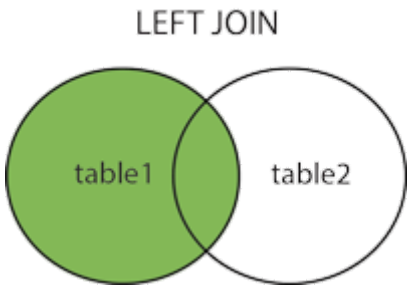
Intermediate SQL

Join Expressions

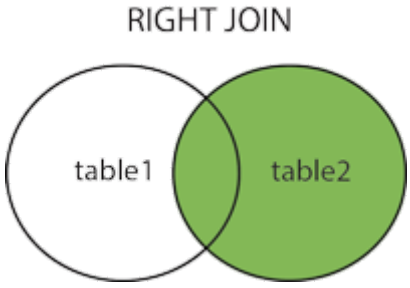
1. Join Conditions
    1. `join using` - a form of natural join that only requires values to match on specified attributes.
    2. `join on` - allows a general predicate over the relations being joined (like a **where** clause).
2. Outer Joins
    1. The **left outer join** preserves tuples only in the relation named before (to the left of) the left outer join operation.

    > **SELECT** *column_name(s)*
    > **FROM** *table1*
    > **LEFT JOIN** *table2*
    > **ON** *table1.column_name = table2.column_name*;

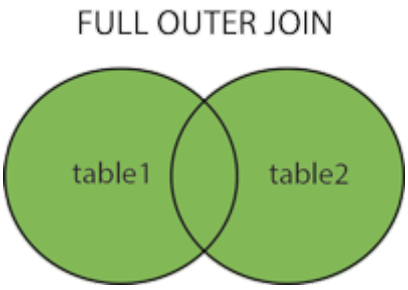
    2. The **right outer join** preserves tuples only in the relation named after (to the right of) the right outer join operation.

    > **SELECT** *column_name(s)*
    > **FROM** *table1*
    > **RIGHT JOIN** *table2*
    > **ON** *table1.column_name = table2.column_name*;

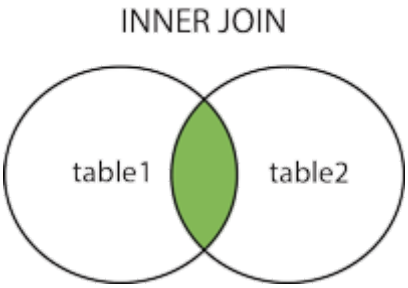
    3. The **full outer join** preserves tuples in both relations.

    > **SELECT** *column_name(s)*
    > **FROM** *table1*
    > **FULL OUTER JOIN** *table2*
    > **ON** *table1.column_name = table2.column_name*
    > **WHERE** *condition*;



3. Join Types and Conditions
    1. Inner Join (normal join/ join)

    > **SELECT** *column_name(s)*
    > **FROM** *table1* **JOIN** *table2*
    >
    > **ON** *table1.column_name = table2.column_name*;



    2. Outer Join (see above)

Views

Views in SQL are kind of virtual tables. A view also has rows and columns as they are in a real table in the database. We can create a view by selecting fields from one or more tables present in the database. A View can either have all the rows of a table or specific rows based on certain condition.

1. View Definition - `create view`

    > **create view** v **as** ;

2. Using Views in SQL Queries

    Example:

    ```
    create view departments total salary(dept name, total salary) as
    select dept name, sum (salary)
    from instructor
    group by dept name;
    ```

3. Materialized Views
    - If the actual relations used in the view definition change, the view is kept up-to-date. Such views are called **materialized views**.
4. Update of a View

    In general, an SQL view is said to be updatable (that is, inserts, updates or deletes can be applied on the view) if the following conditions are all satisfied by the query defining the view:

    - The **from** clause has only one database relation.
    - The **select** clause contains only attribute names of the relation, and does not have any expressions, aggregates, or **distinct** specification.

    - Any attribute not listed in the **select** clause can be set to null; that is, it does not have a not null constraint and is not part of a primary key.
    - The query does not have a **group** by or **having** clause.

Transactions

A transaction consists of a sequence of query and/or update statements.

One of the following SQL statements must end the transaction:

- **Commit work** commits the current transaction; that is, it makes the updates performed by the transaction become permanent in the database. After the transaction is committed, a new transaction is automatically started.
    - saving changes to a document that is being edited
- **Rollback work** causes the current transaction to be rolled back; that is, it undoes all the updates performed by the SQL statements in the transaction. Thus, the database state is restored to what it was before the first statement of the transaction was executed.
    - similar to quitting the edit session without saving changes

Integrity Constraints

Integrity constraints ensure that changes made to the database by authorized users do not result in a loss of data consistency. Thus, integrity constraints guard against accidental damage to the database.

1. Constraints on a Single Relation

   The create table command may also include integrity-constraint statements. In addition to the primary-key constraint, there are a number of other ones that can be included in the create table command. The allowed integrity constraints include
    - not null
    - unique
    - check(<predicate>)
2. Referential Integrity (subset dependencies)
    - foreign key

Authorization

We may assign a user several forms of authorizations on parts of the database. Authorizations on data include:

- Authorization to read data.
- Authorization to insert new data.
- Authorization to update data.
- Authorization to delete data.

Each of these types of authorizations is called a **privilege**.

1. Granting and Revoking of Privileges

   The SQL standard includes the privileges select, insert, update, and delete. The privilege all privileges can be used as a short form for all the allowable privileges.
    - Granting

> **grant** <privilege list>
> **on** <relation name or view name>
> **to** <user/role list>;

The user name public refers to all current and future users of the system. Thus, privileges granted to public are implicitly granted to all current and future users.

- Revoking

> **revoke** <privilege list>
> **on** <relation name or view name>
> **from** <user/role list>;

2. Roles

   Each database user is granted a set of roles(which may be empty) that she is authorized to perform. Any authorization that can be granted to a user can be granted to a role. Roles are granted to users just as authorizations are.

   - Create Role

   > **create role** instructor;

3. Authorization on Views

   A user who creates a view does not necessarily receive all privileges on that view. She receives only those privileges that provide no additional authorization beyond those that she already had.

4. Authorization on Schema

   Only the owner of the schema can carry out any modification to the schema, however, SQL includes a **references** privilege that permits a user to declare foreign keys when creating relations.

   > **grant references**(*attribute*) **on** *relation* **to** *name/role*;

5. Transfer of Privileges

   A user who has been granted some form of authorization may be allowed to pass on this authorization to other users. If we wish to grant a privilege and to allow the recipient to pass the privilege on to other users, we append the **with grant option** clause to the appropriate grant command.

   > **grant select on** *relation* **to** *name/role* **with grant option**;

6. Revoking of Privileges

   Revocation of a privilege from a user/role may cause other users/roles also to lose that privilege. This behavior is called *cascading revocation*. However, the revoke statement may specify restrict in order to prevent cascading revocation:

   > **revoke select on** *relation* **from** *name/role* **restrict**;