.NET Full Stack

Development Program

Day 8 SQL Server Intro

# Outline

## Data Modeling

* ER Diagram

## Normalization

* Database

## SQL

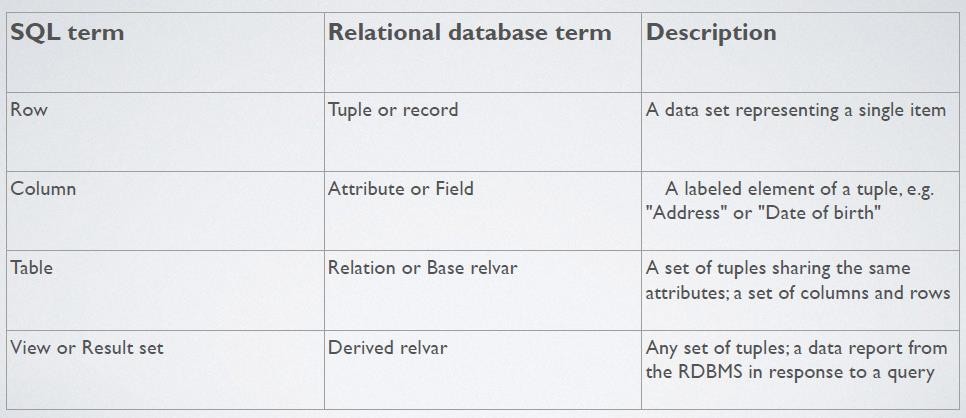
* Constraints

# Data Modeling

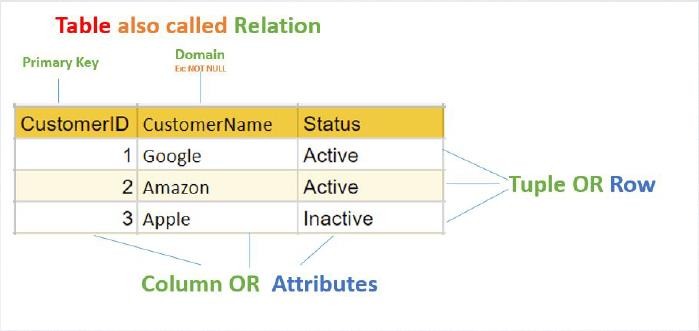
## Data Model is the process of analyzing **business requirement**, designing and creating physical instance(a plan or a blueprint) for the database or data warehouse.

* It is a stage in SDLC(Software Development Life Cycle)
* It also impacts the user interface

# Relational Database Terminology



Relational Database Terminology



# Data Modeling Terminology

## Entity

* + An item that can exist independently or

uniquely identified

## Attribute

* + Column label(name)

## Domain

* + Set of valid values for an attribute

## Relationship

* + How entities relate

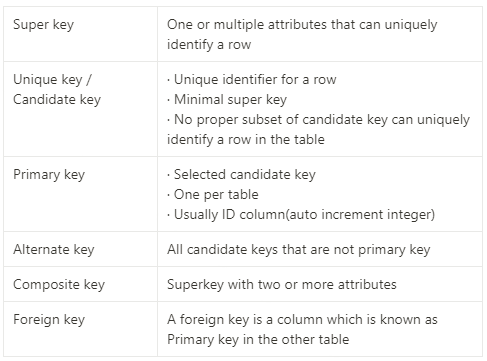
## Degree

* + How many entities in a relationship

## Cardinality

* + Measure of participation

# Keys Used in Data Modeling



Key Used in Data Modeling



# Cardinality and Degree

## Cardinality is the number of times the entity participates in the relationship

* + One-to-One: One element in entityA may link to one element in entityB and vice versa
  + One-to-Many: One element in entityA may link to many elements in entity but one element in entity may only link to one element in entityA
  + Many-to-One: Reverse A and B in one-to-many
  + Many-to-Many: One element in entityA may link to any number of elements in entity and vice

versa

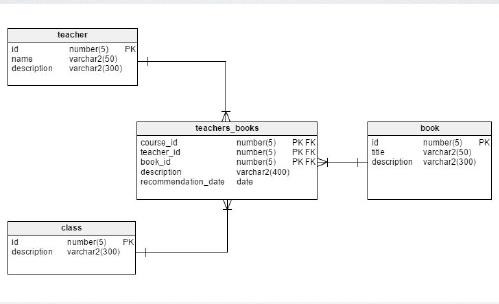
## Degree

* + Degree is the number of entities involved in the relationship and it is usually 2(binary

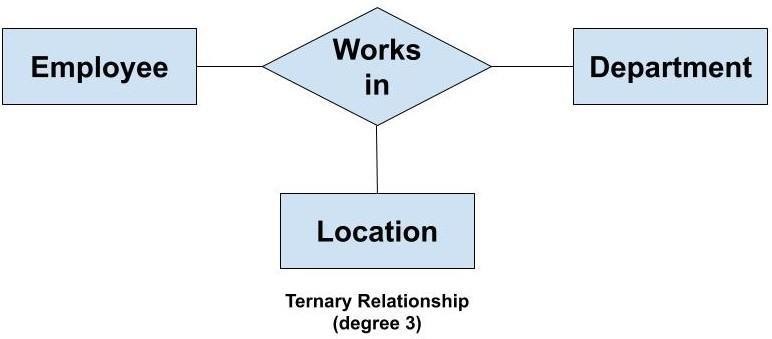
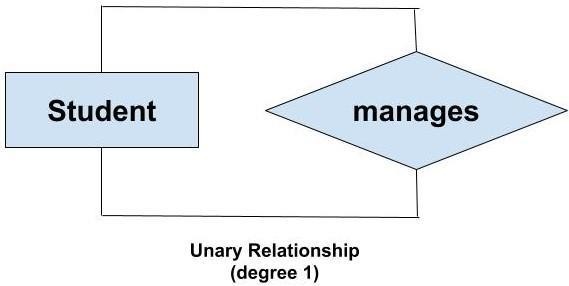
relationship) however Unary and higher degree relationships can exists.

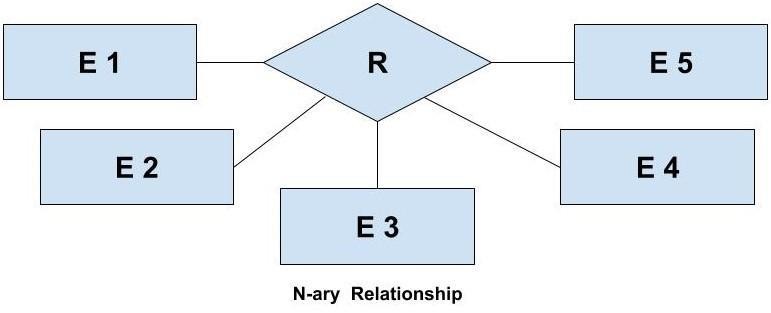
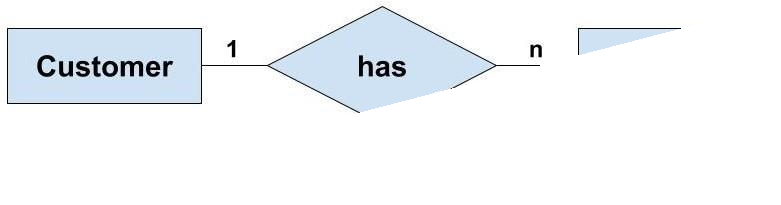
* Cardinality != Degree

# Cardinality and Degree



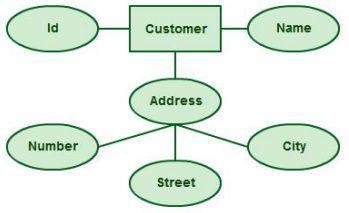
Cardinality and Degree





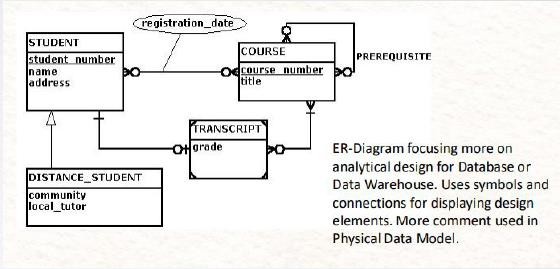
# ERD

## ER-Diagram

* + Entity Relationship Diagram
  + Used to create or design a blueprint of the Database

or Data Warehouse

* + Design Entities, attributes and show relationships

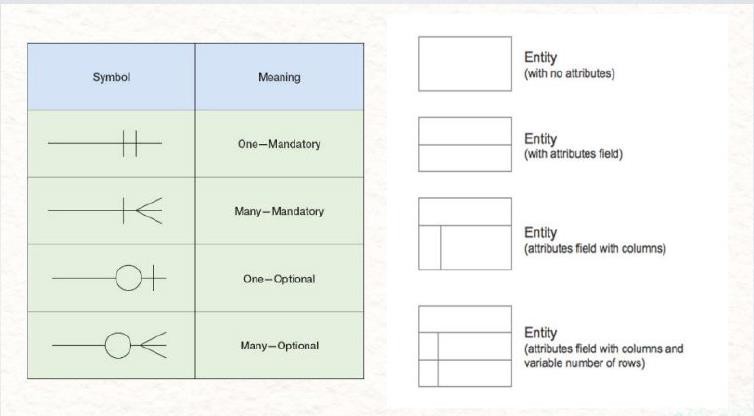


# Crow’s Foot Notation

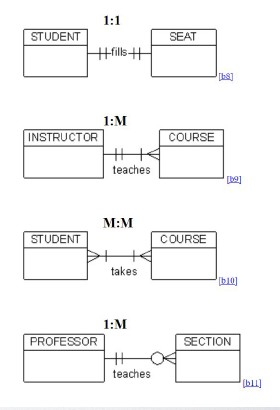
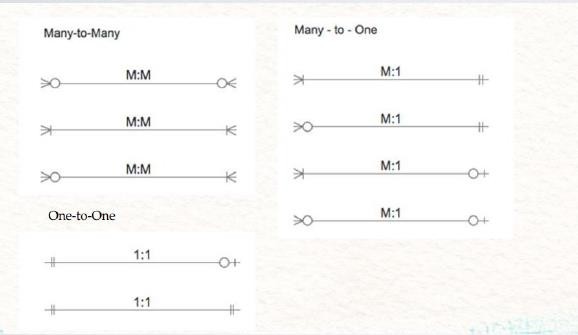
## Connection Symbols display Relationships

* Entity and Attributes in Table like format

# Crow’s Foot Notation



Crow’s Foot Notation



# Convert ERD to Tables

### Each entity type becomes a table

* Each single-valued attribute becomes a column

### Derived attributes are ignored/computed column

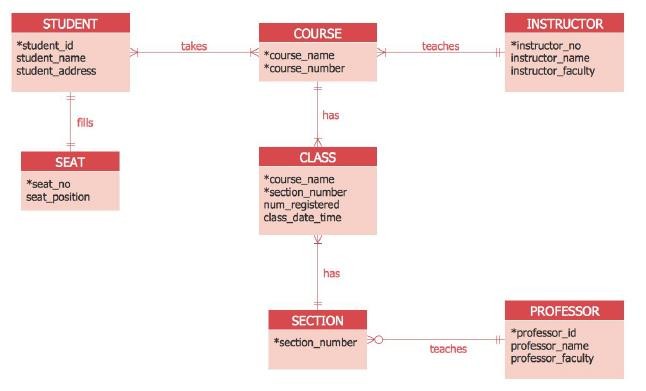
* Multi-valued attributes are represented by a separate table

### Use Conjunction Table to break up the Many-to-Many relationship

* The key attribute of the entity type becomes the primary key or unique key of the

table

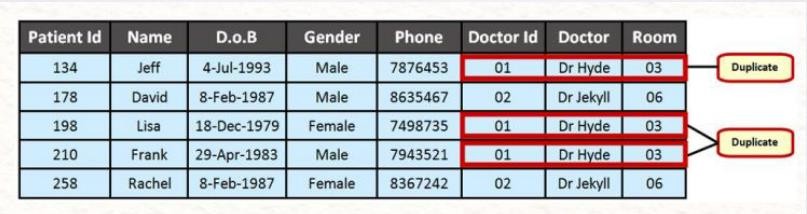
# Convert ERD to Tables



Normalization

## Redundancy

* + Values repeated unnecessarily in multiple records or fields within one or more tables



# Normalization

## Anomaly

* When an attempt is made to modify(update, insert into, or delete from) a relation, the following undesirable side-effects may arise in relations that have not been sufficiently normalized

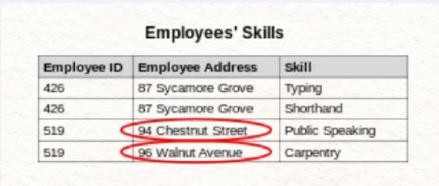
## Anomaly is the issue that may occur because of redundancy

* Types: Update, Insertion and Deletion

# Normalization

## Anomaly Update

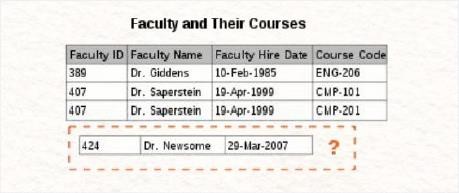
* The same information can be expressed on multiple rows; Therefore, updates to the relation may result in logical inconsistencies



# Normalization

## Anomaly Insertion

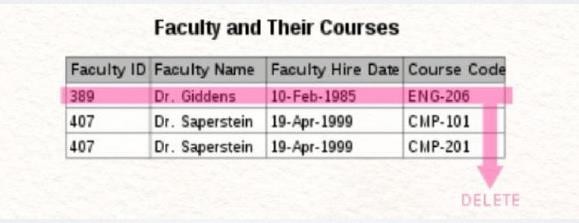
* There are circumstances in which certain facts cannot be recorded at all



# Normalization

## Anomaly Deletion

* Under certain circumstances, deletion of data representing certain facts necessitates deletion of data representing completely different facts



# Normalization

## Main goal of normalization

* Reduce redundancy, avoid anomaly and create a well-structured series of table without error or inconsistencies

## Minimize redesign when extending the database structure

* + New type of data can be accommodated without changing existing structure too much

## Ensure data dependencies are properly enforced by data integrity

constraints(Entity Integrity, Referential Integrity, Domain Integrity)

# Normalization

## Functional Dependencies

* + Functional Dependencies are how different attributes relate in a table
  + At this level, we focus on individual tables
  + We see how individual attributes relate to

the keys in the table

* + - Primary Key & Candidate Keys = Prime Attributes
    - Attributes that aren’t keys = Non- Prime Attributes

## Types of Dependencies

* + Full Dependencies – Depends on all prime

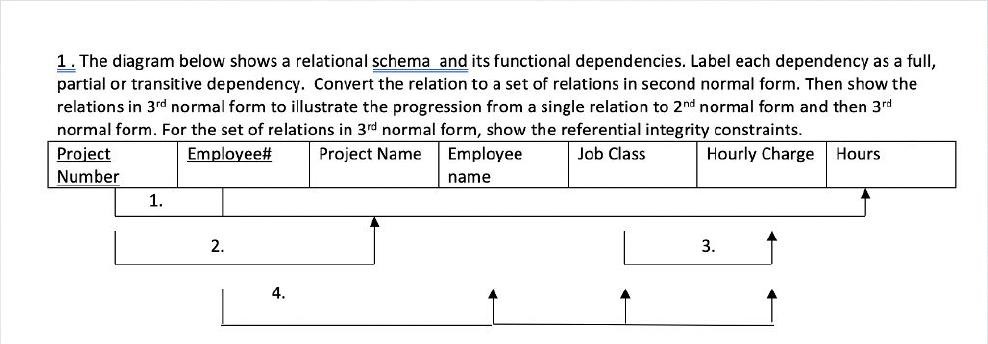
attributes fully

* + Partial Dependencies – Depends on some

Prime Attributes

* + Transitive Dependencies – Depends on an attribute that depends on a Prime Attribute

# Normalization



Normalization

## First Normal Form

* + Each table cell should contain a single value
  + Each record needs to be unique

## Second Normal Form

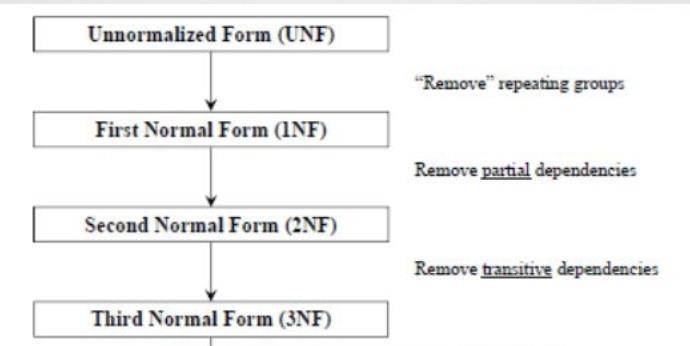
* + Meets all of 1NF
  + Makes sure all non-prime attributes are fully dependent on a prime attribute

## Third Normal Form

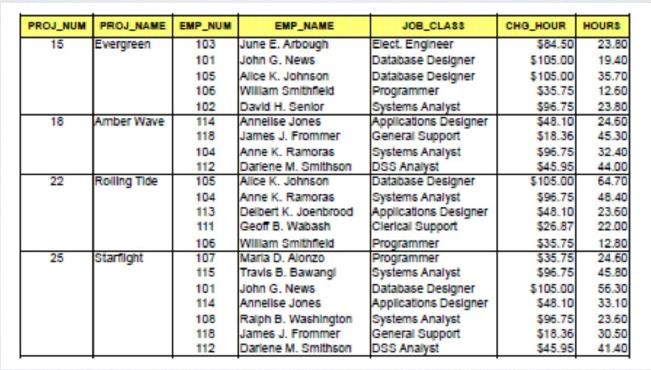
* + Meets 1NF and 2NF
  + Every non-prime attribute is non-transitively dependent on the prime attributes

# Normalization

* Process

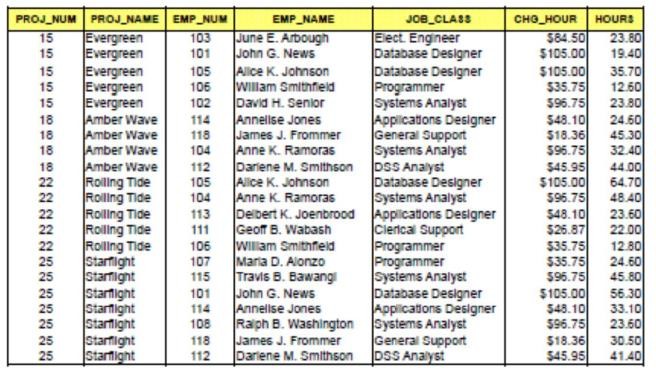


# Normalization

* UNF

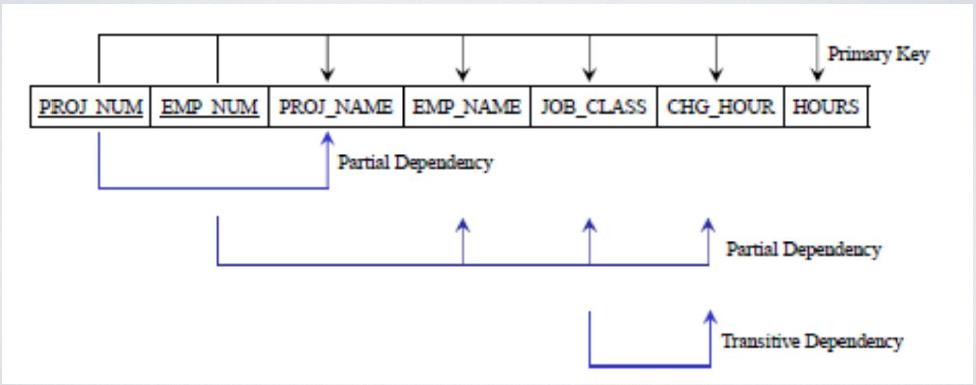
# Normalization

* After 1NF



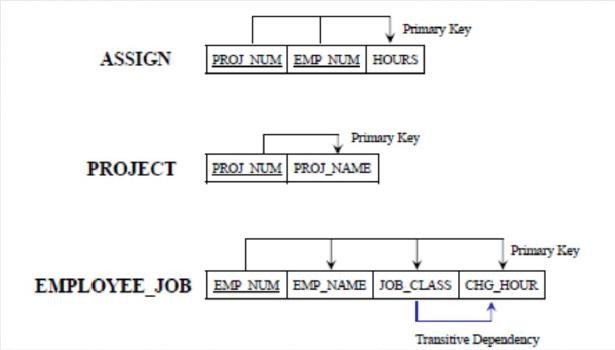
# Normalization

* Dependencies



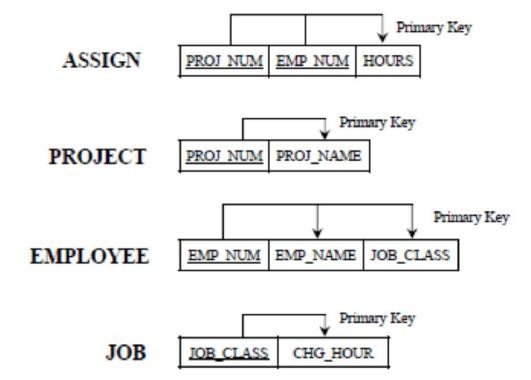
# Normalization

* After 2NF(Remove partial dependencies)



# Normalization

* After 3NF(Remove transitive dependency)



# Normalization

## When do we normalize?

* + Usually normalize in database with lots of read/writes
  + Not too much fetching/only need to fetch a small subset of the data
  + Data integrity is crucial especially because there us a lot of user input

## Cons

* + Normalization is an expensive process
  + Designing can be difficult – Good for final design, not testing
  + More tables leads to more time
  + Joins are costly, having more tables can cause slowing

# Database Integrities

### Entity Integrity

* + Design of the table or entity
  + String PK with no nulls or repeats

### User-Defined Integrity

* + Rules or constraints applied by the user to maintain rules of design

### Domain Integrity

* + Correct and proper domains specified with proper use of columns

### Referential Integrity

* + Proper FK setup with proper PK reference
  + Good design for connection and joins

# Database Management System



Relational Database vs. Non-Relational Database

## Relational Database(SQL)

* + Traditional way of storing data
  + Data is stored in tables with rows and columns
  + Rigid schema with well-defined relationships
  + Difficult to scale
  + Examples: SQL Server, Oracle, MySQL

## Non-Relational Database(NoSQL)

* + Developed more recently
  + Data can be stored in a variety of formats(JSON documents, key-value pairs, wide-column,

graphs)

* + Flexible schema with loose relationships
  + Easily scalable
  + Examples: MongoDB, Redis

# SQL Introduction

SQL(Structured Query Language)

## Data Definition Language(DDL)

* (DML)

## (DCL)

* (DQL)

# SQL

## DDL – Data Definition Language

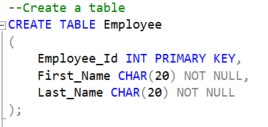
* Create

## Alter

* Drop
* Truncate

# Create

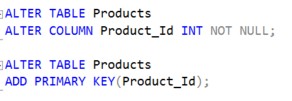
## This command builds a new table and has a predefined syntax.



* In this example, the string CHAR is used to specify the data type. Other data types can be DATE, INT, DECIMAL etc.

# Alter

## An Alter command modifies an existing database table. This command can add up additional column, drop existing columns and even change the data type of columns involved in a database table.



* In this example, we added a unique primary key to the table to add a constraint and enforce a unique value. The constraint “Product\_Id” is a primary key and is on the Products table.

# Drop

* A drop command is used to delete objects such as a table, index or view. A DROP statement **cannot be rolled back**, so once an object is destroyed, there’s no way to recover it.





# Truncate

* Similar to DROP, the TRUNCATE statement is used to quickly remove all records from a table. However, unlike DROP that completely destroys a table, TRUNCATE preserves it’s full structure to be reused later.





# SQL

## DML – Data Manipulation Language

* Insert

## Update

* Delete

# DML

## Insert Statements

* + Insert statements are used to input data into tables
  + The order of data specified should match order of columns
  + If you don’t know the order of columns, specify each column name in the insert statement

## For example

* + Inserting data matching order
    - Insert into TableName values(1, ‘Name’),(2, ‘Name’);
  + Inserting data without knowing matching order
    - Insert into TableName(Col2, Col1) values(‘Name’, 1),(‘Name’, 2);

# DML

## Update Statements

* + Update statements are used to change or modify data inside a table
  + Specify single row depending on statement
  + Use unique identifiers to get correct rows

## For Example

* + Update using unique column
    - Update TableName Set Name = ‘Tim’ Where ID = 2;
  + Update using non-unique column
    - Update TableName Set Name = ‘Hal’ Where Name = ‘Bob’

# DML

## Delete Statements

* + Delete statements are used to remove specific rows inside a table
  + Use unique values to identify the correct rows to delete
  + Delete leave logs and continue identity values

## For Example

* + Deleting rows using specific unique values
    - Delete from TableName Where ID = 1;
  + Deleting rows using non-unique values
    - Delete From TableName Where Name = ‘Jim’

# SQL

## DCL – Data Control Language

* DCL is syntax used to control what permission a user can have

## You can create Roles that users can be grouped into a specific permission there

* You can choose what tables someone in a role can access and even what statements can be performed

# DCL

## Grant

* + Used to “grant” or give permissions to a user or role

## Example:

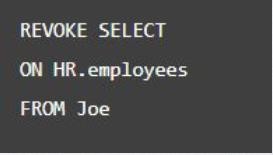
* + Create table permission to a role
    - Grant Create Table to TestRole
  + Add user to the role
    - Exec sp\_addrolemember ‘TestRole’, ‘TestUser’

# DCL

## Revoke

* + Revoke is used to take away permissions given, whether they are Grant or Deny permissions.

## Example

* + Revoking grant permission
  + Revoke Create Table to TestRole

# DCL

## Deny

* Deny is preventing someone from having access at all.

## Deny is not the same as revoke, as revoke is

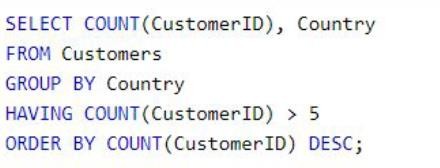
made to take back, deny is made to say NO

## If grant and deny permission are given to the same role, deny will take over

* Example
  + Deny Create Table to TestRole

# SQL

## DQL – Data Query Language

* Select From Where
  + Select: Pick up which column of data you’d like to fetch
  + From: Select which table or data set to fetch.
  + Where: Specific a criteria to sort data by use operators(Filter).
    - Operators: In, Or, And

## Group by, Having, Order by

* + Group by – Used to combine similar values in column
  + Having – filter conditions for aggregate only
  + Order by – display the data by order by a specific column

# DQL Execution Order



Constraint

## A constraint is usually associated with a table and is created with a CREATE CONSTRAINT SQL statement.

* They define certain properties that data in a database must comply with

# Constraint

## Key Constraints

* + Primary Key
    - 1 per table
    - Unique Clustered index
    - Not Null
  + Unique Key
    - 999 per table
    - Unique Non-Clustered index
    - 1 Null Allowed
  + Foreign Key
    - Cannot exist before PK
    - Must be deleted before PK

## Other Constraints

* + Null, Not Null
    - Are nulls allowed
  + Check
    - Data must meet rule
  + Default
    - If nothing, then this
  + Data types
    - Char(2) –States (NY, CA…)
    - Varchar(10) – Names…
    - Money -- Money

# Constraint

## Adding constraints to a table

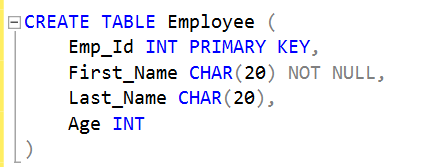
* + Constraints: used to specify rules for the data in table
  + Type: NOT NULL, UNIQUE, PRIMARY KE

## How?

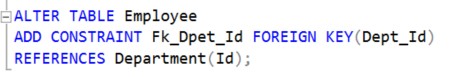
* + Create table then alter to add constraints
  + Add constraint as we specify the column in a table
  + Add constraint in the same create statement, after we specified the table

# Constraint

## Add constraint as we specify the column in a table



* Add constraint in the same create statement after we specified the table



## Create a table then alter it to add constraints

Questions?