Database Management System (CS-2004) Lab

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

School of Computer Engineering

4 Credit

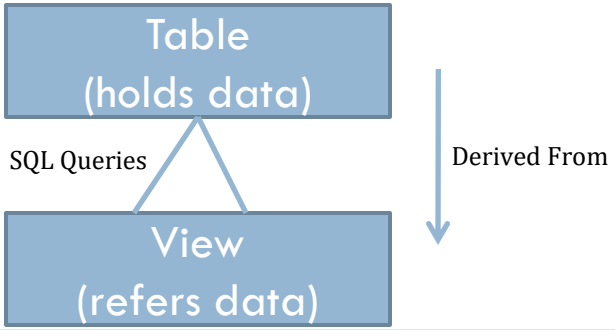
Mr. Rajat Kumar Behera - Associate Professor

Lab Contents

Sr #	Major and Detailed Coverage Area	Lab#
1	❑ Creating and Using Views ❑ Creating and Using Synonyms ❑ Using Indexes to improve performance	7
2	DCL ❑ Grant ❑ Revoke	
3	TCL ❑ Begin ❑ Commit ❑ Rollback	

Views

A view is a **virtual table**. That is, a view looks like a table and acts like a table as far as a user is concerned, but it doesn't require physical storage. A view is actually a composition of a table in the form of a predefined query, which is stored in the database. For example, a view can be created from EMPLOYEE that contains only the employee's name and address, instead of all columns in EMPLOYEE. A view can contain all rows of a table or selected rows from a table. A view can be created from one or many tables. When a view is created, a **SELECT** statement is actually run against the database, which defines the view. The **SELECT** statement that defines the view might simply contain column names from the table, or it can be more explicitly written using various functions and calculations to manipulate or summarize the data that the user sees.



Views cont…

A view is considered a database object, although the view takes up no storage space on its own. The main difference between a view and a table is that data in a table consumes physical storage, whereas a view does not require physical storage because it is actually referring to data from a table. A view is used in the same manner that a table is used in the database, meaning that data can be selected from a view as it is from a table.

Dropping Tables Used by Views

If a table that created a view is dropped, the view becomes inaccessible. An error will be popped-up when trying to query against the view.

Why Views?

- Simplification of Data Access
- Form of Security
- Maintain Summarized Data

Creating Views

- Creating a View from a Single Table
- Creating a View from Multiple Tables
- Creating a View from View(s)

Creating View from a Single Table

Syntax:

CREATE VIEW view_name AS SELECT columns FROM table [WHERE conditions];

Examples:

- CREATE VIEW EMPLOYEE_VIEW AS SELECT * FROM EMPLOYEE;
- CREATE VIEW EMP_NAME_VIEW AS SELECT FIRST_NAME, MIDDLE_NAME, LAST_NAME FROM EMPLOYEE;
- CREATE VIEW EMP_FULL_NAME_VIEW AS SELECT FIRST_NAME || ' ' || MIDDLE_NAME || ' ' || LAST_NAME AS FULL_NAME FROM EMPLOYEE;
- CREATE VIEW CITY_PAY AS SELECT E.CITY, AVG(P PAY_RATE) AVG_PAY FROM EMPLOYEE E, EMPLOYEE_PAY P WHERE E.EMP_ID = P.EMP_ID GROUP BY E.CITY;

Creating View from Multiple Tables

Syntax:

CREATE VIEW view_name AS SELECT columns FROM table1 [INNER | LEFT OUTER | RIGHT OUTER | FULL OUTER] JOIN ON table 2 [conditions] [WHERE conditions] ;

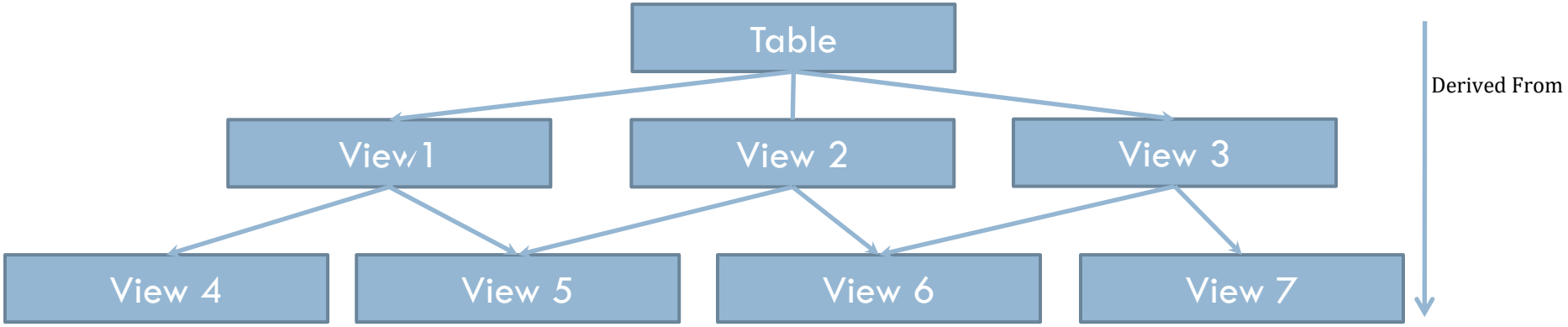
Examples:

- CREATE VIEW EMPLOYEE_SUMMARY AS SELECT E.EMP_ID, E.LAST_NAME, P.POSITION, P.DATE_HIRE, P.PAY_RATE FROM EMPLOYEE E, EMPLOYEE PAY P WHERE E.EMP_ID = P.EMP_ID;
- CREATE VIEW EMPLOYEE_SUMMARY AS SELECT E.EMP_ID, E.LAST_NAME, P.POSITION, P.DATE_HIRE, P.PAY_RATE FROM EMPLOYEE E INNER JOIN EMPLOYEE PAY P ON E.EMP_ID = P.EMP_ID;

Note

When selecting data from multiple tables, the tables must be joined by common columns.

Creating View from Views



A view can be created from another view using the following format:

CREATE VIEW4 AS SELECT * FROM VIEW1;

CREATE VIEW6 AS SELECT * FROM VIEW1 UNION SELECT * FROM VIEW3;

Note

You can create a view from a view many layers deep (a view of a view of a view, and so on). How deep you can go is implementation specific. The only problem with creating views based on other views is their manageability. For example, suppose that you create VIEW2 based on VIEW1 and then create VIEW3 based on VIEW2. If VIEW1 is dropped, VIEW2 and VIEW3 are no good. The underlying information that supports these views no longer exists. Therefore, always maintain a good understanding of the views in the database and on which other objects those views rely.

Views and the ORDER BY Clause

ORDER BY clause cannot be used in the CREATE VIEW statement; however, the GROUP BY clause has the same effect as an ORDER BY clause when it's used in the CREATE VIEW statement.

Example –

```
CREATE VIEW EMPLOYEE _NAMES_VIEW AS
SELECT LAST_NAME || ', ' || FIRST_NAME || ' ' ||MIDDLE_NAME NAME
FROM EMPLOYEE
GROUP BY LAST_NAME || ', ' || FIRST_NAME || ' ' || MIDDLE_NAME;
```

Access, Updation and Dropping the Views

Access : This Oracle CREATE VIEW would create a virtual table based on the result set of the SELECT statement. You can now query the VIEW as follows:

SELECT * FROM EMPLOYEE _NAMES_VIEW;

SELECT * FROM EMPLOYEE _NAMES_VIEW WHERE LENGTH(NAME) > 10;

Updation: You can modify the definition of an VIEW without dropping it by using the CREATE OR REPLACE VIEW statement.

- **Syntax** – CREATE OR REPLACE VIEW view_name AS SELECT columns FROM table WHERE conditions;
- **Example** – CREATE OR REPLACE VIEW EMPLOYEE_VIEW AS SELECT * FROM EMPLOYEE;

Drop: Once the VIEW has been created, it can be dropped with the DROP VIEW Statement.

- **Syntax** – DROP VIEW view_name;
- **Example** – DROP VIEW EMPLOYEE_VIEW;

Updating Data Through a View

You can update the underlying data of a view under certain conditions:

- The view must not involve joins.
- The view must not contain a GROUP BY clause.
- The view must not contain a UNION statement.
- The view cannot contain a reference to the pseudo-column ROWNUM.
- The view cannot contain group functions.
- The DISTINCT clause cannot be used.
- The WHERE clause cannot include a nested table expression that includes a reference to the same table as referenced in the FROM clause.

Conclusion

The view can perform INSERT, UPDATE, and DELETE as long as they honor these caveats.

Synonyms

A synonym is an alternative name for objects such as tables, views, sequences, stored procedures, and other database objects. You generally use synonyms when you are granting access to an object from another schema and you don't want the users to have to worry about knowing which schema owns the object. Synonyms can be created as PUBLIC or PRIVATE. Any user of the database can use a PUBLIC synonym; only the owner of a database and any users that have been granted privileges can use a PRIVATE synonym. Either a database administrator (or another designated individual) or individual users manage synonyms.

Creating and Updating Synonyms

Syntax: CREATE [OR REPLACE] [PUBLIC] SYNONYM [schema .] synonym_name FOR [schema.] object_name

Example:

- CREATE SYNONYM CUST FOR CUSTOMER;
- CREATE OR REPLACE SYNONYM PRODUCT FOR USER1.PRODUCT;

Synonyms cont…

Dropping Synonym

Syntax:

DROP [PUBLIC] SYNONYM [schema .] synonym_name;

Example:

- DROP SYNONYM CUST;
- DROP SYNONYM PRODUCT;

Index

An index **is a pointer** to data in a table. An index in a database is similar to an index in the back of a book. For example, if you want to reference all pages in a book that discuss a certain topic, you first refer to the index, which lists all topics alphabetically, and it refers you to one or more specific page numbers. An index in a database works the same way in that a query is pointed to the exact physical location of data in a table. You are actually being directed to the data's location in an underlying file of the database, but as far as you are concerned, you are referring to a table. Which would be faster, looking through a book page by page for some information or searching the book's index and getting a page number? Of course, using the book's index is the most efficient method. It can save a lot of time, especially if the book is large. If you have a book of just a few pages, however, it might be faster to check the pages for the information than to flip back and forth between the index and pages of the book. When a database does not use an index, it is performing what is typically called a **full table scan**, the same as flipping through a book page by page.

Index cont…

An index is typically stored separately from the table for which the index was created. An index's main purpose is to improve the performance of data retrieval. Indexes can be created or dropped with no effect on the data. However, after an index is dropped, performance of data retrieval might be slowed. Indexes do take up physical space and can often grow larger than the table. Therefore, you should consider them when estimating your database storage needs.

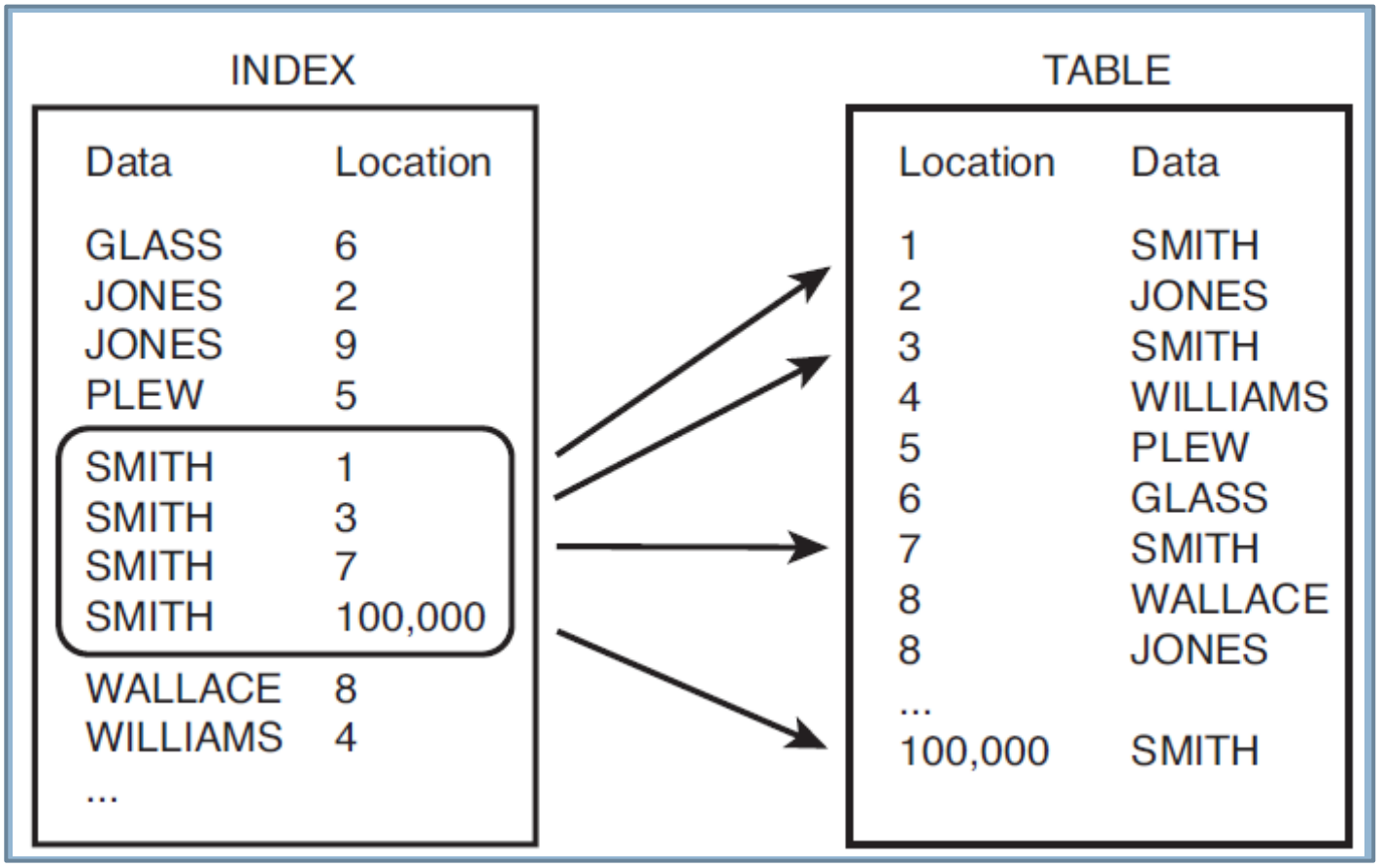
How do Indexes Work?

When an index is created, it records the location of values in a table that are associated with the column that is indexed. Entries are added to the index when new data is added to the table. When a query is executed against the database and a condition is specified on a column in the WHERE clause that is indexed, the index is first searched for the values specified in the WHERE clause. If the value is found in the index, the index returns the exact location of the searched data in the table. Figure shown next illustrates the functioning of an index.

How Indexes Work?

Suppose the following query was issued:

SELECT * FROM TABLE_NAME WHERE NAME = 'SMITH';



NAME index is referenced to resolve the location of all names equal to SMITH. After the location is determined, the data can quickly be retrieved from the table. The data, in this case, names is alphabetized in the index.

Index Creation

The syntax for creating an index in Oracle/PLSQL is:

CREATE INDEX index_name ON table_name (column1, column2, ... column_n)

- **Single-Column Indexes:** Indexing on a single column of a table is the simplest and most common manifestation of an index. Obviously, a single-column index is one that is created based on only one table column.

  Example - CREATE INDEX NAME_IDX ON EMPLOYEE (LAST_NAME);

- **Composite Indexes -** A composite index is an index on two or more columns of a table. You should consider performance when creating a composite index, because the order of columns in the index has a measurable effect on the data retrieval speed. Generally, the most limiting value should be placed first for optimum performance. However, the columns that are always specified in your queries should be placed first.

  Example - CREATE INDEX ORD_IDX ON ORDER(CUST_ID, PROD_ID);

Index cont...

Rename an Index

Syntax : ALTER INDEX index_name RENAME TO new_index_name;

Example : ALTER INDEX NAME_IDX RENAME TO EMP_NAME_IDX;

Drop an Index

Syntax : DROP INDEX index_name;

Example : DROP INDEX EMP_NAME_IDX;

DCL – Data Control Language

The DCL commands are used to enforce database security i.e. to perform any operation in the database, such as for creating tables, sequences or views, we need privileges.

- Privileges are of two types
  - **System :** creating session, table etc are all types of system privilege.
  - **Object :** any command or query to work on tables comes under object privilege.
- There are two types of DCL commands. They are GRANT and REVOKE.
  - **GRANT -** This command is used to provide access or privileges on the database objects to the users.
  - **REVOKE -** This command removes the user access rights or privileges to the database objects.
- Only Database Administrator's or owner's of the database object can provide/remove privileges on a database object.

DCL cont…

- To allow a user to create session: ***grant create session to username;***
- To allow a user to create table: ***grant create table to username;***
- To grant all privilege to a user : ***grant sysdba to username;***
- To grant permission to create table : ***grant create any table to username***
- To grant permission to drop any table : ***grant drop any table to username***
- To take back Permissions : ***revoke create table from username***

TCL – Transaction Control Language

TCL commands are used to manage transactions in database. These are used to manage the changes made by DML statements. It also allows statements to be grouped together into logical transactions.

- **Commit command** : Commit command is used to permanently save any transaction into database. **Syntax** : commit;
- **Rollback command** : This command restores the database to last committed state. It is also use with savepoint command to jump to a savepoint in a transaction. **Syntax**: rollback to savepoint-name;
- **Savepoint command :** This command is used to temporarily save a transaction so that you can rollback to that point whenever necessary. **Syntax**: savepoint savepoint-name;

Examples:

```
INSERT into class values(5,'Rahul');
COMMIT;
UPDATE class set name='Abhijit' where id='5';
SAVEPOINT A;
INSERT into class values(6,'Chris');
SAVEPOINT B;
ROLLBACK TO A;
```

Thank You
End of Lab 7

Assignment

Reference Tables

- **EMPLOYEE** table with the attributes:

  ID, LAST_NAME, FIRST_NAME, MIDDLE_NAME, FATHER_NAME, MOTHER_NAME, SEX, HIRE_DATE, DOB, ADDRESS, CITY, STATE, ZIP, PHONE, PAGER, SUPERVISOR_ID, INJECTED_DATE

- **SCHOOL** table with the attributes:

  ID, NAME, INJECTED_DATE

- **EMPLOYEE _ALIGNMENT** table with the attributes:

  EMPLOYEE_ID, SCHOOL_ID, INJECTED_DATE

- **JOB** table with the attributes:

  ID, NAME, TITLE, SALARY, BONUS , INJECTED_DATE

- **EMPLOYEE _PAY** table with the attributes:

  EMPLOYEE_ID, JOB_ID, INJECTED_DATE

Assignment

1. Create a view to include all employee and school, but hide INJECTED_DATE
2. Create a view to include all employee and JOB, but hide INJECTED_DATE
3. Create a view to include all employee and PAY, but hide INJECTED_DATE
4. Create a view to include school name, number of employees and average salary by each school
5. Grant only select privilege to another user for EMPLOYEE table
6. Grant only update privilege to another user for SCHOOL table
7. Grant only delete privilege to another user for JOB table
8. Create a synonym to include all employee and school
9. Create a synonym to include all employee, School and JOB
10. Create a synonym to include all employee , School , JOB and PAY
11. Create index on Sex in Employee

Assignment

12. Create index on First Name, Middle Name and Last Name in Employee
13. Save the transaction as save point.
14. Rename the index created for question 12
15. Drop the index renamed for question 14
16. Rename the view created for question 3
17. Drop the view renamed in question 16
18. Commit transaction done for the steps from 1 to 17
19. Drop the view created for question 1
20. Rollback the entire transaction