INTERMEDIATE SQL

Operations

- Join Expressions
- Views
- Index
- Integrity Constraints
- SQL Data Types and Schemas
- Authorization

Joined Relations

- Join operations take two relations and return as a result another relation.
- The join operations are typically used as subquery expressions in the **from clause**

Joined Relations types

- Inner Join
- Left Outer Join
- Right Outer Join
- Full Outer Join

Full outer Join Syntax

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

Example Table Customers

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

Example Table Orders

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10308	2	7	1996-09-18	3
10309	37	3	1996-09-19	1
10310	77	8	1996-09-20	2

Example Query

- SELECT Customers.CustomerName, Orders.OrderID
  FROM Customers
  FULL OUTER JOIN Orders ON

Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName;

Output

- Full Outer Join

CustomerName	OrderID
Null	10309
Null	10310
Alfreds Futterkiste	*Null*
Ana Trujillo Emparedados y helados	10308
Antonio Moreno Taquería	*Null*

Views

- A view provides a mechanism to hide certain data from the view of certain users.
- Any relation that is not of the conceptual model but is made visible to a user as a “virtual relation” is called a view.

Syntax of view

- CREATE VIEW *view_name* AS
  SELECT *column1*, *column2*, ...
  FROM *table_name*
  WHERE *condition*;

Example Views

- CREATE VIEW [Brazil Customers] AS
  SELECT CustomerName, ContactName
  FROM Customers
  WHERE Country = 'Brazil';

SQL Updating a View

- A view can be updated with the CREATE OR REPLACE VIEW statement.
  SQL CREATE OR REPLACE VIEW Syntax
  CREATE OR REPLACE VIEW *view_name* AS
  SELECT *column1*, *column2*, ...
  FROM *table_name*
  WHERE *condition*;

Example

- The following SQL adds the "City" column to the "Brazil Customers" view:
- CREATE OR REPLACE VIEW [Brazil Customers] AS
  SELECT CustomerName, ContactName, City
  FROM Customers
  WHERE Country = 'Brazil';

SQL Dropping a View

- A view is deleted with the DROP VIEW statement.
- Syntax:
- DROP VIEW *view_name*;
- Example:
- DROP VIEW [Brazil Customers];

INDEX

- The CREATE INDEX statement is used to create indexes in tables.
- Indexes are used to retrieve data from the database more quickly than otherwise.

Syntax

- CREATE INDEX *index_name*
  ON *table_name* (*column1*, *column2*, ...);

CREATE UNIQUE INDEX Syntax:

- CREATE UNIQUE INDEX *index_name*
  ON *table_name* (*column1*, *column2*, ...);

CREATE INDEX Example

CREATE INDEX idx_lastname
ON Persons (LastName);

If you want to create an index on a combination of columns,

CREATE INDEX idx_pname
ON Persons (LastName, FirstName);

Drop Index

- ALTER TABLE *table_name*
  DROP INDEX *index_name*;

Integrity Constraints

- **Definition:**
- Integrity Constraints are the protocols that a table's data columns must follow.
- These are used to restrict the types of information that can be entered into a table.

Types

- Primary key
- Foreign key
- NOT NULL CONSTRAINTS
- Unique Key
- Primary Key Vs. Unique Key

Primary Key

Syntax

```
Create table table_name
(
Column_name1 datatype NOT NULL,
Column_name2 datatype,
.....
Column_namendatatype,
 PRIMARY KEY (column_name1)
);
```

Example code

- create table student(id int not null,name varchar(20),marks int,grade varchar(5),**primary key(id)**);
- select * from student;
- Output:
- The id field is set to primary key constraints.

Foreign Key

Syntax:

```
CREATE TABLE table_name1
(
 Column_name1 datatype NOT NULL,
Column_name2 datatype ,
 ...
Column_nameNdatatype ,
  PRIMARY KEY (Column_name1)
);
```

Cont.,

```
CREATE TABLE table_name2
(
  Column_name1 datatype NOT NULL,
  Column_name2 datatype  NOT NULL,
  Column_name3 datatype  ,
  ....
Column_nameNdatatype ,
 PRIMARY KEY (Column_name1, Column_name2),
 FOREIGN KEY (Column_name1) REFERENCES
table_name2 (Column_name1) ON DELETE CASCADE
);
```

Example

```
CREATE TABLE studnew
(
stu_id   INT NOT NULL,
stu_name varchar(20),
stu_class Varchar(20),
PRIMARY KEY (stu_id)
);
```

Cont.,

```
CREATE TABLE classnew
(
stu_id   INT NOT NULL,
class_id INT NOT NULL,
PRIMARY KEY (stu_id, class_id),
FOREIGN KEY (stu_id) REFERENCES stud (stu_id)
ON DELETE CASCADE
);
```

NOT NULL CONSTRAINTS

- The not null constraint tells a column that it can't have any null values in it.

Syntax:

```
Create table table_name
(
Column_name1 datatype NOT NULL,
Column_name2 datatype,
……
Column_namendatatype,
);
```

Syntax

Syntax:

Create table table_name

(

Column_name1 datatype NOT NULL,

Column_name2 datatype,

……

Column_namendatatype,

);

Example Code

- create table student1(id int not null,name varchar(20),marks int,grade varchar(5));

Unique Key

- A collection of one or more table fields/columns that uniquely identify a record in a database table is known as a unique key.

Syntax

```
CREATE TABLE Table_name (
Column_Name1 DataType NOT NULL UNIQUE,
Column_Name2 DataType NOT NULL,
Column_Name3 DataType,
    .....
Column_NameNDataType
);
```

Example Code

- CREATE TABLE Student_DB (
- S_ID int NOT NULL UNIQUE,
- L_Name varchar(255) NOT NULL,
- F_Name varchar(255),
- Age int);

Dropping Constraints

- DROP a UNIQUE Constraint

Syntax

- ALTER TABLE TABLE_NAME DROP CONSTRAINT UNIQUE _KEY FIELD;

Code

ALTER TABLE Student_DB

DROP CONSTRAINT UKlname;

Cont.,

Dropping Primary Key Constraints

- Code
- ALTER TABLE shops drop pkkey;

Dropping Foreign key Constraint

- Code
- ALTER TABLE Orders
  DROP FOREIGN KEY FK_pid;
- ALTER TABLE Orders
  DROP CONSTRAINT FK_pid;

Authorization

- Authorization is a privilege provided by the Database Administer. Users of the database can only view the contents they are authorized to view.

Types

- **Primary Permission -** This is granted to users publicly and directly.
- **Secondary Permission -** This is granted to groups and automatically awarded to a user if he is a member of the group.
- **Public Permission -** This is publicly granted to all the users.
- **Context sensitive permission -** This is related to sensitive content and only granted to a select users.