** | ***Number\_of***  ***\_Employees*** | ***Number\_of\_ Project\_Ma nagers*** | ***EmailID*** |
| ***1*** | *XYZ Public Company* | *1000* | *80* | [*xyz15@gmail.com*](mailto:xyz15@gmail.com) *")* |
| ***2*** | *ABCDPublic Company* | *800* | *35* | [*abc*](mailto:abcd25@gmail.co)[*d25@gmail.co*](mailto:d25@gmail.co) *m* |
| ***3*** | *Agra Public Company* | *1200* | *30* | [*agra101@gmail.c*](mailto:agra101@gmail.c) *om* |
| ***4*** | *imperial Company* | *1110* | *40* | [*imperial@gmail.c*](mailto:imperial@gmail.c) *om* |

**Text Book:**

1. Sql/ Pl/SQL, Bayross, Ivan

**Reference Book:**

1. Database System Concepts, Henry F. Korth, S. Sudarshan

**Web Reference:**

1. https:/[/www.javatpoint.com/t](http://www.javatpoint.com/tcl-commands-in-)c[l-commands-in-](http://www.javatpoint.com/tcl-commands-in-) sql#:~:text=In%20SQL%2C%20TCL%20stands%20for,take%20place%20in%20a%2 0database.
2. https:/[/www.ge](http://www.geeksforgeeks.org/tcl-full-)e[ksforgeeks.org/tcl-full-](http://www.geeksforgeeks.org/tcl-full-) form/#:~:text=TCL%20stands%20for%20Transaction%20Control,the%20data%20sto red%20in%20DBMS.

**Video:**

<https://youtu.be/W0dJ8e_IIZI>

### Viva Question:

1. What is the difference between rollback and rollback to ?
2. What is role of Savepoint?
3. Can we have multiple savepoints?
4. What is transaction Control language?
5. Which SQL command is used for initiating transaction?
6. What is the role of Commit command?
7. Why do we need to commit the transaction?
8. Which commands is to revert back to the previous consistent state?
9. What do you understand by partially committed state?
10. How can be perform recovery to the previous consistent state?

# Experiment No. 2

### Mapped Course Outcomes- CO1

**CO1:** Execute and apply advanced level SQL queries

**AIM:** To implement DCL commands in SQL.

**Objective:** Use a Database crated in Experiment 1 and Grant permission to user at various level.

**Apparatus Required:**

A computer system with Oracle 10g installed.

**Theory related to experiment:**

GRANT CREATE SESSION TO username;

Grant all privilege to a User:

sysdba is a set of priviliges which has all the permissions in it. So if we want to provide all the privileges to any user, we can simply grant them the sysdba permission.

GRANT sysdba TO username Allow a User to create session:

When we create a user in SQL, it is not even allowed to login and create a session until and unless proper permissions/priviliges are granted to the user. Following command can be used to grant the session creating priviliges.

GRANT CREATE SESSION TO username;

Allow a User to create table

To allow a user to create tables in the database, we can use the below command, GRANT CREATE TABLE TO username;

Provide user with space on tablespace to store table

Allowing a user to create table is not enough to start storing data in that table. We also must provide the user with priviliges to use the available tablespace for their table and data.

ALTER USER username QUOTA UNLIMITED ON SYSTEM;

The above command will alter the user details and will provide it access to unlimited tablespace on system.

Grant permission to create any table

Sometimes user is restricted from creating come tables with names which are reserved for system tables. But we can grant privileges to a user to create any table using the below command,

GRANT CREATE ANY TABLE TO username

Grant permission to drop any table

As the title suggests, if you want to allow user to drop any table from the database, then grant this privilege to the user,

GRANT DROP ANY TABLE TO username

To take back Permissions

And, if you want to take back the privileges from any user, use the REVOKE command.

### Program Code:

GRANT CREATE SESSION TO username; GRANT CREATE TABLE TO username;

ALTER USER username QUOTA UNLIMITED ON SYSTEM;

GRANT CREATE ANY TABLE TO username GRANT DROP ANY TABLE TO username

***Output:***

***SQL query successfully executed. However, the result set is empty.***

*Permission will be Granted to Users on various level.*

**Revoke Command:** The Oracle REVOKE statement revokes system and object privileges from a user.

The basic syntax of the Oracle REVOKE statement:

REVOKE {system\_privilege | object\_privilege } FROM user;

To revoke all system privileges from a user, you can use the following statement: REVOKE ALL PRIVILEGES FROM user;

### Program Code:

REVOKE SELECT, INSERT, UPDATE, DELETE ON ot.customers

FROM bob;

***SQL query successfully executed. However, the result set is empty.***

### Text Book:

1. Sql/ Pl/SQL, Bayross, Ivan

### Reference Book:

1. Database System Concepts, Henry F. Korth, S. Sudarshan

### Viva Questions:

1. What is purpose of GRANT Command?
2. What is different level of permission that can be granted?
3. How grant command benefits database?
4. What is revoke command?
5. How grant and revoke are different?
6. How to prevent user from accessing a table?
7. How to allow user to create a database?
8. Why do we grant and revokes permissions?

# Experiment No. 3

**Mapped Course Outcomes- CO2**

**CO2:** Create views of data and Implement transaction control using locks.

**AIM:** To analyse and create locks and different types of locks.

**Objective:** Lock a flight table in share mode and in some other possible manner.

**Apparatus Required:**

A computer system with Oracle 10g installed

**Theory related to experiment:**

A lock is a mechanism associated with a table used to restrict the unauthorized access of the data in a table. MySQL allows a client session to acquire a table lock explicitly to cooperate with other sessions to access the table's data.

MySQL also allows table locking to prevent it from unauthorized modification into the same table during a specific period.

Table Locking in MySQL is mainly used to solve concurrency problems. It will be used while running a transaction, i.e., first read a value from a table (database) and then write it into the table (database).

MySQL provides two types of locks onto the table, which are:

**READ LOCK:** This lock allows a user to only read the data from a table.

**WRITE LOCK:** This lock allows a user to do both reading and writing into a table.

Default storage engine used in MySQL is InnoDB. The InnoDB storage engine does not require table locking manually because MySQL automatically uses row-level locking for InnoDB tables.

Therefore, we can do multiple transactions on the same table simultaneously to read and write operations without making each other wait. All other storage engines use table locking in MySQL.

The LOCK TABLE statement allows you to explicitly acquire a shared or exclusive table lock on the specified table. The table lock lasts until the end of the current transaction.

### Syntax:

LOCK TABLE table-Name IN { SHARE | EXCLUSIVE } MODE

**Program Code:**

To lock the entire Flights table in share mode to avoid a large number of row locks, use the following statement:

LOCK TABLE Flights IN SHARE MODE; SELECT \*

FROM Flights

WHERE orig\_airport > 'OOO';

### Row Share Table Locks (RS):

LOCK TABLE Flights IN ROW SHARE MODE;

### Lock Table in Exclusive Mode:

LOCK TABLE Flights IN EXCLUSIVE MODE;

### Row Exclusive Table Locks (RX):

LOCK TABLE Flights IN ROW EXCLUSIVE MODE;

### Share Row Exclusive Table Locks:

LOCK TABLE Flights IN SHARE ROW EXCLUSIVE MODE;

***Output:***

***SQL query successfully executed. However, the result set is empty.***

*Locks will be implemented on various level and program executed successfully.*

### Text Book:

1. Sql/ Pl/SQL, Bayross, Ivan

### Reference Book:

1. Database System Concepts, Henry F. Korth, S. Sudarshan

### Web Reference:

1. [https://docs.oracle.com/cd/B13789\_01/server.101/b10759/statements\_9015.htm#:~:te](https://docs.oracle.com/cd/B13789_01/server.101/b10759/statements_9015.htm#%3A~%3Atext%3DRW%20EXCLUSIVE%20is%20the%20same%2Cupdating%2C%20inserting%2C%20or%20deleting.%26text%3DSee%20ROW%20SHARE%20.%26text%3DSHARE%20permits%20concurrent%20queries%20but%20prohibits%20updates%20to%20the%20locked%20table) [xt=RW%20EXCLUSIVE%20is%20the%20same,updating%2C%20inserting%2C%2](https://docs.oracle.com/cd/B13789_01/server.101/b10759/statements_9015.htm#%3A~%3Atext%3DRW%20EXCLUSIVE%20is%20the%20same%2Cupdating%2C%20inserting%2C%20or%20deleting.%26text%3DSee%20ROW%20SHARE%20.%26text%3DSHARE%20permits%20concurrent%20queries%20but%20prohibits%20updates%20to%20the%20locked%20table) [0or%20deleting.&text=See%20ROW%20SHARE%20.&text=SHARE%20permits%2](https://docs.oracle.com/cd/B13789_01/server.101/b10759/statements_9015.htm#%3A~%3Atext%3DRW%20EXCLUSIVE%20is%20the%20same%2Cupdating%2C%20inserting%2C%20or%20deleting.%26text%3DSee%20ROW%20SHARE%20.%26text%3DSHARE%20permits%20concurrent%20queries%20but%20prohibits%20updates%20to%20the%20locked%20table) [0concurrent%20queries%20but%20prohibits%20updates%20to%20the%20locked%2](https://docs.oracle.com/cd/B13789_01/server.101/b10759/statements_9015.htm#%3A~%3Atext%3DRW%20EXCLUSIVE%20is%20the%20same%2Cupdating%2C%20inserting%2C%20or%20deleting.%26text%3DSee%20ROW%20SHARE%20.%26text%3DSHARE%20permits%20concurrent%20queries%20but%20prohibits%20updates%20to%20the%20locked%20table) [0table.](https://docs.oracle.com/cd/B13789_01/server.101/b10759/statements_9015.htm#%3A~%3Atext%3DRW%20EXCLUSIVE%20is%20the%20same%2Cupdating%2C%20inserting%2C%20or%20deleting.%26text%3DSee%20ROW%20SHARE%20.%26text%3DSHARE%20permits%20concurrent%20queries%20but%20prohibits%20updates%20to%20the%20locked%20table)
2. https:/[/www.ibm.com/do](http://www.ibm.com/docs/en/informix-servers/14.10?topic=scope-row-key-locks)c[s/en/informix-servers/14.10?topic=scope-row-key-locks](http://www.ibm.com/docs/en/informix-servers/14.10?topic=scope-row-key-locks)

# Viva Questions:

1. What is use of locks?
2. Why are the different type of locks?
3. How locks ensure concurrency?
4. What is Row Exclusive Lock?
5. Why do we use various level of locking?
6. What is difference between share mode and exclusive mode of locking?
7. How can we create locks?
8. What is other name of share lock?

# Experiment No. 4

**Mapped Course Outcomes- CO2**

**CO2:** Create views of data and Implement transaction control using locks.

**AIM:** To create and perform queries on sequences, synonyms and views.

**Objective:** Create table and create sequence. Use this sequence to add data in sequence in it. Create Synonym and Views. Drop and update a View.

**Apparatus Required:**

A computer system with Oracle 10g installed

## Theory related to experiment:

A sequence is a user defined schema bound object that generates a sequence of numeric values.

Sequences are frequently used in many databases because many applications require each row in a table to contain a unique value and sequences provides an easy way to generate them.

The sequence of numeric values is generated in an ascending or descending order at defined intervals and can be configured to restart when exceeds max\_value.

**Syntax:**

CREATE SEQUENCE sequence\_name START WITH initial\_value INCREMENT BY increment\_value MINVALUE minimum value MAXVALUE maximum value CYCLE|NOCYCLE ;

**sequence\_name:** Name of the sequence.

**initial\_value:** starting value from where the sequence starts.

Initial\_value should be greater than or equal to minimum value and less than equal to maximum value.

**increment\_value:** Value by which sequence will increment itself. Increment\_value can be positive or negative.

**minimum\_value:** Minimum value of the sequence.

**maximum\_value:** Maximum value of the sequence.

**cycle:** When sequence reaches its set\_limit. it starts from beginning.

**nocycle:** An exception will be thrown if sequence exceeds its max\_value.

1. Sequence query creating sequence in ascending order:

CREATE SEQUENCE sequence\_1 start with 1

increment by 1

minvalue 0

maxvalue 100 cycle;

**Program Code**: create a table named students with columns as id and name.

CREATE TABLE students

( ID number(10), NAME char(20) ); Now insert values into table

INSERT into students VALUES(sequence\_1.nextval,'Ramesh'); INSERT into students VALUES(sequence\_1.nextval,'Suresh');

### Output:

|  |  |
| --- | --- |
| ID | NAME |
| 1 | Ramesh |
| 2 | Suresh |

**SYNONYMS**

A **SYNONYM** provides another name for database object, referred to as original object, that may exist on a local or another server. A synonym belongs to schema, name of synonym should be unique. A synonym cannot be original object for an additional synonym and synonym cannot refer to user-defined function.

The query below results in an entry for each synonym in database. This query provides details about synonym metadata such as the name of synonym and name of the base object.

select \*

from sys.synonyms ;

### Syntax –

CREATE SYNONYM synonymname

FOR servername.databasename.schemaname.objectname;

### Views:

In SQL, a view is a virtual table based on the result-set of an SQL statement.

A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

You can add SQL statements and functions to a view and present the data as if the data were coming from one single table.

A view is created with the CREATE VIEW statement.

**CREATE VIEW Syntax** CREATE VIEW *view\_name* AS SELECT *column1*, *column2*, ... FROM *table\_name*

WHERE *condition*;

## Program Code:

CREATE VIEW Customers AS SELECT CustomerName, ContactName FROM Customers

WHERE Country = 'England;

***Output:***

***SQL query successfully executed. However, the result set is empty.***

*View will be created for the Table. Program is executed Successfully.*

**DROP VIEW Syntax**

DROP VIEW *view\_name*;

## Program Code:

DROP VIEW Customers;

### Output:

SQL query successfully executed. However, the result set is empty.

### Updating a View

**A view can be updated with the CREATE OR REPLACE VIEW statement.**

CREATE OR REPLACE VIEW Syntax CREATE OR REPLACE VIEW *view\_name* AS SELECT *column1*, *column2*, ...

FROM *table\_name*

WHERE *condition*;

## Program Code:

CREATE OR REPLACE VIEW Customers AS

SELECT CustomerName, ContactName, City FROM Customers

WHERE Country = 'England';

***Output:***

***SQL query successfully executed. However, the result set is empty.***

*View will be replaced with the new values.*

**Text Book:**

1. Sql/ Pl/SQL, Bayross, Ivan

### Reference Book:

1. Database System Concepts, Henry F. Korth, S. Sudarshan
2. Database System Concepts, Korth, Henry

### Web References:

1. https:/[/www.mssqltips.](http://www.mssqltips.com/sqlservertip/6219/create-alter-drop-and-query-sql-server-views/)c[om/sqlservertip/6219/create-alter-drop-and-query-sql-server-views/](http://www.mssqltips.com/sqlservertip/6219/create-alter-drop-and-query-sql-server-views/)

### Video:

<https://youtu.be/Ja5-aVI1TBY>

## Viva Questions:

* 1. Why do we use the Views?
  2. What the role of Sequence?
  3. How do we can set the minimum and maximum value of a sequence?
  4. How do we can create a synonym?
  5. How to create view?
  6. Which attribute can be used with SEQUENCE to set a starting sequence?
  7. Which commands is used to drop a view?
  8. Which command is used to replace a view?
  9. What is the role of synonyms?
  10. How to create sequence?

# Experiment No.5

**Mapped Course Outcomes- CO3**

**CO3:** Apply the concept of PL/SQL programming for control structures and cursors

**AIM:** To Implement PL/SQL programming using Control Structures.

**Objective:** Write PL/SQL commands using control structure to give bonus to employee if sales is greater than his /her quota.

**Apparatus Required:**

A computer system with Oracle 10g installed.

## Theory related to experiment:

PL/SQL statements that control the flow of execution in a PL/SQL program. PL/SQL has three categories of control statements:

**Conditional selection statements**, which run different statements for different data values. 1.IF statements and

1. CASE statements

**Loop statements**, which run the same statements with a series of different data values. 1.LOOP

1. FOR LOOP
2. WHILE LOOP

The EXIT statement transfers control to the end of a loop. The CONTINUE statemnt exits the current iteration of a loop and transfers control to the next iteration. Both EXIT and CONTINUE have an optional WHEN clause, where you can specify a condition.

**Sequential control statements**, which are not crucial to PL/SQL programming.

The sequential control statements are GOTO, which goes to a specified statement, and NULL, which does nothing.

The IF THEN statement has this structure:

### Syntax:

IF *condition* THEN

*statements*

END IF;

•If the condition is true, the statement run; otherwise the IF statement does nothing.

**Program Code:**

DECLARE

sales NUMBER(8,2) := 12100; quota NUMBER(8,2) := 10000; bonus NUMBER(6,2);

emp\_id NUMBER(6) := 120; BEGIN

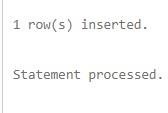
IF sales > (quota + 200) THEN bonus := (sales - quota)/4;

ELSE

bonus := 50; END IF;

UPDATE employees SET salary = salary + bonus WHERE employee\_id = emp\_id; END;

**Output:**



### using nested if:

DECLARE

sales NUMBER(8,2) := 12100;

quota NUMBER(8,2) := 10000; bonus NUMBER(6,2);

emp\_id NUMBER(6) := 120; BEGIN

IF sales > (quota + 200) THEN bonus := (sales - quota)/4;

ELSE

IF sales > quota THEN bonus := 50;

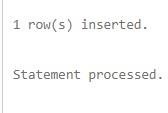
ELSE

bonus := 0; END IF;

END IF;

UPDATE employees SET salary = salary + bonus WHERE employee\_id = emp\_id; END;

**Output:**



Values will be update based on the if and nested if condition and program is executed successfully.

### Text Book:

* 1. Sql/ Pl/SQL, Bayross, Ivan

### Reference Book:

* 1. Database System Concepts, Henry F. Korth, S. Sudarshan

**Web Reference:**

1. <https://www.tutorialspoint.com/plsql/plsql_loops.htm>
2. <https://tutorialink.com/dbms/control-structures.dbms>
3. https://docs.oracle.com/cd/E18283\_01/appdev.112/e17126/controlstatements.htm #:~:text=PL%2FSQL%20has%20three%20categories,series%20of%20different%20d ata%20values.

**Video:**

<https://www.youtube.com/embed/5jj6hE1rvek?feature=oembed>

### Viva Question:

1. Why do we need Control Structure in SQL?
2. What are different loops used in SQL?
3. What is the purpose of Nested IF statement?
4. What is role of CASE?
5. Why If..Else is used?
6. Which control statement will be used when we have to defined two paths based on a certain condition?
7. Which control statement will be used when we have to repeat certain operation multiple time until certain condition is met?
8. Why do we use declare?
9. What will happen if we do not use begin statement?
10. What will happen if we do remove declare in the above program code?

# Experiment No. 6

### Mapped Course Outcomes- CO3

**CO3:** Apply the concept of PL/SQL programming for control structures and cursor

**AIM:** To Implement Pl/SQL programming using Cursors.

**Objective:** Write a program to create cursor and open a cursor.

**Apparatus Required:**

A computer system with Oracle 10g installed

## Theory related to experiment:

A **cursor** is a pointer to this context area. PL/SQL controls the context area through a cursor. A cursor holds the rows (one or more) returned by a SQL statement. The set of rows the cursor holds is referred to as the **active set**.

You can name a cursor so that it could be referred to in a program to fetch and process the rows returned by the SQL statement, one at a time. There are two types of cursors −

•Implicit cursors

•Explicit cursors

### Implicit Cursors:

•Implicit cursors are automatically created by Oracle whenever an SQL statement is executed, when there is no explicit cursor for the statement. Programmers cannot control the implicit cursors and the information in it.

•Whenever a DML statement (INSERT, UPDATE and DELETE) is issued, an implicit cursor is associated with this statement. For INSERT operations, the cursor holds the data that needs to be inserted. For UPDATE and DELETE operations, the cursor identifies the rows that would be affected.

### Explicit cursors

•Explicit cursors are programmer-defined cursors for gaining more control over the **context area**. An explicit cursor should be defined in the declaration section of the PL/SQL Block. It is created on a SELECT Statement which returns more than one row.

The syntax for creating an explicit cursor is − CURSOR cursor\_name IS select\_statement; **Declaring the Cursor**

Declaring the cursor defines the cursor with a name and the associated SELECT statement. For example −

CURSOR c\_customers IS

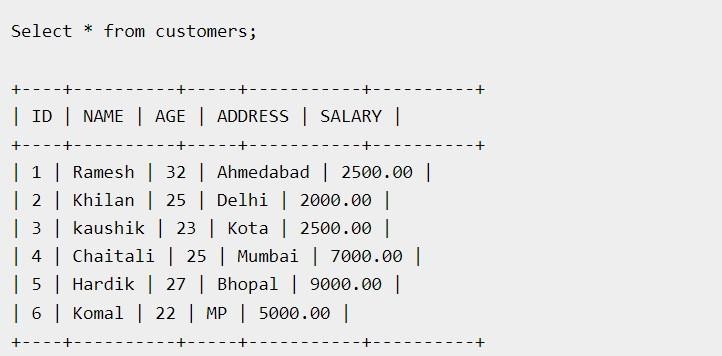
SELECT id, name, address FROM customers;

### Opening the Cursor

Opening the cursor allocates the memory for the cursor and makes it ready for fetching the rows returned by the SQL statement into it. For example, we will open the above defined cursor as follows −

OPEN c\_customers;

### Sample Input:



**Program Code:**

DECLARE

c\_id customers.id%type; c\_name customers.name%type;

c\_addr customers.address%type; CURSOR c\_customers is

SELECT id, name, address FROM customers; BEGIN

OPEN c\_customers; LOOP

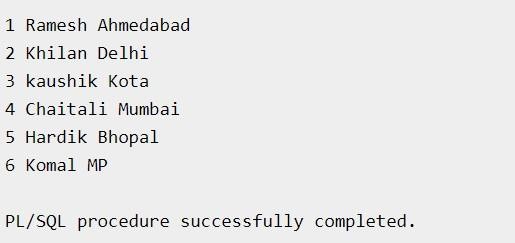
FETCH c\_customers into c\_id, c\_name, c\_addr; EXIT WHEN c\_customers%notfound;

dbms\_output.put\_line(c\_id || ' ' || c\_name || ' ' || c\_addr); END LOOP;

CLOSE c\_customers; END;

/

### Output:



**Text Book:**

1. Sql/ Pl/SQL, Bayross, Ivan

### Reference Book:

1. Database System Concepts, Henry F. Korth, S. Sudarshan

**Web Reference:**

1. <https://www.tutorialspoint.com/plsql/plsql_cursors.htm>
2. [**https://www.geeksforgeeks.org/cursors-in-pl-sql/**](https://www.geeksforgeeks.org/cursors-in-pl-sql/)

# Viva Questions:

1. What is role of Cursor?
2. What is the different type of the cursors used?
3. How do we cursors?
4. Name the SQL command with whom the implicit cursor is always associated?
5. What is the difference between implicit and explicit cursor?
6. Can you write program code to create a cursor?
7. Can you write a code to open a cursor?
8. Implicit cursors are automatically created in the oracle or do we need to create separately?

# Experiment No. 7

### Mapped Course Outcomes- CO3, CO4

**CO3: A**pply the concept of PL/SQL programming for control structures and cursors.

**CO4:** Apply the concept of PL/SQL programming for performing operations on packages and triggers.

**AIM:** To Implement Pl/SQL programming using exception handling.

**Objective:** Create a program to handle the exception.

**Apparatus Required:**

A computer system with Oracle 10g installed

## Theory related to experiment:

### Exceptions:

An exception is an error condition during a program execution. PL/SQL supports programmers to catch such conditions using **EXCEPTION** block in the program and an appropriate action is taken against the error condition.

There are two types of exceptions −

•System-defined exceptions

•User-defined exceptions

### Advantages of PL/SQL Exceptions:

Using exceptions for error handling has several advantages. Without exception handling, every time you issue a command, you must check for execution errors:

Error processing is not clearly separated from normal processing; nor is it robust. If you neglect to code a check, the error goes undetected and is likely to cause other, seemingly unrelated errors.

With exceptions, you can handle errors conveniently without the need to code multiple checks, as follows:

BEGIN SELECT ...

SELECT ...

SELECT ...

... EXCEPTION

WHEN NO\_DATA\_FOUND THEN -- catches all 'no data found' errors

Exceptions improve readability by letting you isolate error-handling routines. The primary algorithm is not obscured by error recovery algorithms. Exceptions also improve reliability. You need not worry about checking for an error at every point it might occur. Just add an exception handler to your PL/SQL block. If the exception is ever raised in that block (or any sub-block), you can be sure it will be handled.

### User-defined Exceptions

PL/SQL allows you to define your own exceptions according to the need of your program. A user-defined exception must be declared and then raised explicitly, using either a RAISE statement or the procedure **DBMS\_STANDARD.RAISE\_APPLICATION\_ERROR**.

### The syntax for declaring an exception is −

DECLARE

my-exception EXCEPTION;

Exceptions can be declared only in the declarative part of a PL/SQL block, subprogram, or package. You declare an exception by introducing its name, followed by the keyword EXCEPTION.

Exception and variable declarations are similar. But remember, an exception is an error condition, not a data item. Unlike variables, exceptions cannot appear in assignment statements or SQL statements. However, the same scope rules apply to variables and exceptions.

**Program Code:**

DECLARE

out\_of\_stock EXCEPTION; number\_on\_hand NUMBER(4);

BEGIN

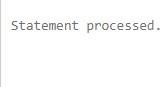
IF number\_on\_hand < 1 THEN RAISE out\_of\_stock;

END IF;

EXCEPTION

WHEN out\_of\_stock THEN dbms\_output.put\_line('No such customer!'); END;

### Output:



Program will raise Out of Stock exception.

### Text Book:

1. Sql/ Pl/SQL, Bayross, Ivan

### Reference Book:

1. Database System Concepts, Henry F. Korth, S. Sudarshan

### Web Reference:

<https://www.tutorialspoint.com/plsql/plsql_exceptions.htm>

# Viva Question:

1. Why Exceptions occurs?
2. What are type of Exception?
3. How do we can handle it?
4. What System defined exception are different from user-defined exceptions?
5. How can we raise exception?
6. Name some of the System defined Exceptions?
7. How can you create your exception?
8. Which command is used to handle the Exception?
9. What are the advantages of exception handling?
10. Why exception handling is required?

# Experiment No. 8

**Mapped Course Outcomes- CO4**

**CO4:** Apply the concept of PL/SQL programming for performing operations on packages and triggers.

**AIM:** Perform various operations on Packages and Triggers

**Objective:** Perform various operations on Packages and create Triggers.

**Apparatus Required:**

A computer system with Oracle 10g installed

## Theory related to experiment:

**Packages:**

Packages are schema objects that groups logically related PL/SQL types, variables and subprograms. A package will have two mandatory parts:

* Package specification
* Package body or definition

### Package Specification:

The specification is the interface to the package. It just DECLARES the types, variables, constants, exceptions, cursors, and subprograms that can be referenced from outside the package. In other words, it contains all information about the content of the package, but excludes the code for the subprograms.

All objects placed in the specification are called public objects. Any subprogram not in the package specification but coded in the package body is called a private object.

The following code snippet shows a package specification having a single procedure. You can have many global variables defined and multiple procedures or functions inside a package.

## Program Code:

*CREATE PACKAGE cust\_sal AS PROCEDURE find\_sal(c\_id number); END cust\_sal;*

/

### Output:

When the above code is executed at SQL prompt, it produces the following result:



### Package Body:

When the above code is executed at SQL prompt, it produces the following result: Package body created

The package body has the codes for various methods declared in the package specification and other private declarations, which are hidden from code outside the package.

The CREATE PACKAGE BODY Statement is used for creating the package body. The following code snippet shows the package body declaration for the cust\_sal package created above.

## Program Code:

*CREATE OR REPLACE PACKAGE BODY cust\_sal AS*

*PROCEDURE find\_sal(c\_id customers.id%TYPE) IS c\_sal customers.salary%TYPE;*

*BEGIN*

*SELECT salary INTO c\_sal FROM customers*

*WHERE id = c\_id; dbms\_output.put\_line('Salary: '|| c\_sal); END find\_sal;*

*END cust\_sal;*

/

### Output:

When the above code is executed at SQL prompt, it produces the following result:

*Package body created*

### Using the Package Elements:

The package elements (variables, procedures or functions) are accessed with the following syntax:

*package\_name.element\_name;*

Consider, we already have created above package in our database schema, the following program

uses the find\_sal method of the cust\_sal package:

## Program Code:

*DECLARE*

*code customers.id%type := &cc\_id; BEGIN*

*cust\_sal.find\_sal(code); END;*

*/*

### Output:

When the above code is executed at SQL prompt, it prompts to enter customer ID, and when you

enter an ID, it displays the corresponding salary as follows:

*Enter value for cc\_id: 1 Salary: 3000*

*PL/SQL procedure successfully completed.*

### Triggers:

Triggers are stored programs, which are automatically executed or fired when some events occur. Triggers are, in fact, written to be executed in response to any of the following events:

* + A database manipulation (DML) statement (DELETE, INSERT, or UPDATE).
  + A database definition (DDL) statement (CREATE, ALTER, or DROP).
  + A database operation (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).

Triggers could be defined on the table, view, schema, or database with which the event is associated.

### Creating Triggers

**The syntax for creating a trigger is:**

*CREATE [OR REPLACE ] TRIGGER trigger\_name*

*{BEFORE | AFTER | INSTEAD OF }*

*{INSERT [OR] | UPDATE [OR] | DELETE}*

*[OF col\_name]*

*ON table\_name*

*[REFERENCING OLD AS o NEW AS n] [FOR EACH ROW]*

*WHEN (condition) DECLARE*

*Declaration-statements BEGIN*

*Executable-statements EXCEPTION*

*Exception-handling-statements END;*

Where,

CREATE [OR REPLACE] TRIGGER trigger\_name : Creates or replaces an existing trigger with the trigger\_name.

* + {BEFORE | AFTER | INSTEAD OF}: This specifies when the trigger would be executed. The INSTEAD OF clause is used for creating trigger on a view.
  + {INSERT [OR] | UPDATE [OR] | DELETE}: This specifies the DML operation.
  + [OF col\_name]: This specifies the column name that would be updated.
  + [ON table\_name]: This specifies the name of the table associated with the trigger.
  + [REFERENCING OLD AS o NEW AS n]: This allows you to refer new and old values for various DML statements, like INSERT, UPDATE, and DELETE.
  + [FOR EACH ROW]: This specifies a row level trigger, i.e., the trigger would be executed for each row being affected. Otherwise the trigger will execute just once when the SQL statement is executed, which is called a table level trigger.
  + WHEN (condition): This provides a condition for rows for which the trigger would fire.

This clause is valid only for row level triggers.

To start with, we will be using the CUSTOMERS table:

Select \* from customers;

+ + + + + +

| ID | NAME | AGE | ADDRESS | SALARY |

+ + + + + +

| 1 | Ramesh| 32 | Ahmedabad | 2000.00 |

|  |  |  |  |
| --- | --- | --- | --- |
| | 2 | | Khilan | 25 | Delhi | | | 1500.00 | |
| | 3 | | Kaushik | 23 | Kota | | | 2000.00 | |
| | 4 | | Chaitali | | 25 | Mumbai | | 6500.00 | |
| | 5 | | Hardik | | 27 | Bhopal | | 8500.00 | |
| | 6 | | Komal | | 22 | MP | | 4500.00 | |

+ + + + + +

he following program creates a row level trigger for the customers table that would fire for INSERT or UPDATE or DELETE operations performed on the CUSTOMERS table. This trigger will display the salary difference between the old values and new values:

## Program Code:

*CREATE OR REPLACE TRIGGER display\_salary\_changes BEFORE DELETE OR INSERT OR UPDATE ON customers FOR EACH ROW*

*WHEN (NEW.ID > 0) DECLARE*

*sal\_diff number;*

*BEGIN*

*sal\_diff := :NEW.salary - :OLD.salary; dbms\_output.put\_line('Old salary: ' || :OLD.salary); dbms\_output.put\_line('New salary: ' || :NEW.salary); dbms\_output.put\_line('Salary difference: ' || sal\_diff); END;*

*/*

### Output:

When the above code is executed at SQL prompt, it produces the following result:

*Trigger created.*

Here following two points are important and should be noted carefully:

* + OLD and NEW references are not available for table level triggers, rather you can use them for record level triggers.
  + If you want to query the table in the same trigger, then you should use the AFTER keyword, because triggers can query the table or change it again only after the initial changes are applied and the table is back in a consistent state.
  + Above trigger has been written in such a way that it will fire before any DELETE or INSERT or UPDATE operation on the table, but you can write your trigger on a single or multiple operations, for example BEFORE DELETE, which will fire whenever a record will be deleted using DELETE operation on the table*.*

### Text Book:

1. Sql/ Pl/SQL, Bayross, Ivan

### Reference Book:

1. Database System Concepts, Henry F. Korth, S. Sudarshan

**Web Reference:**

* 1. <https://www.tutorialspoint.com/plsql/plsql_packages.htm>
  2. https://docs.oracle.com/cd/B19306\_01/server.102/b14200/statements\_6006.ht m

# Viva Question:

1. Why do we create Packages?
2. Why do we create triggers?
3. How do we create triggers?
4. Why do we need package body?
5. Why do we need package specifications?
6. Write code for the create trigger?
7. How do we specify when the trigger will be executed?
8. How do we specify which column need to be updated?

# Experiment No.- 9

### Mapped Course Outcomes- CO5

**CO5:** Analyze and apply advanced normalization technique and redundant functional dependencies

**AIM:** Prepare a Case Study explaining the need for converting tables to fourth and fifth Normal forms.

**Objective:** Understand the process of converting table into fourth and fifth Normal forms.

**Theory related to experiment:**

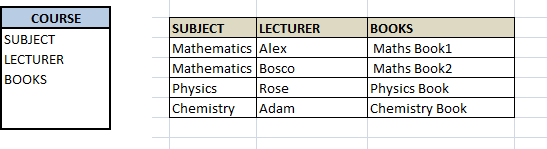
**Fourth Normal Form (4NF):**

The 4NF comes after 1NF, 2NF, 3NF, and Boyce-Codd Normal Form. It was introduced by Ronald Fagin in 1977.

In the fourth normal form,

* + It should meet all the requirement of 3NF
  + Attribute of one or more rows in the table should not result in more than one rows of the same table leading to multi-valued dependencies

**Case Study:** To understand the need of converting table in to fourth normal form, we should understand that we are trying to reduce multivalued dependency. consider a table with Subject, Lecturer who teaches each subject and recommended Books for each subject.



If we observe the data in the table above it satisfies 3NF. But LECTURER and BOOKS are two independent entities here. There is no relationship between Lecturer and Books. In the above case study either Alex or Bosco can teach Mathematics. For Mathematics subject , student can refer either ‘Maths Book1’ or ‘Maths Book2’.

SUBJECT –> LECTURER SUBJECT–>BOOKS

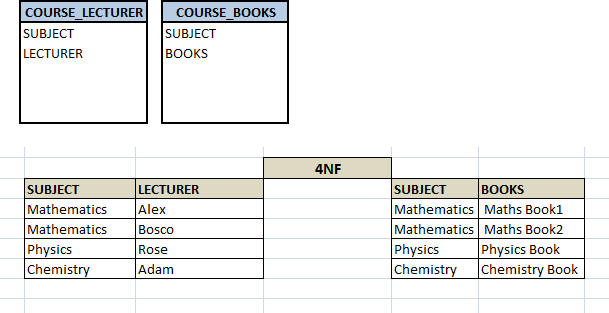
This is a multivalued dependency on SUBJECT. If we need to select both lecturer and books recommended for any of the subject, it will show up (lecturer, books) combination, which implies lecturer who recommends which book. This is not correct.

SELECT c.LECTURER, c.BOOKS FROM COURSE c WHERE SUBJECT =

'Mathematics';

### Solution:

To eliminate this dependency, we divide the table into two as below:



Now if we want to know the lecturer names and books recommended for any of the subject, we will fire two independent queries. Hence it removes the multi-valued dependency and confusion around the data. Thus the table is in 4NF

--Select the lecturer names

SELECT c.SUBJECT , c.LECTURER FROM COURSE c WHERE c.SUBJECT =

'Mathematics';

--Select the recommended book names

SELECT c.SUBJECT , c.BOOKS FROM COURSE c WHERE c.SUBJECT = 'Mathematics';

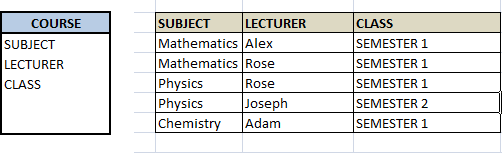
## FIFTH NORMAL FORM:

A database is said to be in 5NF, if and only if,

* + It’s in 4NF
  + If we can decompose table further to eliminate redundancy and anomaly, and when we re-join the decomposed tables by means of candidate keys, we should not be losing the original data or any new record set should not arise. In simple words, joining two or more decomposed table should not lose records nor create new records.

**Case Study:** Consider an case study of different Subjects taught by different lecturers and the lecturers taking classes for different semesters.

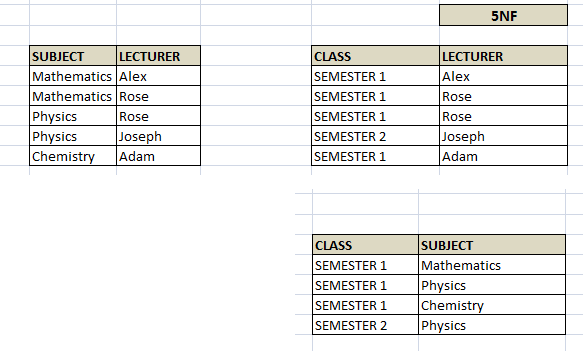
Note: Please consider that Semester 1 has Mathematics, Physics and Chemistry and Semester 2 has only Mathematics in its academic year!!



In above table, Rose takes both Mathematics and Physics class for Semester 1, but she does not take Physics class for Semester 2. In this case, combination of all these 3 fields is required to identify a valid data. Imagine we want to add a new class – Semester3 but do not know which Subject and who will be taking that subject. We would be simply inserting a new entry with Class as Semester3 and leaving Lecturer and subject as NULL. As we discussed above, it’s not a good to have such entries. Moreover, all the three columns together act as a primary key, we cannot leave other two columns blank!

### Solution:

Hence we have to decompose the table in such a way that it satisfies all the rules till 4NF and when join them by using keys, it should yield correct record. Here, we can represent each lecturer’s Subject area and their classes in a better way. We can divide above table into three – (SUBJECT, LECTURER), (LECTURER, CLASS), (SUBJECT, CLASS)



Now, each of combinations is in three different [tables](https://www.tutorialcup.com/dbms/tables.htm). If we need to identify who is teaching which subject to which semester, we need join the keys of each table and get the result.

For example, who teaches Physics to Semester 1, we would be selecting Physics and Semester1 from table 3 above, join with table1 using Subject to filter out the lecturer names. Then join with table2 using Lecturer to get correct lecturer name. That is we joined key columns of each table to get the correct data. Hence there is no lose or new data – satisfying 5NF condition.

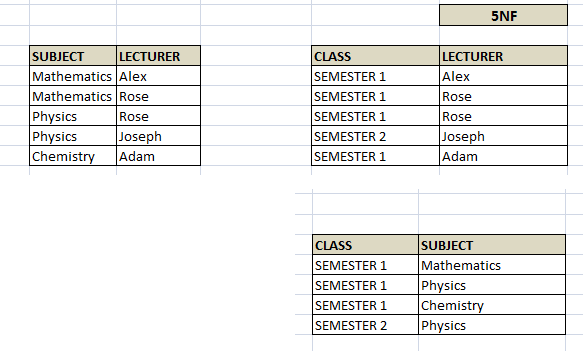
SELECT t3.Class, t3.Subject, t1.Lecturer FROM TABLE3 t3, TABLE3 t2, TABLE3 t1,

where t3.Class = 'SEMESTER1' and t3.SUBJECT= 'PHYSICS'

AND t3.Subject = t1.Subject AND t3.Class = t2.Class

AND t1.Lecturer = t2.Lecturer;

### Output:



**Text Book:**

1. Sql/ Pl/SQL, Bayross, Ivan

### Reference Book:

1. Database System Concepts, Henry F. Korth, S. Sudarshan

# Viva Question:

1. What is Normalization?
2. What are 4th and 5th Normal forms?
3. Why do we need to convert relation in to 4 NF?
4. Why do we need to convert relation in to 5NF?
5. How can be convert relation into 4NF?
6. How can be convert relation into 5NF?
7. What is join Dependency?
8. What is multivalued dependency?
9. What is denormalization?
10. How do we achieve denormalization?

# Experiment No. 10

### Mapped Course Outcomes- CO5

**CO5:** Analyze and apply advanced normalization technique and redundant functional dependencies.

**AIM:** Prepare a Case Study on Functional dependency with examples for redundant functional dependencies.

**Objective:** Analyse Functional dependency and find redundant functional dependencies and remove them.

**Theory related to experiment:**

### Functional Dependency

Functional dependency in DBMS, as the name suggests is a relationship between attributes of a table dependent on each other. Introduced by E. F. Codd, it helps in preventing data redundancy and gets to know about bad designs.

To understand the concept thoroughly, let us consider P is a relation with attributes A and B. Functional Dependency is represented by -> (arrow sign)

Then the following will represent the functional dependency between attributes with an arrow sign −

Functional Dependency:

A -> B

**B-** functionally dependent on **A A-**determinant set

**B-** dependent attribute

### Example

The following is an example that would make it easier to understand functional dependency −

We have a <Department> table with two attributes − DeptId and DeptName.

DeptId = Department ID DeptName = Department Name

The DeptId is our primary key. Here, DeptId uniquely identifies the DeptName attribute. This is because if you want to know the department name, then at first you need to have the DeptId.

|  |  |
| --- | --- |
| **DeptId** | **DeptName** |
| 001 | Finance |
| 002 | Marketing |
| 003 | HR |

Therefore, the above functional dependency between DeptId and DeptName can be determined as DeptId is functionally dependent on DeptName −

DeptId -> DeptName

## Types of Functional Dependency

Functional Dependency has three forms −

* + Trivial Functional Dependency
  + Non-Trivial Functional Dependency
  + Completely Non-Trivial Functional Dependency

### Trivial Functional Dependency:

It occurs when B is a subset of A in − A ->B

### Example

We are considering the same <Department> table with two attributes to understand the concept of trivial dependency.

The following is a trivial functional dependency since DeptId is a subset of DeptId and DeptName

{ DeptId, DeptName } -> Dept Id

### Non –Trivial Functional Dependency:

It occurs when B is not a subset of A in − A ->B

Example

DeptId -> DeptName

The above is a non-trivial functional dependency since DeptName is a not a subset of DeptId. Completely Non - Trivial Functional Dependency

It occurs when A intersection B is null in − A ->B

## Armstrong’s Axioms Property of Functional Dependency

Armstrong’s Axioms property was developed by William Armstrong in 1974 to reason about functional dependencies.

The property suggests rules that hold true if the following are satisfied:

### Transitivity

If A->B and B->C, then A->C i.e. a transitive relation.

### Reflexivity

A-> B, if B is a subset of A.

### Augmentation

The last rule suggests: AC->BC, if A->B

## Redundant functional dependencies:

A functional dependency in the set is redundant if it can be derived from the other functional dependencies in the set. A redundant FD can be detected using the following steps:

**Step 1:** Start with a set of S of functional dependencies (FDs).

**Step 2:** Remove an FD f and create a set of FDs S' = S - f .

**Step 3:** Test whether f can be derived from the FDs in S'; by using the set of Armstrong's axioms and derived rules.

**Step 4:** If f can be so derived, it is redundant , and hence S' = S. Otherwise replace f into S'; so that now S' = S + f.

**Step 5:** Repeat steps 2 to 4 for all FDs in S.

### Algorithm: Membership algorithm to find redundant functional dependency

An algorithm (called membership algorithm) can be developed to find redundant FDs, that is, to determine whether an FD f(A -> B) can be derived from a set of FDs S. Using the algorithm the redundant functional dependency can be checked.

### Input: Let F be a set of FDs for relation R.

**Steps:**

* 1. **F' = F - f** //find out new set of FDs by removing f from F.
  2. **T = A** //set T = determinant of A -> B

### for each FD:X -> yin F' Do

* + 1. **If X** ⊆ **T** Then //if X is contained in T

**T = T** 𝖴 **Y** //add Y to T End if

* 1. if **B** ⊆ **T** then //if B is contained in T

f : A -> B is redundant. //given FD f: A -> B is redundant. End if

**Output:** Decision Whether a given FD f: A -> B is redundant or not.

## Case Study:

Suppose, there is a relation R(A,B,X,Y,Z) and the following set of FDs is given for the table R.

Z -> A B -> X AX -> Y

ZB -> Y.

Now, It is the case that these functional dependency must be implement for the table. Having too many functional dependency increases the computational time. So, How can be reduce this computational time?

### Solution:

The following set of FDs is given for the table R. Z -> A

B -> X AX -> Y ZB -> Y.

In order to reduce computational time, removing redundant functional dependency helps to achieve the target. To find the redundant functional dependency, following steps need to executed:

**Step 1:** Start with a set of S of functional dependencies (FDs).

**Step 2:** Remove an FD f and create a set of FDs S' = S - f .

**Step 3:** Test whether f can be derived from the FDs in S'; by using the set of Armstrong’s axioms and derived rules.

**Step 4:** If f can be so derived, it is redundant, and hence S' = S. Otherwise replace f into S'; so that now S' = S + f.

**Step 5:** Repeat steps 2 to 4 for all FDs in S.

Hence, in this case study, we will apply these steps. It gives us following redundant functional dependencies:

Because ZB -> Y Can be derived from other FDs in the set, it can be shown to be redundant. The following argument can be given:

* + Z -> A by augmentation rule will yield ZB -> AB.
  + B -> X and AX -> Y by pseudo-transitivity rule will yield AB -> Y.
  + ZB -> AB and AB -> Y by transitivity rule will yield ZB -> Y.

### Output:

ZB -> Y is a redundant functional dependency. It must be removed.

### Text Book:

1. Sql/ Pl/SQL, Bayross, Ivan

### Reference Book:

1. Database System Concepts, Henry F. Korth, S. Sudarshan

### Web References:

1. <https://www.javatpoint.com/dbms-functional-dependency>
2. https://opentextbc.ca/dbdesign01/chapter/chapter-11-functional- dependencies/#:~:text=A%20functional%20dependency%20(FD)%20is,determines% 20the%20value%20of%20Y.

# Viva Question:

1. Why we remove redundant functional dependency?
2. What is functional dependency?
3. How to remove redundant functional dependency?
4. What is the role of functional dependency?
5. What are the armstrong’s Axioms?
6. What is algorithm to remove redundant functional dependency?
7. What is trivial Functional dependency?
8. What is the prime Attribute?
9. What is the non-trivial functional dependency?
10. What is transitive functional dependency?

**Continuous Assessment**

**Lab MST**