## Introduction and foundation and Key definitions

### Database and DBMS

A **database** can be defined as a collection of related data items within a specific business process or problem setting has a target group of users and applications

A Database Management System (**DBMS**), is the software package used to define, create, use and maintain a database consists of several software modules

DBMS > Databases > Tables > Columns and Rows > Values (data)

### File vs. Database Approach to Data Management (Advantage use DB)

* File approach

duplicate or redundant information will be stored

danger of inconsistent data

strong coupling between applications and data

hard to manage concurrency control

hard to integrate applications aimed at providing cross-company services

* Database approach

superior to the file approach in terms of efficiency, consistency and maintenance

loose coupling between applications and data

facilities provided for data querying and retrieval

### Elements of a Database System

Database Model

Data Model

Catalog

Database Users

Database Languages

#### Database Model

A database model fundamentally determines in which manner data can be stored, organized and manipulated. The most popular example of a database model is the relational model

##### Database state

Database state represents the data in the database at a particular moment

also called the current set of instance

typically changes on an ongoing basis

#### Data Model

A data model provides a clear and unambiguous description of the data items, their relationships and various data constraints.

There are four types of data model or schema:

1. Conceptual
2. Logical
3. Physical
4. External

##### Conceptual

A conceptual data model provides a high-level description of the data items with their characteristics and relationships

A communication instrument between information architect and business user

Should be implementation independent, user-friendly, and close to how the business user perceives the data

usually represented using an Entity Relationship **(ER) model**, or an object-oriented model

##### Logical

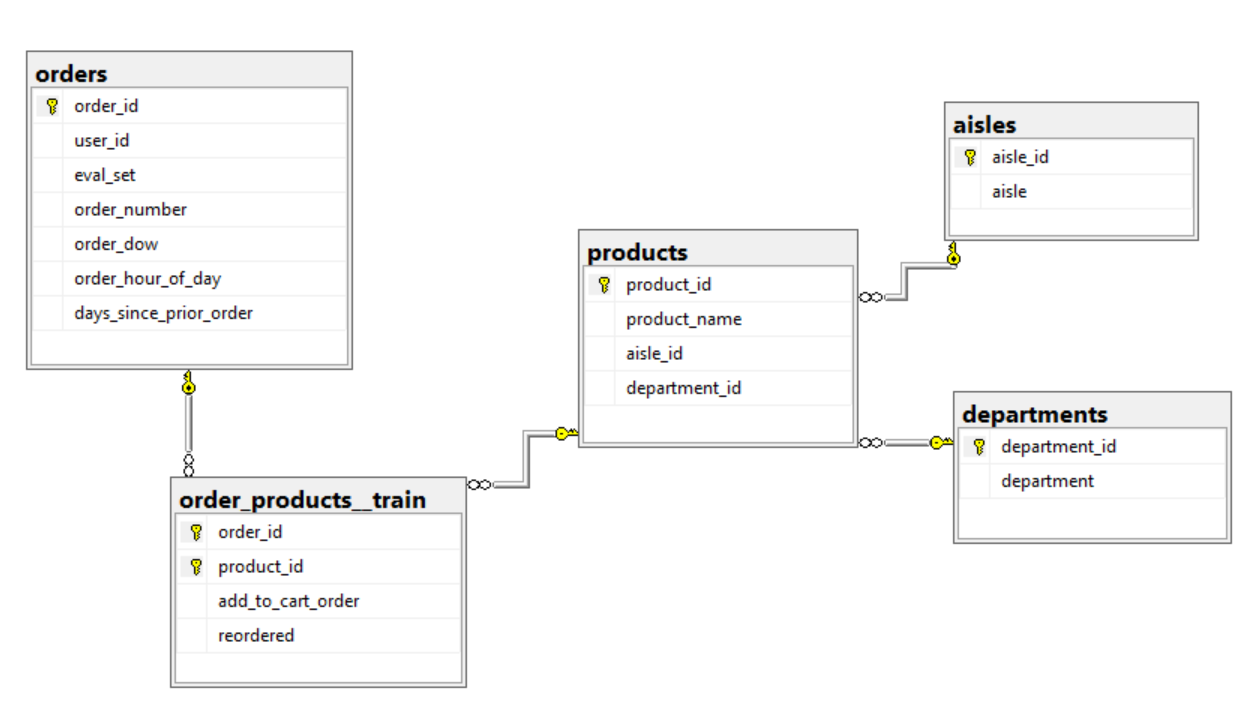
Logical data model is a translation or mapping of the conceptual data model towards a specific implementation environment

Can be a Hierarchical, CODASYL, Relational, Object-oriented, Graph, XML or NoSQL model

Clearly describes which data is stored where, in what format, which indexes are provided to speed up retrieval, etc.

Highly DBMS specific

**Ex:**



##### Physical

Physical data model represents how the model will be built in the database

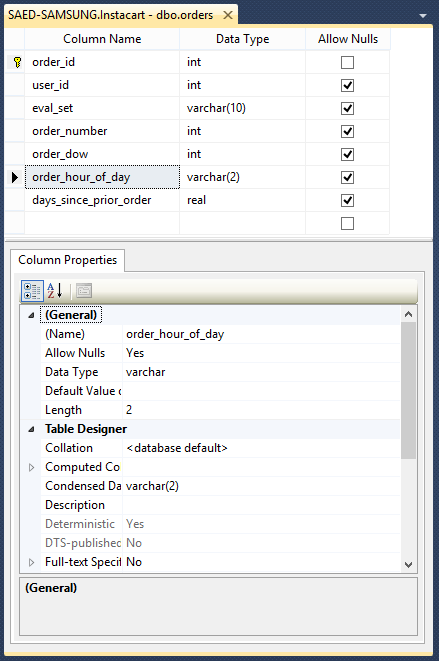
A physical database model shows all table structures, including column name, column data type, column constraints, primary key, foreign key, and relationships between tables

Convert entities into tables

Convert relationships into foreign keys

Convert attributes into columns

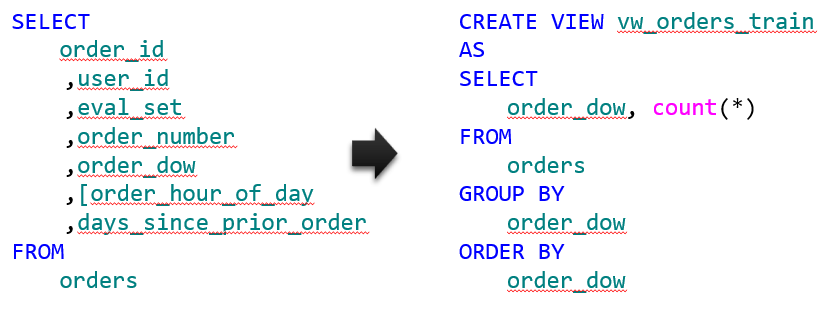
**Ex:**



##### External

External data model contains various subsets of the data items in the logical model, also called views, tailored towards the needs of specific applications or groups of users

**Ex:**

****

#### Catalog

Heart of the DBMS

Contains the data definitions, or metadata, of your database application

Stores the definitions of the views, logical and internal data models, and synchronizes these three data models to make sure their consistency is guaranteed

#### Database Users

**Information architect** designs the conceptual data model

**Database designer** translates the conceptual data model into a logical and physical data model

**Database administrator (DBA)** is responsible for the implementation and monitoring of the database

**Application developer** develops database applications in a programming language such as Java or Python

**Business user** will run these applications to perform specific database operations

#### Database Languages

**Data Definition Language (DDL)** is used by the DBA to express the database's external, logical and internal data models

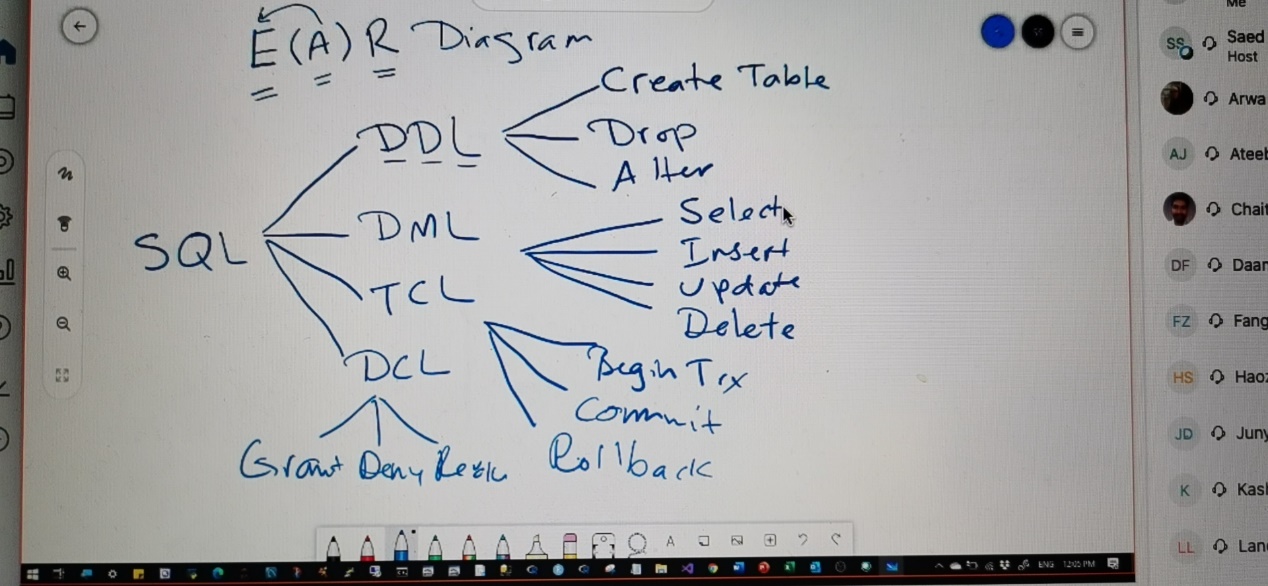
definitions are stored in the catalog

**Data Manipulation Language (DML)** is used to retrieve, insert, delete, and modify data

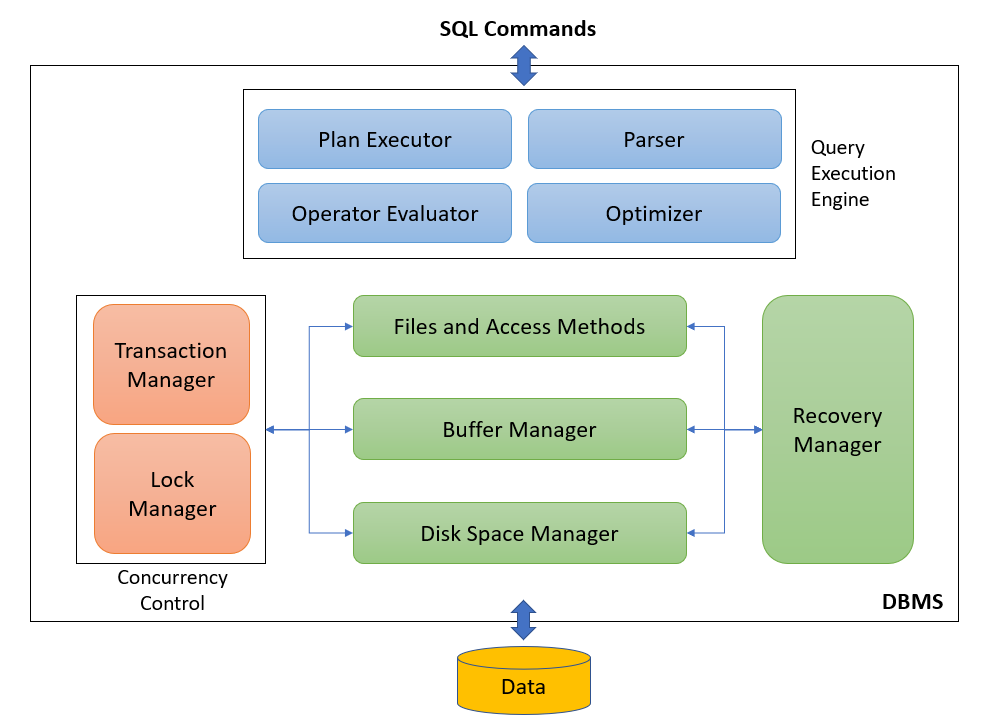
DML statements can be embedded in a programming language, or entered interactively through a front-end querying tool

**Structured Query Language (SQL)** offers both DDL and DML statements for relational database systems

#### ER & SQL.jpg



## DBMS



### Query Execution

#### query plan

A query plan is a set of steps that the database management system executes in order to complete the query.

#### reason

The reason we have query plans is that the SQL you write may declare your intentions, but it does not tell SQL the exact logic flow to use.

#### query optimizer

The query optimizer determines the exact logic flow and the result is the query plan.

### Buffer Manager

The buffer manager is the software layer that is responsible for bringing pages from physical disk to main memory as needed.

The buffer manages the available main memory by dividing the main memory into a collection of frames.

The goal of the buffer manager is to ensure that the data requests made by programs are satisfied by copying data from secondary storage devices into buffer.

Data must be in RAM for DBMS to operate on it. Buffer manager hides the fact that not all data is in RAM.

### Transactions, Recovery, Concurrency Control

DBMS must support **ACID** (Atomicity, Consistency, Isolation, Durability) properties

**Errors** may occur in the DBMS or its environment

Majority of databases are multi-user databases and concurrent access to the same data may induce different types of anomalies

#### Transactions

Transaction is a set of database operations induced by a single user or application, that should be considered as **one undividable unit of work**

A transaction is a **sequence of reads and writes**, e.g., For example, transfer between two bank accounts of the same customer

Transaction always ‘**succeeds’** or ‘**fails’** in its entirety

Transaction renders database from one consistent state into another **consistent state**

##### Transaction Lifecycle

Transactions can be specified implicitly or explicitly

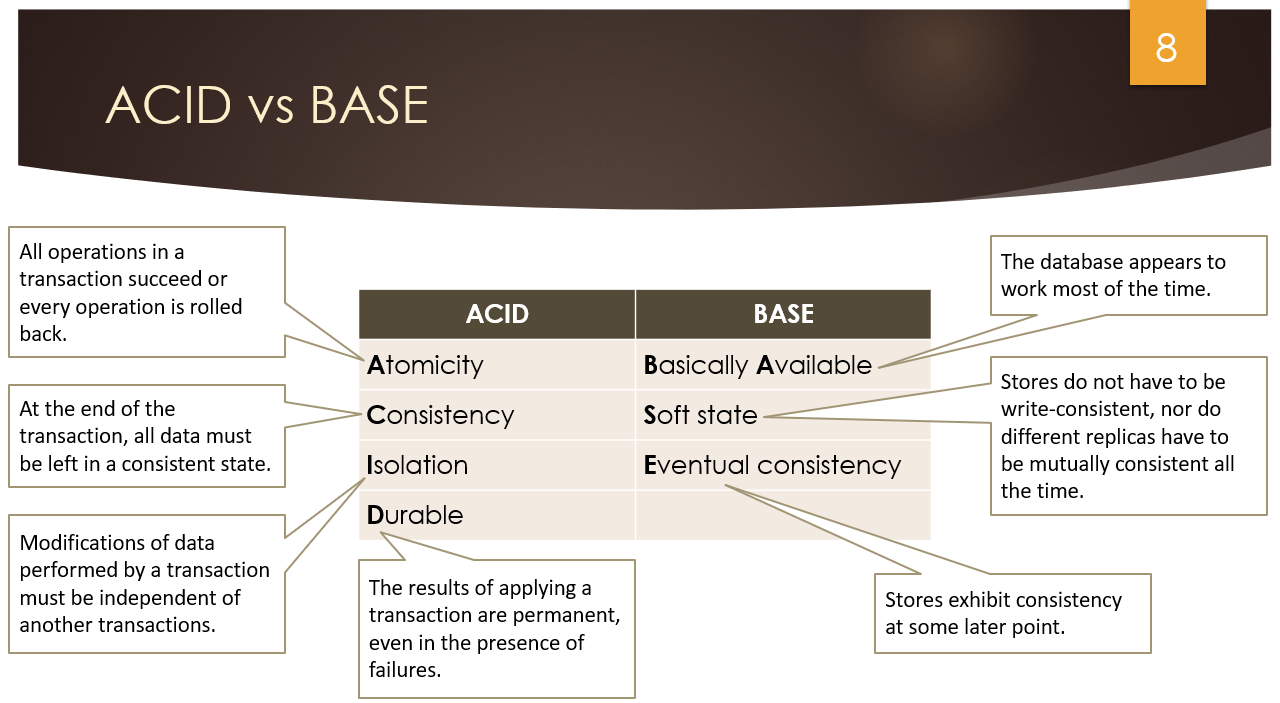
Explicitly: begin\_transaction and end\_transaction

Implicitly: first executable SQL statement

Once the first operation is executed, the transaction is **active**

If transaction completed successfully, it can be **committed**. If not, it needs to be **rolled back**

##### ACID vs BASE



#### Types of Failure & Recovery

##### Transaction failure

Transaction failure results from an error in the logic that drives the transaction’s operations and/or in the application logic

##### System failure

System failure occurs if the operating system or the database system crashes

##### Media failure

Media failure occurs if the secondary storage is damaged or inaccessible

##### Recovery

Recovery: activity of ensuring that, whichever of the problems occurred, the database is returned to a consistent state without any data loss afterwards

#### Log File

* A unique log **sequence number** and a unique **transaction** **identifier**
* A marking to denote the **start of a transaction**, along with the transaction’s start time and indication whether the transaction is **read only or read/write**
* **Identifiers of the database records** involved in the transaction, as well as the operation(s) they were subjected to
* **Before images** of all records that participated in the transaction
* **After images** of all records that were changed by the transaction
* The **current state** of the transaction (active, committed or aborted)
* Logfile may also contain **checkpoints**
* **Write ahead log** strategy: all updates are registered on the logfile before written to disk

#### Concurrency Control

Process of **managing simultaneous operations** on the database without having them interfere with one another

##### Purpose of locking

Purpose of locking is to ensure that, in situations where different concurrent transactions attempt to access the same database object, access is only granted in such a way that no conflicts can occur

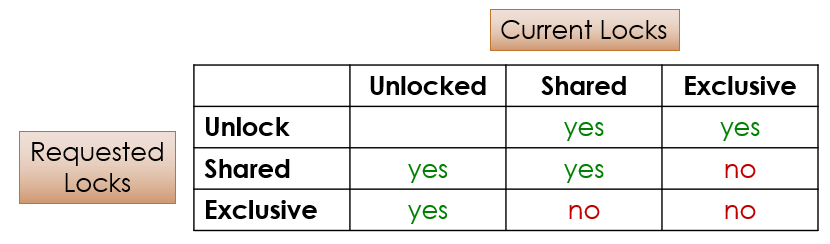
##### lock

A lock is a variable that is associated with a database object, where the variable’s value constrains the types of operations that are allowed to be executed on the object at that time

##### Lock Manager

**Lock manager** is responsible for granting locks (**locking**) and releasing locks (**unlocking**) by applying a locking protocol

Lock manager implements locking protocol which is a set of rules to determine what locks can be granted in what situation based on a compatibility matrix.



##### Exclusive and Shared Lock

An **exclusive lock** (x-lock or write lock) means that a single transaction acquires the sole privilege to interact with that specific database object at that time. No other transactions can read or write it.

A **shared lock** (s-lock or read lock) guarantees that no other transactions will update that same object for as long as the lock is held. Other transactions may hold a shared lock on that same object as well, however they are only allowed to read it.

##### Lock-based Concurrency control

Strict Two-phase Locking (Strict 2PL) Protocol:

* Each transaction must obtain the **appropriate lock** before accessing an object
* All locks held by a transaction are **released** when the transaction is completed
* **Growing phase** where locks are acquired on resources
* **Shrinking phase** where locks are released

### Advantages of Database Systems and Database Management

* Data Independence
* Managing Structured, Semi-Structured and Unstructured Data
* Managing Data Redundancy
* Specifying Integrity Rules
* Concurrency Control
* Backup and Recovery Facilities
* Data Security
* Performance Utilities

#### Data Independence

**Data independence** implies that changes in data definitions have minimal to no impact on the applications

**Physical data independence** implies that neither the applications, nor the views or logical data model must be changed when changes are made to the data storage specifications in the internal data model

**Logical data independence** implies that software applications are minimally affected by changes in the conceptual or logical data model

#### Managing Structured, Semi-Structured and Unstructured Data

##### Structured data

can be described according to a formal logical data model

ability to express integrity rules and enforce correctness of data

also facilitates searching, processing and analyzing the data

E.g., number, name, address and email of a student

##### Unstructured data

no finer grained components in a file or series of characters that can be interpreted in a meaningful way by a DBMS or application

E.g., document with biographies of famous NY citizens

##### Semi-structured data

XML data

#### Managing Data Redundancy

**Duplication** of data can be desired in distributed environments to improve data retrieval performance

DBMS is now responsible for the management of the redundancy by providing synchronization facilities to safeguard **data consistency**

Compared to the file approach, the DBMS guarantees **correctness of the data** without user intervention

#### Specifying Integrity Rules

**Syntactical rules** specify how the data should be represented and stored

customerID is an integer; birthdate should be stored as month, day and year

**Semantical rules** focus on the semantical correctness or meaning of the data

customerID is unique; account balance should be > 0; customer cannot be deleted if he/she has pending invoices

**Integrity rules** are specified as part of the conceptual\logical data model and stored in the catalog

directly enforced by the DBMS instead of applications

#### Concurrency Control

DBMS has built in facilities to support **concurrent or parallel execution** of database programs

Key concept is a **database transaction**

sequence of read/write operations considered to be an atomic unit in the sense that either all operations are executed or none at all

**Read/write operations** can be executed at the same time by the DBMS

DBMS should avoid **inconsistencies**!

##### DBMS must support ACID

* **Atomicity** requires that a transaction should either be executed in its entirety or not all
* **Consistency** assures that a transaction brings the database from one consistent state to another
* **Isolation** ensures that the effect of concurrent transactions should be the same as if they would have been executed in isolation
* **Durability** ensures that the database changes made by a transaction declared successful can be made permanent under all circumstances

#### Backup and Recovery Facilities

Backup and recovery facilities can be used to deal with the effect of **loss of data** due to hardware or network errors, or bugs in system or application software

Backup facilities (e.g., log files) can either perform a **full or incremental backup**

Recovery facilities allow to restore the data to a **previous state** after loss or damage occurred

#### Data Security

Data security can be enforced by the DBMS

Some users have **read** **access**, whilst others have **write** **access** to the data (role-based functionality)

Data access can be managed via logins and passwords assigned to **users** or **user** **accounts**

Each account has its own **authorization** **rules** that can be stored in the **catalog**

#### Performance Utilities

##### Three KPIs of a DBMS are

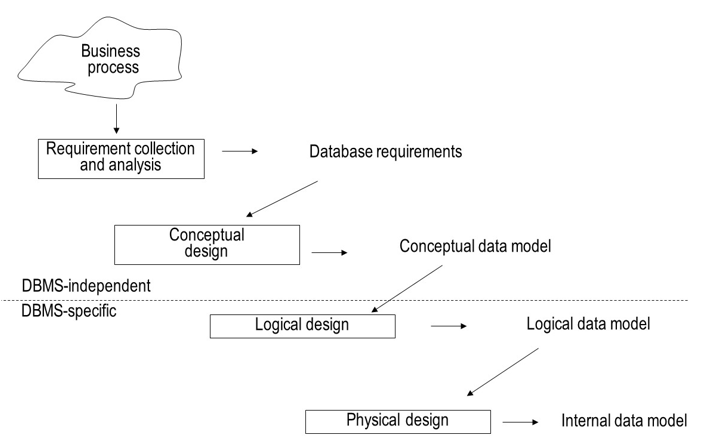
* **response** **time** denoting the time elapsed between issuing a database request and the successful termination thereof
* **throughput** **rate** representing the number of transactions a DBMS can process per unit of time
* **space** **utilization** referring to the space utilized by the DBMS to store both raw data and metadata

DBMSs come with various types of **utilities** aimed at improving these KPIs such as:

* Utilities to distribute and optimize **data** **storage**
* Utilities to **tune** **indexes** for faster query execution
* Utilities to tune queries to improve **application** **performance**
* Utilities to **optimize** **buffer** management

## Conceptual Data Modeling using the ER model Diagram

### Phases of Database Design



### Entity Relationship (ER) model

Entity Types

Attribute Types

Relationship Types

Weak Entity Types

Examples of the ER model

Limitations of the ER model

#### Entity Types

Entity type represents a business concept with an unambiguous meaning to a particular set of users

Examples: supplier, student, product or employee

Entity is one particular occurrence or instance of an entity type

#### Attribute Types

Attribute type represents a property of an entity type.

Example: name and address are attribute types of the entity type supplier

Attribute is an instance of an attribute type

* Domains
* Key Attribute Types
* Simple versus Composite Attribute Types
* Single-Valued versus Multi-Valued Attribute Types
* Derived Attribute Type

##### Domains

A **domain** specifies the set of values that may be assigned to an attribute for each individual entity

Example: gender: male and female

A domain can also contain **null values**

null value: value is not known, not applicable or not relevant

Domains are **not** **displayed** in an ER model

##### Key Attribute Types

A **key attribute type** is an attribute type whose values are distinct for each individual entity

Examples: supplier number, product number, social security number

A key attribute type can also be a **combination** of attribute types

Example: combination of flight number and departure date

##### Simple versus Composite Attribute Types

A **simple or atomic attribute type** cannot be further divided into parts (e.g., supplier number, supplier status)

A **composite attribute type** is an attribute type that can be decomposed into other meaningful attribute types (e.g., address, name)

##### Single-Valued versus Multi-Valued Attribute Types

A **single-valued attribute type** has only one value for a particular entity (e.g., product number, product name)

A **multi-valued attribute type** is an attribute type that can have multiple values (e.g., email address)

##### Derived Attribute Type

A **derived attribute type** is an attribute type which can be derived from another attribute type (e.g., age)

#### Relationship Types

##### Definition

A **relationship** represents an association between two or more entities

A **relationship** **type** then defines a set of relationships among instances of one, two or more entity types

##### Degree and Roles

The **degree of a relationship type** corresponds to the number of entity types participating in the relationship type:

**Unary**: degree 1, **binary**: degree 2, **ternary**: degree 3

The **roles** of a relationship type indicate the various directions that can be used to interpret it. For example, a relationship "marriage" and its two roles "husband" and "wife". It has also become prevalent to name roles with phrases such as “is the owner of” and “is owned by”.

##### Cardinalities

Every relationship type can be characterized in terms of its cardinalities, which **specify the minimum or maximum number of relationship instances** that an individual entity can participate in

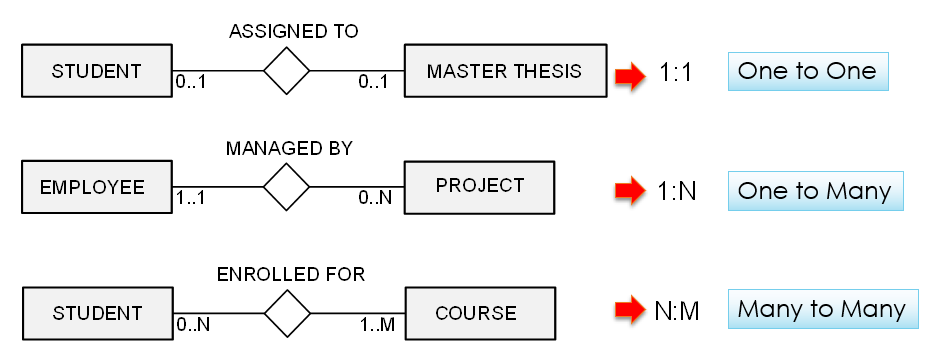
**Minimum** cardinality can be 0 or 1

If 0: **partial** participation

If 1: **total** participation or existence dependency

**Maximum** cardinality can be 1 or N

3 options for **binary** relationship types: 1:1, 1:N, and M:N



###### Participation Constraints

Participation Constraints represents how an entity is involved in the relation.

If all the entity values are participating in any relation, then it is called **total** **participation**.

If only few values of an entity is part of relation, then it is a **partial** **participation**.

##### Relationship Attribute Types

**Relationship** **type** can also have **attribute** **types**

These attribute types can be migrated to one of the participating entity types in case of a 1:1 or 1:N relationship type

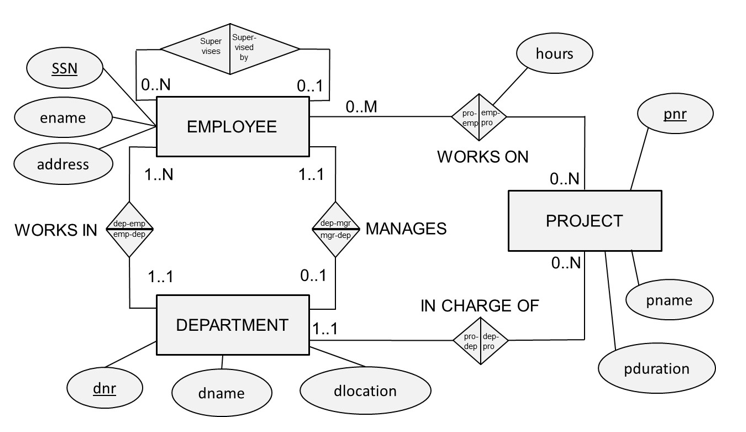
#### Weak Entity Types

A **strong** **entity** **type** is an entity type that has a key attribute type

A **weak** **entity** **type** is an entity type that does not have a key attribute type of its own

In a relational database, a weak entity is an entity that cannot be uniquely identified by its attributes alone; therefore, it must use a foreign key in conjunction with its attributes to create a primary key. The foreign key is typically a primary key of an entity it is related to.

#### Examples of the ER model



#### Design the ER model

* Identify the **entity** **types**
* Identify the **relationship** **types** and assert their degree
* Assert the **cardinality** **ratios** and participation constraints (total versus partial participation)
* Identify the **attribute** **types** and assert whether they are simple or composite, single or multiple valued, derived or not
* Link each **attribute** type to an **entity** type or a **relationship** type
* Denote the **key attribute type(s)** of each entity type
* Identify the **weak entity type(s)** and their partial keys

#### Limitations of the ER model

**Domains** are not included in the ER model

Examples: hours should be positive; prodtype must be red, white or sparkling, supstatus is an integer between 0 and 100

**Functions** are not included in the ER model

Examples: calculate average number of projects an employee works on; determine which supplier charges the maximum price for a product

ER model **cannot model temporal constraints**

Example: a project needs to be assigned to a department after one month

ER model **cannot guarantee the consistency** across multiple relationship types

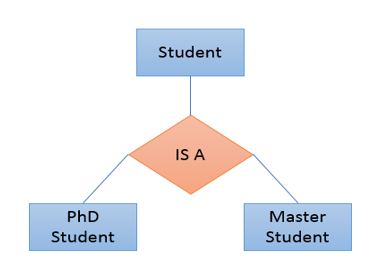
Example: suppliers can only be assigned to purchase orders for products they can supply

### Enhanced Entity Relationship (EER) Model

#### Specialization

Specialization (**Top Down Approach**) is the process of identifying subsets of an entity set (the superclass) that share some distinguishing characteristics.

Typically, **the** **superclass** is defined first, **the** **subclasses** are defined next, and subclass-specific attributes and relationship sets are then added.



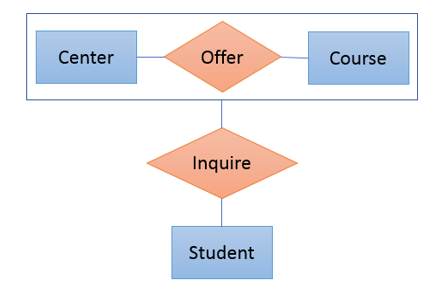
#### Generalization

Generalization (**Bottom Up Approach**) consists of identifying some common characteristics of a collection of entity sets and creating a new entity set that contains entities possessing these common characteristics.



#### Aggregation

**Aggregation** is a process when relation between two entities is treated as a single entity. It allows us to indicate that a relationship set participates in another relationship set.



## Database Normalization

### Normalization

A process of organizing the data in database to **avoid** **data redundancy** and **insertion**, **update** and **deletion** anomaly

A process of organizing and reorganizing data into tables (relations) in such a way that the results of using database are always **clear and as intended.**

It ensures optimum structure with atomic data.

### Super Key & Candidate Key

A **super** **key** is a set of attributes which can uniquely identify a tuple.

The minimal set of attribute which can uniquely identify a tuple is known as **candidate** **key** (e.g., NetID).

**A candidate key is a super key** from which you cannot remove any fields. A candidate key is a super key but **not vice versa.**

An attribute that is a part of one of the candidate keys is known as **prime attribute.**

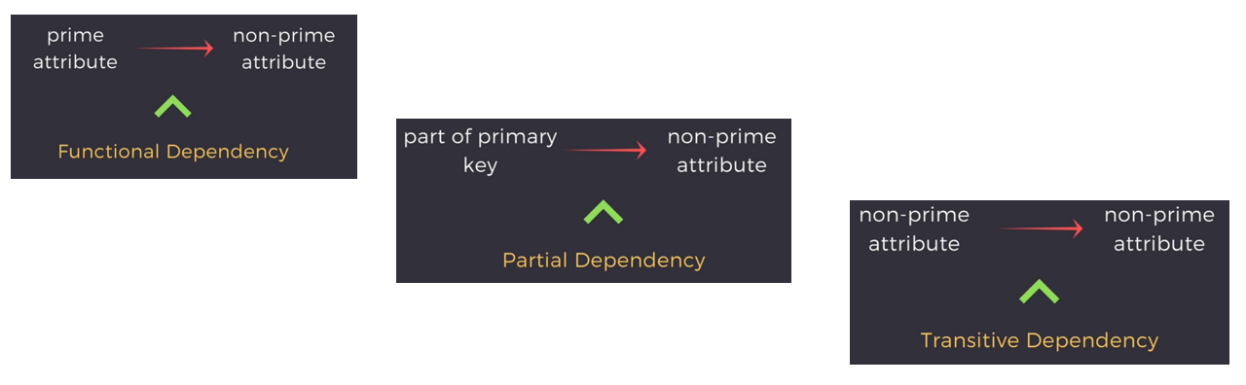
An attribute is not part of any candidate key known as **non-prime attribute.**

There can be more than one candidate key in a table (e.g., NetID or Student\_No).

**Primary** **key** is a chosen candidate key for a table

**Alternate** **keys** are other candidate keys

### Attribute Dependency



### 1NF – First Normal Form

**Each attribute of a table must have atomic values**

No repeated tuples (records or rows)

No repeated attributes (columns or fields)

No multi-value data (attribute value)

No repeated groups at the intersection of each row and column.

### 2NF – Second Normal Form

The table is in 1NF.

**No** non-prime attribute is **dependent** on only **a portion of** the primary key.

2NF only a problem with compound (composite) keys.

### 3NF – Third Normal Form (BCNF)

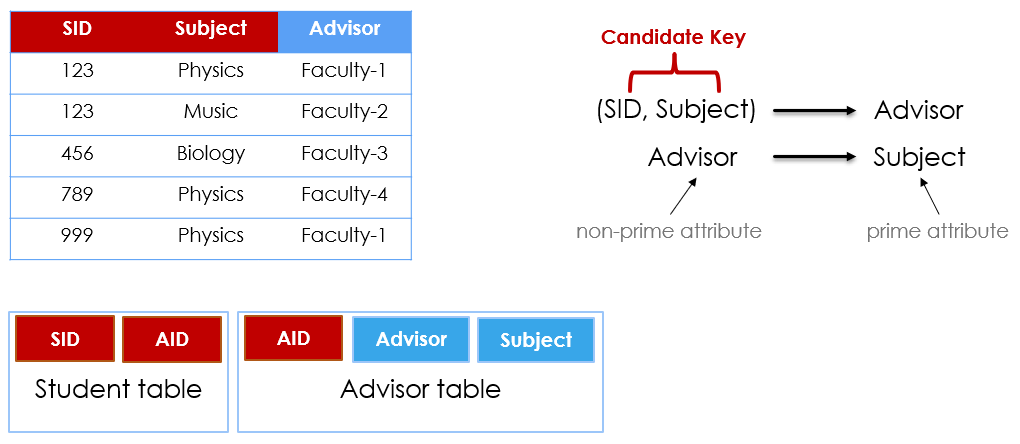
The Table is in 2NF.

All non-prime attributes are **not dependent** on any other non-prime attributes.

The only determinants are candidate keys (Boyce-Codd Normal Form)

#### BCNF

Example:



### 4NF – Forth Normal Form

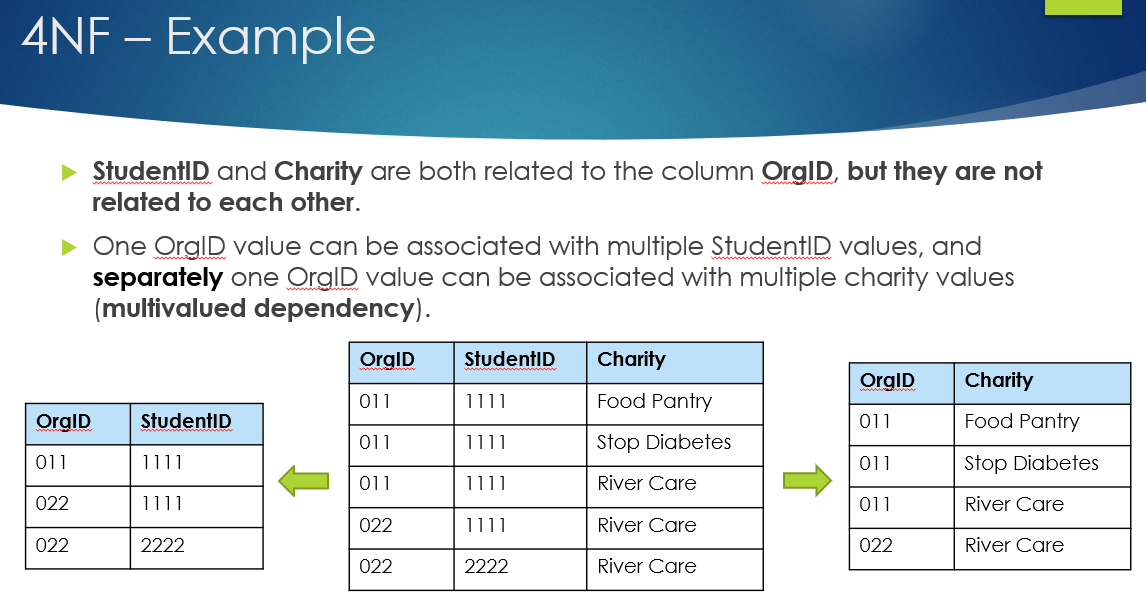
The Table is in 3NF or BCNF.

There are **no multivalued** dependencies.

Multivalued dependencies occur when the relation is trying to represent more than one many-to-many relations.

students-teachers-courses

suppliers-products-locations



### 5NF - Fifth Normal Form

The Table is in 4NF.

If we decompose a table further to eliminate redundancy an anomaly, and when we rejoin the decomposed tables using the candidate keys, we should **not lose or gain tuples.**

### Summary (Difference)

|  |  |
| --- | --- |
| **1NF** | No repeating groups in tuples (rows), attributes (columns) or values (data) |
| **2NF** | 1NF and no partial dependencies |
| **3NF** | 2NF and no transitive dependencies |
| **3.5NF** | 3NF and For any dependency A > B, A should be a super key. |
| **4NF** | 3NF or 3.5NF and no multivalued dependencies |
| **5NF** | 4NF and rejoining the table should not lose or gain tuples |

## Advanced SQL

### SQL Control-Flow Statements

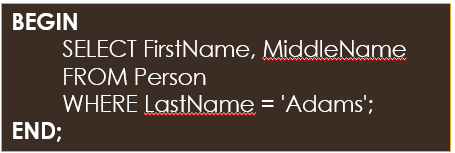
#### BEGIN TRANSACTION

Marks the starting point of **an explicit, local transaction.** Explicit transactions start with the **BEGIN** **TRANSACTION** statement and end with the **COMMIT** or **ROLLBACK** statement.

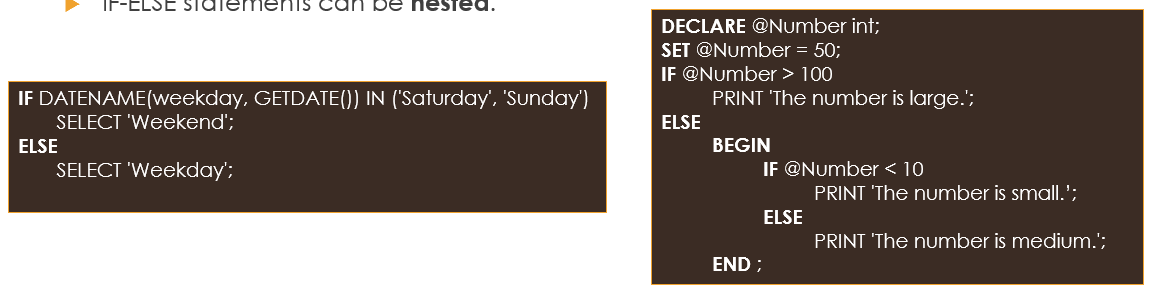


#### BEGIN-END

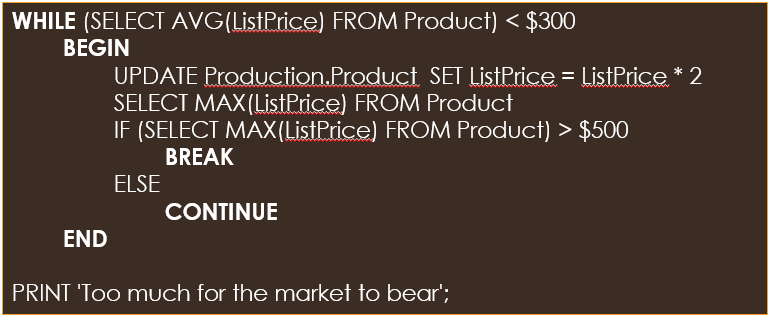
Encloses a series of SQL statements so that **a group of SQL statements** can be executed.



#### IF-ELSE



#### WHILE-BREAK-CONTINUE



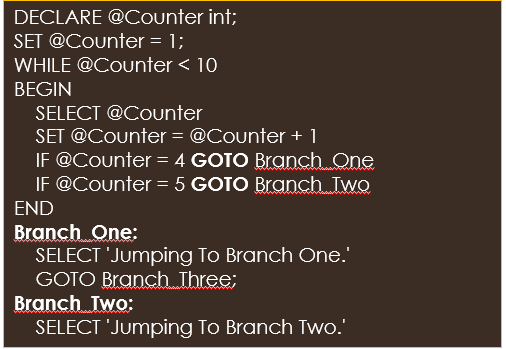
#### GOTO

**Alters the flow of execution** to a label.

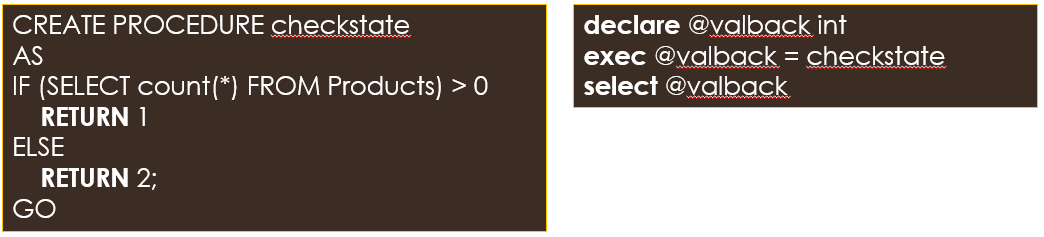
The SQL statement or statements that follow GOTO are **skipped** and processing continues at the label.

GOTO statements and labels can be **used anywhere** within a procedure, batch, or statement block.

GOTO statements can be **nested**.

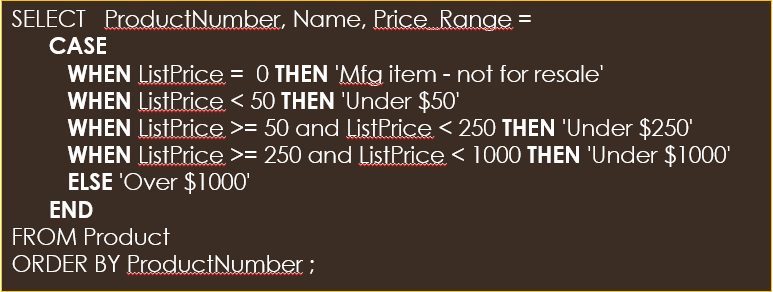


#### RETURN



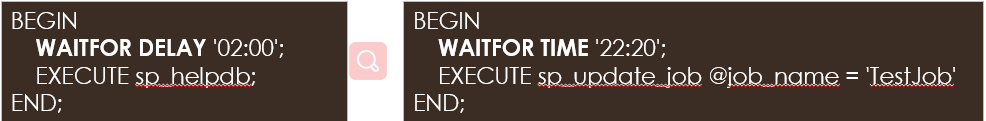
#### CASE

**CASE-[WHEN-THEN]-ELSE-END** provides a structured method of evaluating **a list of options** and then **returning a single value**.



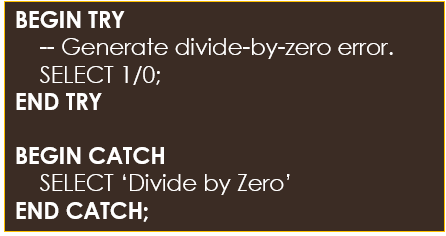
#### WAITFOR

**Blocks the execution** of a batch, stored procedure, or transaction until a specified time or time interval is reached.



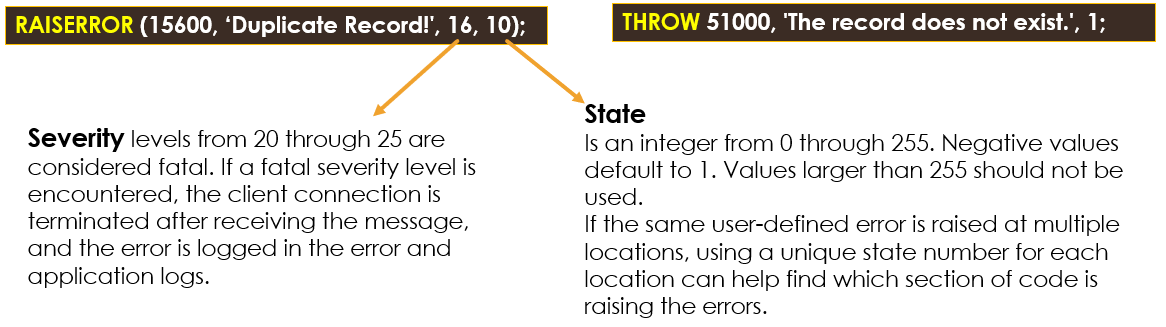
#### TRY … CATCH

A group of SQL statements can be enclosed in a **TRY** **block**. If an error occurs in the TRY block, control is passed to another group of statements that is enclosed in a **CATCH** **block**.



#### RAISERROR & THROW

**Raises an exception** and transfers execution to a CATCH block of a TRY…CATCH construct.



##### Differences Between RAISERROR and THROW

|  |  |
| --- | --- |
| **RAISERROR statement** | **THROW statement** |
| If a msg\_id is passed to RAISERROR, the ID must be defined in sys.messages. | The error\_number parameter does not have to be defined in sys.messages. |
| The msg\_str parameter can contain printf formatting styles. | The message parameter does not accept printf style formatting. |
| The severity parameter specifies the severity of the exception. | There is no severity parameter. The exception severity is always set to 16. |

### SQL Functions

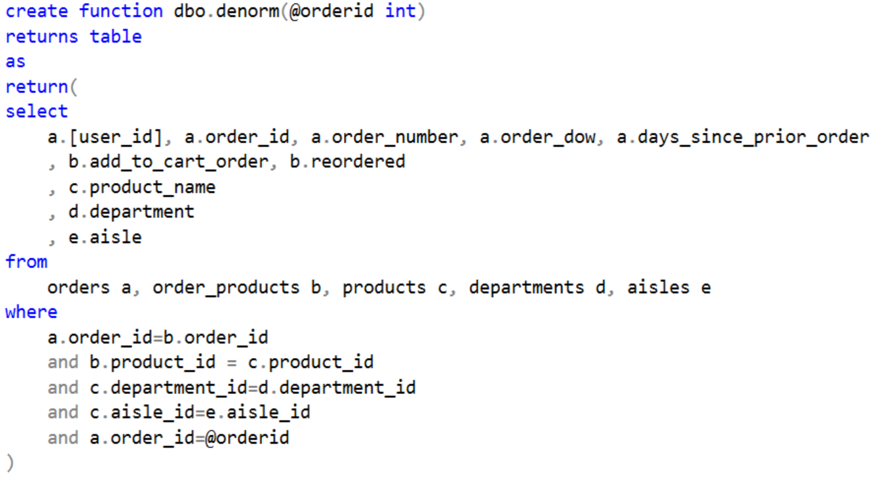
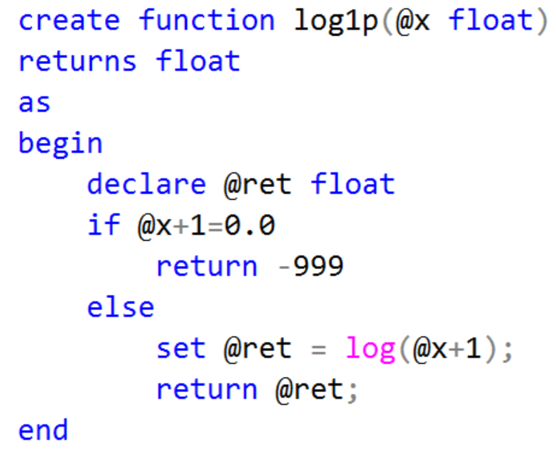
User can create **functions** in SQL Server for saving the SQL statements permanently in the system.

The functions are calls as **User Defined Functions (UDF).**

A **UDF** is the database object that contains a set of SQL statements.

The function accepts **input** as parameters, performs **actions** and the result set is returned as action. The **return** **value** can be a result set or a single value.

**Scalar** **functions** returns one value and **Table-valued functions** returns a table of results.



### Stored Procedures

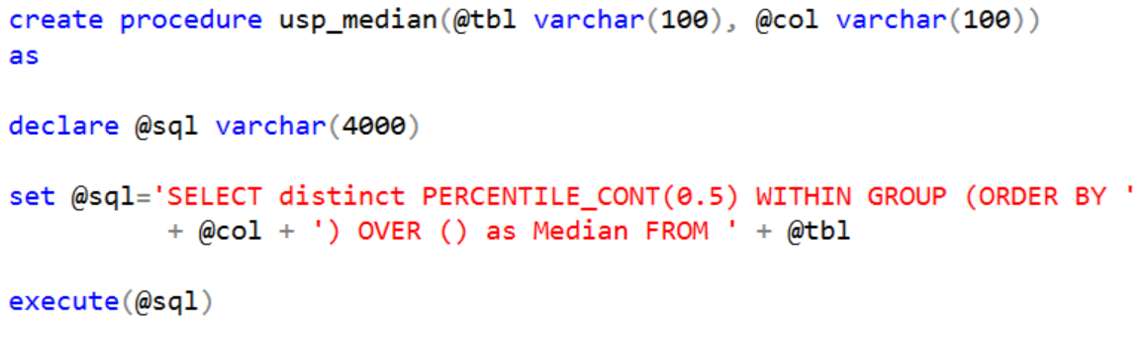
If you find yourself using the same query over and over again, it would make sense to put it into a stored procedure.

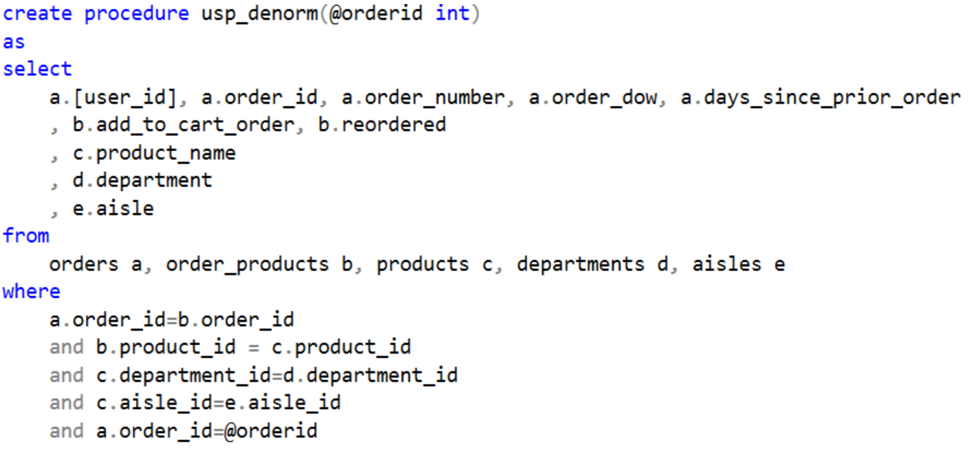
A **stored procedure** is an already written SQL statement that is saved in the database.

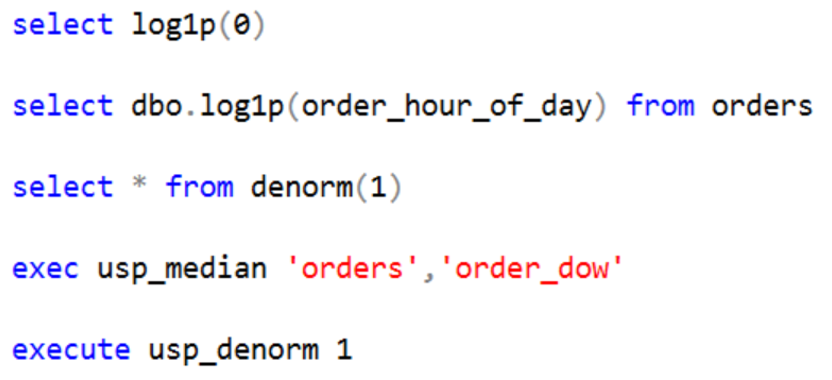
When you put this SQL statement in a stored procedure, you can then run the stored procedure from the database’s command environment using the **exec command.**

Like a function, a stored procedure can be used to perform a **calculation**.

Like a view, a stored procedure can be used to create and store a **query**.







### SQL Triggers

A **SQL trigger** is a special type of stored procedures that is **automatically** executed when an event occurs in a specific database server **(Event-Condition-Action)**

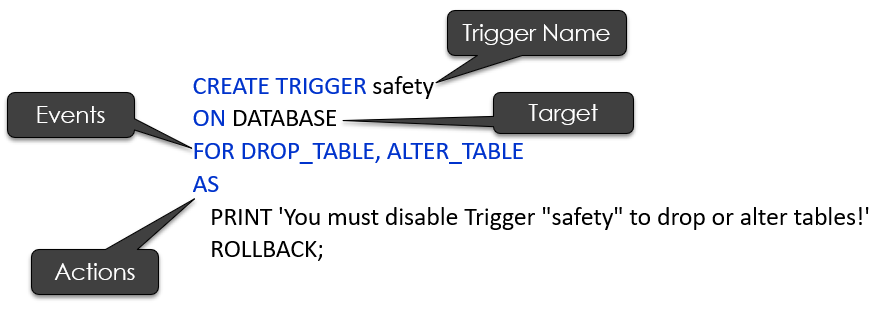
Triggers and Stored Procedures are **SQL** **statements**. However, a stored procedure has to be **executed by a user,** while a trigger is **executed by the system** as the result of an event

**Events** that cause triggers to be activated include **insert, update, delete, create, alter** and **drop**.

One **drawback** to using triggers instead of stored procedures is that they cannot accept parameters.

#### DDL Triggers

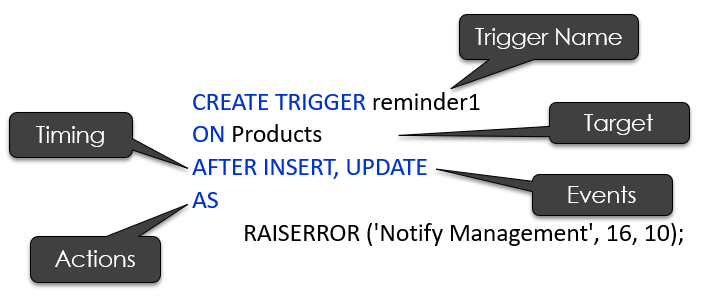
A DDL trigger will be fired in response to different Data Definition Language **(DDL) events**, such as **CREATE, ALTER, DROP T-SQL statements**



#### DML Triggers

A DML trigger will be fired in response to different Data Manipulation Language (**DML) events,** such as **INSERT, UPDATE or DELETE** action

The DML triggers can be used to maintain **data integrity** and **enforce** **business** **rules** by performing **auditing** **processes** and other post DML actions.



#### SQL Triggers Syntax

**Trigger** **name** - unique within one database schema

**Timing** - depends on the order of controlled events (before or after or instead of)

**Triggering** **event** - event which fires the trigger (E)

**Filtering** **condition** - checked when the triggering event occurs (C)

**Target** - table (or view) against which the trigger is fired; they should be both created within the same schema

**Trigger** **action** - SQL statements, executed when the trigger fires; surrounded by Begin ... End (A)

#### Types of SQL Triggers

##### How many times

**How many times** should the trigger body execute when the triggering event takes place?

**Per** **statement**: the trigger body executes once for the triggering event. This is the default.

**For** **each** **row**: the trigger body executes once for each row affected by the triggering event.

##### When

**When** the trigger can be fired

Relative to the execution of an SQL DML statement (**before** or **after** or **instead of**)

**Exactly in a situation** depending on specific system resources (e.g. signal from the system clock, expiring timer, exhausting memory)

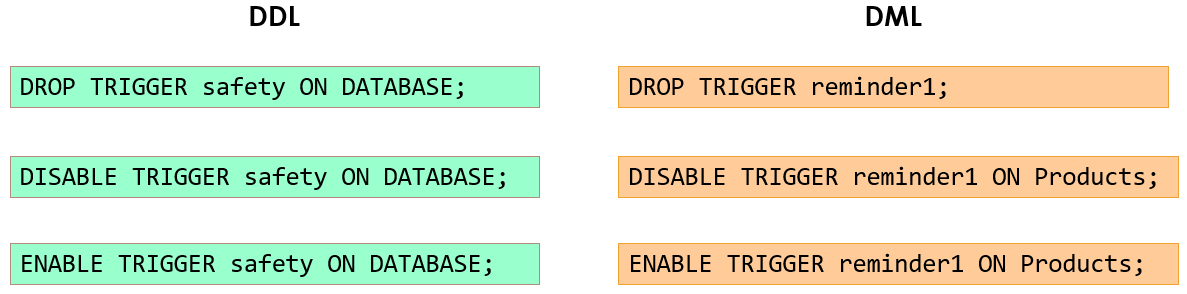
#### Trigger Execution Order

* Execute all **BEFORE** **STATEMENT** triggers
* **Disable** temporarily all **integrity** **constraints** recorded against the table
* Loop for each row in the table
  + Execute all **BEFORE** **ROW** triggers
  + Execute the SQL statement **against the row** and **perform integrity constraint checking** of the data
  + Execute all **AFTER ROW** triggers
* Complete deferred **integrity constraint** checking against the table
* Execute all **AFTER** **STATEMENT** triggers

##### Timing - INSTEAD OF

An INSTEAD OF trigger is a trigger that allows you to **skip** an INSERT, DELETE, or UPDATE statement to a table or a view and **execute other statements defined in the trigger** instead. The actual insert, delete, or update operation **does not occur** at all.

#### DROP, DISABLE and ENABLE Triggers



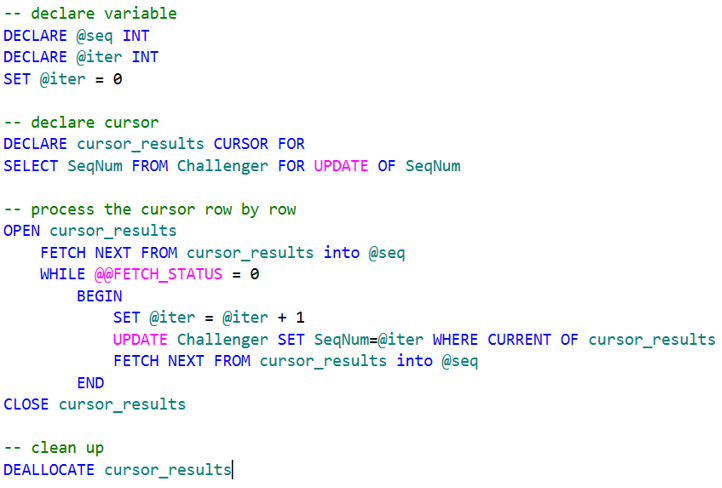
### SQL Cursors

Operations in a relational database act on **a complete set of rows**.

This complete set of rows returned by the statement is known as **the result set**.

Applications, especially interactive online applications, cannot always work effectively with the entire result set as a unit. These applications need **a mechanism to work with one row or a small block of rows** at a time.

**Cursors** are an extension to result sets that provide that mechanism.



## Parallel and Distributed Databases

### why

Data is too large, too big and too many (**Big Data**)

In a centralized database:

Data is located in one place (one server) and all DBMS functionalities are done by that server

Enforcing ACID properties of transactions, taking care of concurrency control, recovery mechanisms and answering queries

Applications are by nature distributed

There is a need for parallel and distributed processing

Divide a big problem into many smaller ones to be solved in parallel

Increase bandwidth (e.g., Faster response time for queries)

**Sharing data** is the key of a distributed DB (cooperative approach)

Since data is distributed, users that share that data can have it placed at the site they work on, with local control (local autonomy)

Distributed and parallel databases improve **reliability** and **availability**

They have **replicated components** and thus eliminate single points of failure

The distribution of data and the parallel/distributed processing is not visible to the users (transparency)

A parallel database aims principally **linear speedup and linear scaleup**

### Parallel Database System

Seeks to improve performance through **parallelization** of various operations, such as: loading data, building indices, valuating queries and more

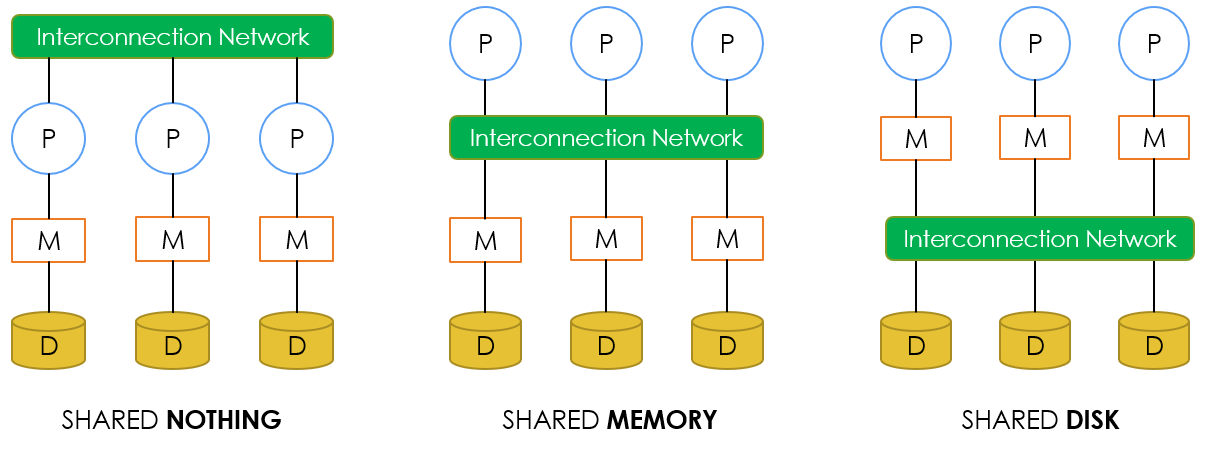
Although data may be stored in a distributed fashion in such a system, **the distribution is governed solely by performance considerations.**

A parallel database aims principally linear speedup and linear scaleup

**Linear speedup** means the time taken for operations decreases linearly in proportion to the increase in the numbers of CPUs and disks

**Linear scaleup** means performance is sustained if the number of CPUs and disks are increased in proportion to the amount of data

#### Architectures for Parallel Databases



##### SHARED NOTHING

In a shared-nothing parallel system, each CPU has local main memory and disk space, but no two CPUs can access the same storage area.

All communication between CPUs is through a **network connection.**

##### SHARED MEMORY

In a shared-memory parallel system, multiple CPUs are attached to an interconnection network and can access a **common region of main memory.**

Every processor has its own disk

Single memory address-space for all processors

Reading or writing to far memory can be slightly more expensive

Every processor can have its own local memory and cache as well

##### SHARED DISK

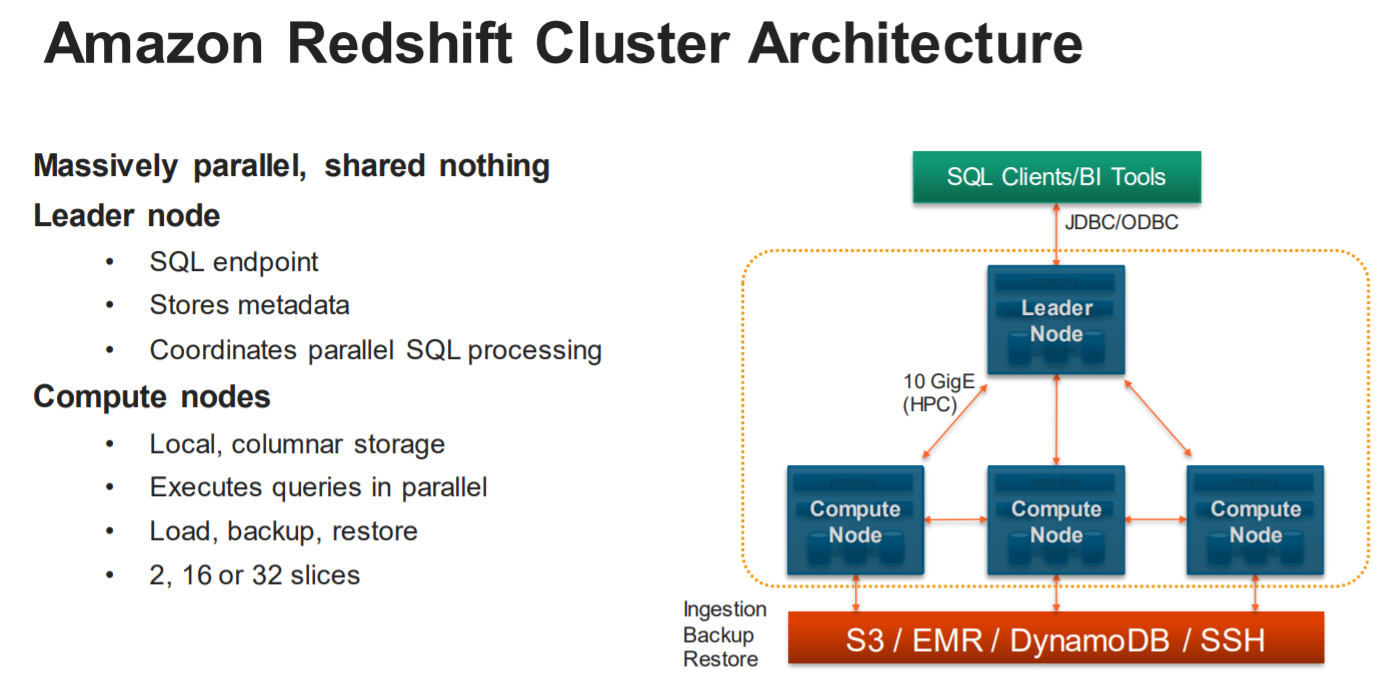
In a shared-disk system, each CPU has a private memory and **direct access to all disks** through an interconnection network.

Every processor has its own memory (not accessible by others)

All machines can access all disks in the system

Number of disks does not necessarily match the number of processors

#### Amazon Redshift



#### Types of parallelism in database systems

**Interquery parallelism**: Parallelism between queries. Multiple queries generated by concurrent transactions can be executed in parallel

**Intraquery parallelism** : Parallelism within a query

**Interoperation parallelism**: A query may consist of multiple, independent operations that can be executed in parallel. There are two forms of interoperation parallelism:

**Independent**: Operations in a query that do not depend on one another are executed in parallel. It does not provide a high degree of parallelism

**Pipeline** : The output of operation A is consumed by another operation B, before A has produced the entire output Many machines, each doing one step in a multi-step process

**Intraoperation parallelism**: An operator may be executed as many small, independent sub-operations. The relational model is ideal for implementing intraoperation parallelism

### Distributed Databases

A **collection** of multiple, logically interrelated databases distributed over a computer network with the following features:

Improved **reliability** and **availability** through distributed transactions

Improved **performance**

Allowing **data sharing** while maintaining some measure of local control

Easier and more economical **system expansion**

Distributed Database Management System (**D-DBMS**) is a software system that permits the management of the DDB and provides an access mechanism that **makes this distribution transparent to the users**

#### Distributed Databases – Functionalities

* To access remote sites and transmit queries and data among the various sites via a communication network
* To keep track of the data distribution and replication in the DDBMS catalog
* To devise execution strategies for queries and transactions that access data from more than one site
* To decide on which copy of a replicated data item to access
* To maintain the consistency of copies of a replicated data item
* To maintain the global conceptual schema of the distributed database
* To recover from individual site crashes and from new types of failures such as failure of a communication link

#### Distributed Databases – Date’s 12 Rules

Rule 0: to the user, a distributed system should look exactly like a non-distributed system

Autonomy, Independence and Transparency

Local Autonomy

all the data in the distributed network is owned and managed locally

Independency

no reliance on central site

Transparency

to the user, a distributed system should look exactly like a non-distributed system

##### Rule 1: Local Autonomy

all the data in the distributed network is owned and managed locally

##### Rule 2: No reliance on central site

no reliance on central site

##### Rule 3: Continuous operation

##### Rule 4: Location Independence

##### Rule 5: Fragmentation Independence

##### Rule 6: Replication Independence

##### Rule 7: Distributed query processing

##### Rule 8: Distributed transaction management

##### Rules 9-11: Hardware, OS and Network Independence

##### Rule 12: DBMS independence

#### Recovery control: Two-phase commit protocol

**FIRST** **PHASE**: Each site must force-write all log entries for local resources used by the transaction out to its physical log. Assuming the force-write is successful, each site gives the OK.

**SECOND** **PHASE**: If the site where the transaction has been submitted receives all the OK form the other sites, it force-writes an entry to its own physical log. If anything has gone BAD, then general ROLLBACK otherwise the decision is COMMIT.

## NoSQL (Not Only SQL)

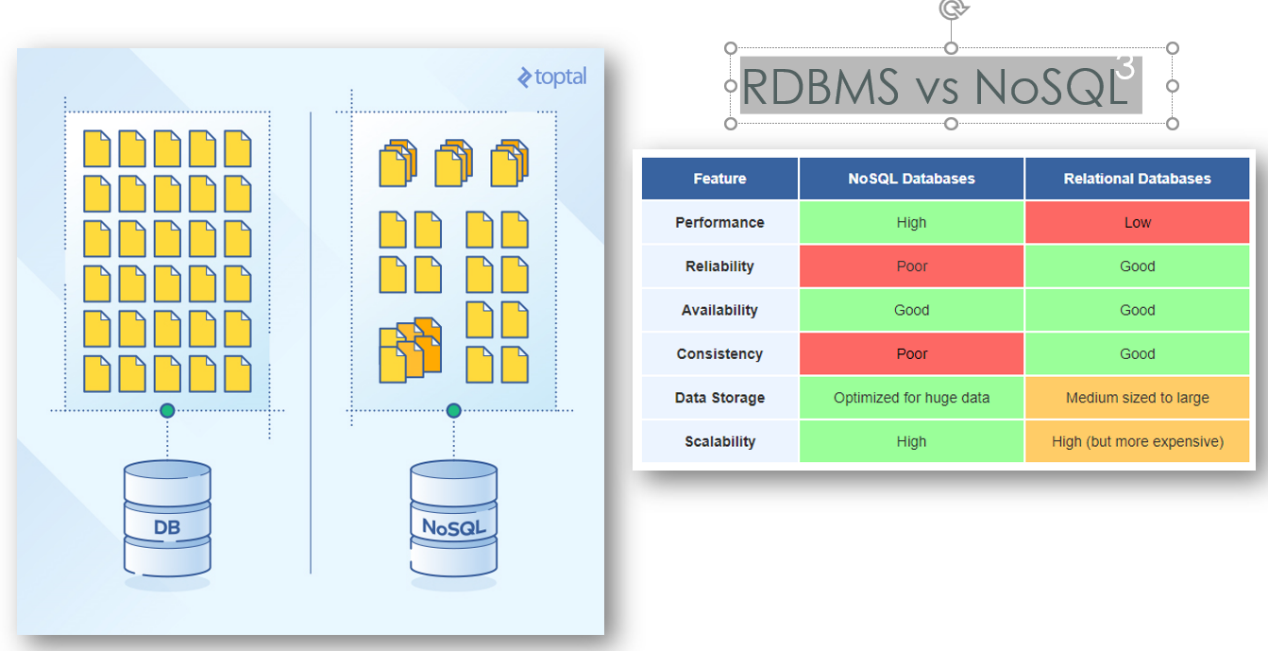
### What is NoSQL?

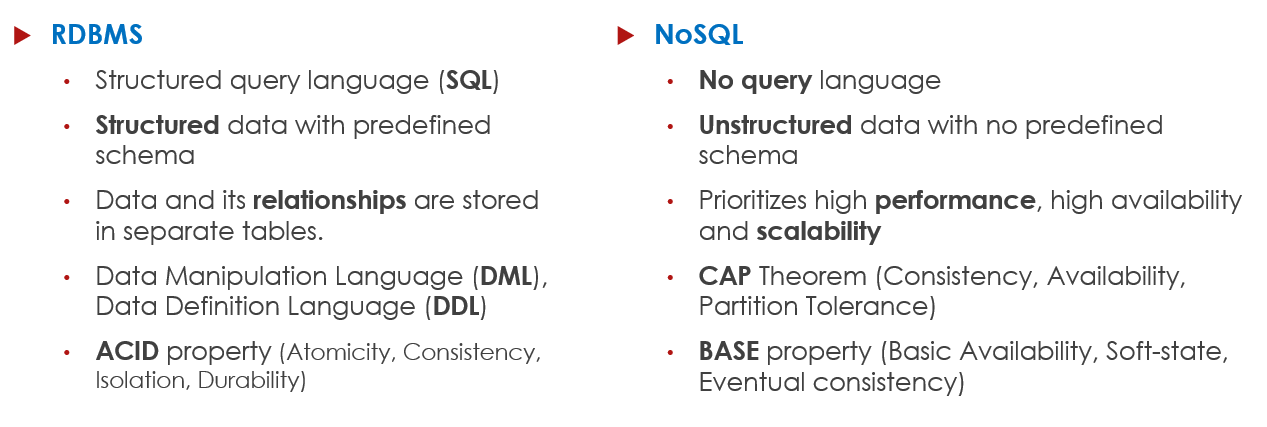
NoSQL is a **non-relational database management systems,** different from traditional relational database management systems in some significant ways.

It is designed for **distributed** **data** **stores** where very large scale of data storing needs (for example Google or Facebook which collects terabits of data every day for their users).

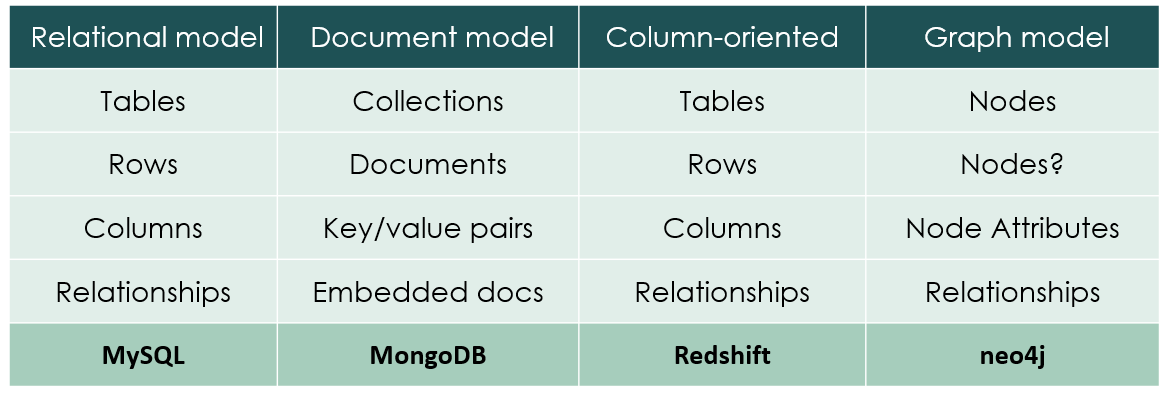
These type of data storing may **not require fixed schema, avoid join operations** and typically **scale horizontally.**

### RDBMS vs NoSQL





### Relational & NoSQL data models



### CAP Theorem

#### Consistency

**Consistency** - This means that the data in the database remains consistent after the execution of an operation. For example after an update operation all clients see the same data.

#### Availability

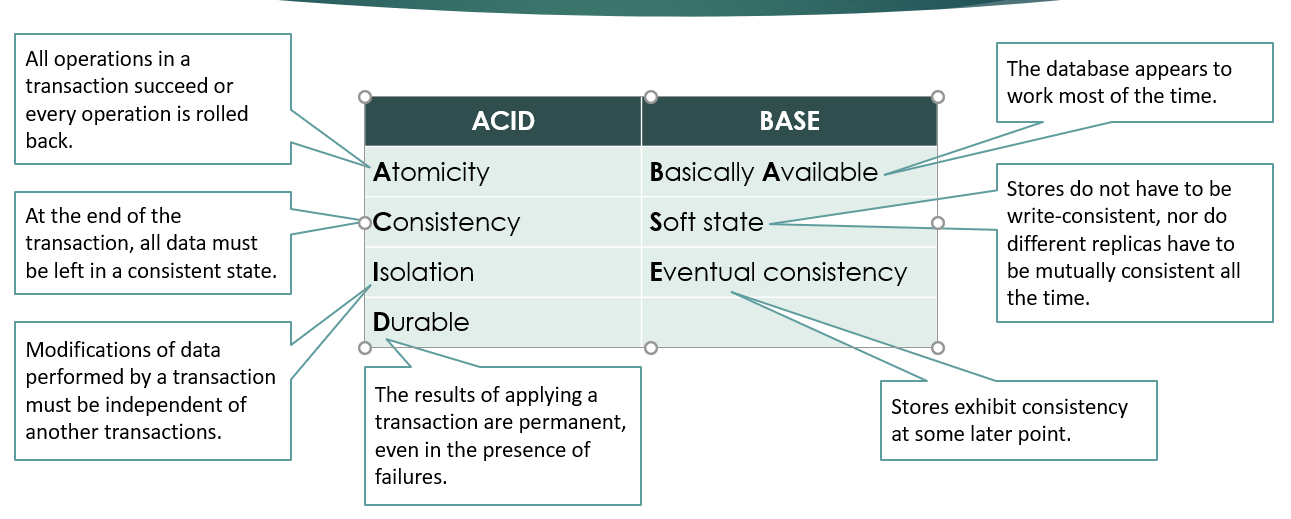
**Availability** - This means that the system is always on (service guarantee availability), no downtime.

#### Partition Tolerance

**Partition Tolerance** - This means that the system continues to function even the communication among the servers is unreliable, i.e. the servers may be partitioned into multiple groups that cannot communicate with one another.

### ACID vs BASE

|  |  |
| --- | --- |
| **ACID** | **BASE** |
| **A**tomicity | **B**asically **A**vailable |
| **C**onsistency | **S**oft state |
| **I**solation | **E**ventual consistency |
| **D**urable |  |



### NoSQL Categories

There are four general types (most common categories) of NoSQL databases. Each of these categories has its own specific attributes and limitations. There is not a single solutions which is better than all the others, however there are some databases that are better to solve specific problems.

#### Key-Value stores

Key-value stores are **most** **basic** types of NoSQL databases.

Designed to handle **huge amounts of data.**

Key value stores allow developer to **store schema-less data.**

In the key-value storage, database stores data as **hash table** where each **key is unique** and the value can be **String, JSON, BLOB/BSON,** etc.

Key-Value stores can be used as **collections, dictionaries, associative arrays**, etc.

Key-Value stores follow the **'Availability'** and **'Partition'** aspects of CAP theorem.

Key-Values stores would work well for **shopping cart contents**, or individual values like a landing page URI, or a default account number.

##### Typical Key-Value API

* get(key) -- Extract the value given a key
* put(key, value) -- Create or update the value given its key
* delete(key) -- Remove the key and its associated value
* execute(key, operation, parameters) -- Invoke an operation to the value (given its key) which is a special data structure (e.g. List, Set, Map .... etc.)

#### Document oriented

Data in this model is stored inside a collection of **documents**.

A document is **a key value collection** where the key allows access to its value.

Documents are typically **schemaless** and therefore are flexible and easy to change.

Documents are stored into **collections** in order to group different kinds of data.

Documents can contain many **different key-value pairs, or key-array pairs,** or even **nested documents.**

#### Column-oriented

Column-oriented databases primarily **work on columns** and every column is treated individually.

Values of a single column are **stored contiguously.**

Column stores data in column **specific files.**

All data within each column datafile have the **same type** which makes it ideal for **compression**.

Column stores can **improve the performance** of queries as it can access specific column data.

High performance on **aggregation queries** (e.g. COUNT, SUM, AVG, MIN, MAX).

Works on data warehouses (**DW**) and business intelligence (**BI**), customer relationship management (**CRM**), Library card catalogs, etc.

#### Graph

A graph database stores data in a **graph**.

It is capable of **elegantly representing** any kind of data in a highly accessible way.

A graph database is a collection of **nodes** (vertices) and **relationships** (edges).

Each **node represents an entity** (such as a student or business) and each **edge represents a connection or relationship** between two nodes.

Every node and edge are defined by a **unique identifier.**

Each node knows its **adjacent nodes.**

As the number of nodes increases, the cost of a **local step (or hop)** remains the same.

### Replication and Sharding

**Replication** copies the entire database across all nodes in the distributed system.

**Sharding** divides the data inside the database and partitions pieces of it to different nodes. Databases can be sharded horizontally (by rows) or vertically (by columns).

### MongoDB

NoSQL

Used for high-volume data storage

Stores data in flexible, JSON-like documents

#### MongoDB Features?

Each database contains collections which in turn contains documents

No predefined schema

Data model allows you to represent hierarchical relationships, to store arrays, and other more complex structures more easily

Scalable

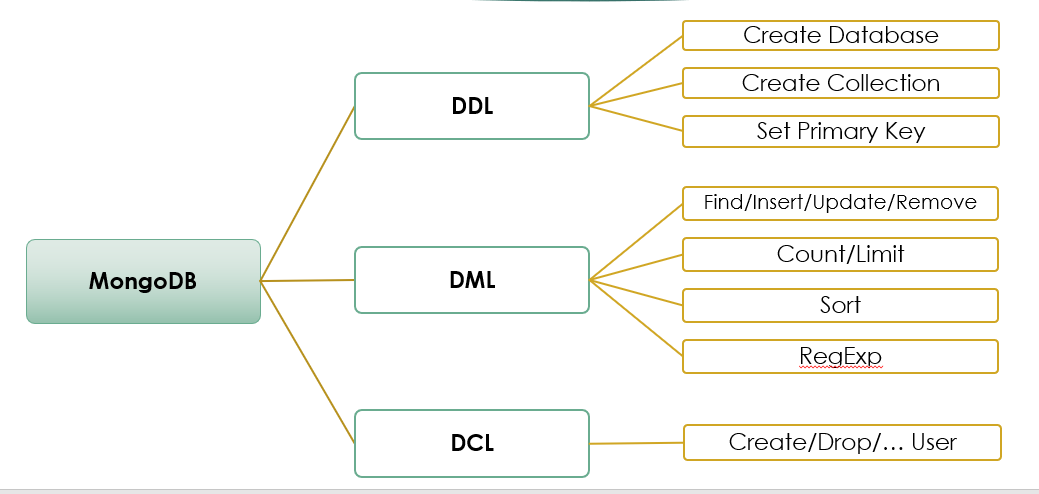
#### Architecture - Key Components



##### Document

JSON (JavaScript Object Notation)

#### MongoDB API



#### MongoDB – Management System

##### Database security

Enable user-based access control

Configure role-based access control

Use of encryption protocol such as TLS or SSL to provide communication security

Enable auditing

Run MongoDB server instance with a separate user id

##### Backup procedures

**Mongodump** - read data from database creates high fidelity BSON file, which **mongorestore** can use to populate a MongoDB database.

**MongoDB cloud manager** - continually backs up MongoDB replica sets and sharded clusters by reading oplog data. It has graphical UI.

##### Monitoring

**mongostat** - how many times database operations such as insert, query, update, delete, etc. occur on server. It gives an idea how much load the sever is handling.

**mongotop** - tracks and reports current read and write activity of a MongoDB instance, report these statistics on a per collection basis.

**severStatus** – returns an overview of the status of db, with details on disk usage, memory use, connection established, etc.

##### Indexing and Performance considerations

**Indexes** support the efficient execution of queries in MongoDB.

Without indexes, MongoDB must perform a **collection scan,** i.e. scan every document in a collection, to select those documents that match the query statement.

Collections come with a **default \_id index** (mongoDB’s unique id)

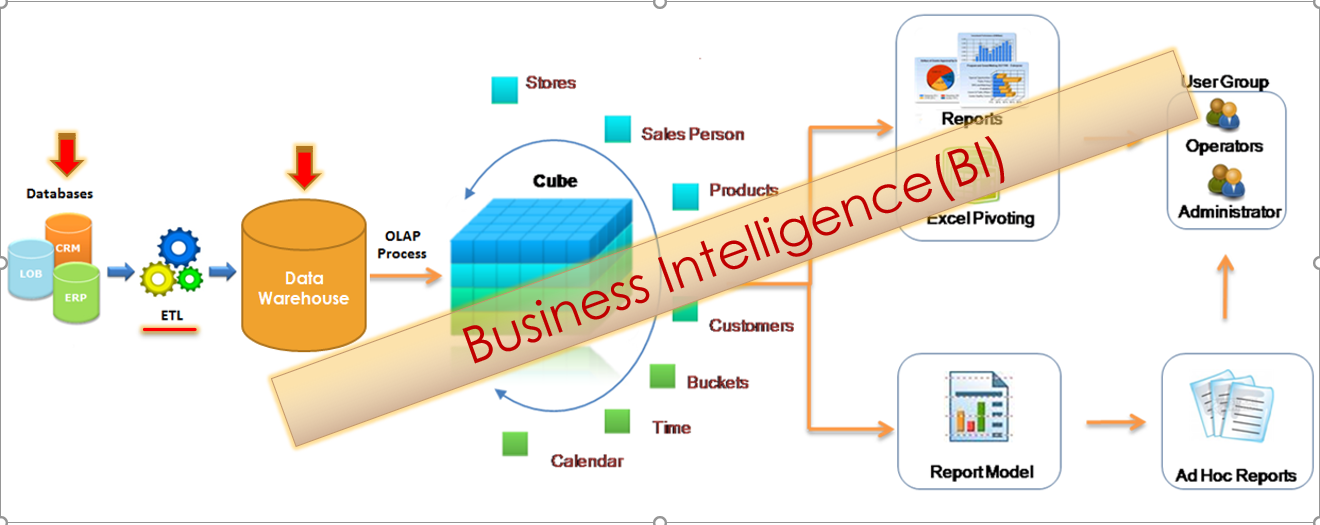
Shouldn’t create a lot of indexes as it can slow down the create/ delete/update operations.

#### Replication

It is the process of ensuring that the same data is available on more than one MongoDB Server

## Data Warehousing & Business Intelligence

### Data for Decision Making



### Data & Process Integration

**Data integration** aims at providing a unified view and/or unified access over heterogeneous, and possibly distributed, data sources

**Process integration** deals with sequencing of tasks in a business process but also governs data flows in these processes (e.g., loan approval process)

**control-flow:** correct sequencing of tasks

**data flow**: inputs/outputs of the tasks

Two types of **dependencies**

**sequence dependency**: execution of service B depends on the completion of the execution of service A

**data dependency**: execution of service B depends on data provided by service A

### Data Quality

Data quality can be defined as “**fitness for use**”:

**data accuracy**: to be accurate, a data values must be the right value and must be represented in a consistent and unambiguous form

**data completeness:** refers to an indication of whether or not all the data necessary to meet the current and future business information demand are available in the data resource

**data consistency**: ensures that the same data that is being used in different parts of the system will always be the same. The data is consistent across the entire system.

**data accessibility**: allows users to quickly find data in a (very) large DW

**data timeliness :** the availability and accessibility of data in making business decisions

### Data Warehouse

“A **data warehouse** (DW or DWH) is a subject-oriented, integrated, time-variant, and nonvolatile collection of data in support of **management’s decision-making process**.” (Inmon, 1996)

In computing, a data warehouse, also known as an enterprise data warehouse (**EDW**), is a system used for **reporting** and **data analysis,** and is considered a **core** component of **business intelligence**

DWs are **central repositories** of **integrated data** (current and historical ) from one or more disparate sources

#### Definition

|  |  |  |
| --- | --- | --- |
|  | **Transactional Database** | **Data Warehouse** |
| **Usage** | Day to day business operations | Decision support at tactical/strategic level |
| **Data latency** | Real-time data | Periodic snapshots, incl. historical data |
| **Design** | Application oriented | Subject Oriented |
| **Normalization** | Normalized data | Denormalized data |
| **Data manipulation** | Insert/Update/Delete/Select | Insert/Select |
| **Transaction management** | Important | Less of a concern |
| **Type of queries** | simple transactional queries | Simple and complex (ad-hoc) queries |

#### Data Warehouse Schemas

Star schema

Snowflake schema

Fact constellation



#### Specific Schema Issues

Surrogate keys

Granularity of the fact table

Factless Fact Tables

Optimizing dimensions

Defining junk dimensions

Defining outrigger tables

Slowly changing dimensions

Rapidly changing dimensions

### Extract, Transform, Load (ETL) Process

**Capture data** from multiple, heterogeneous sources and **integrate** into a single persistent store

ETL activities

**extract** data

**transform** data

**load** transformed data

Can **consume up to 80%** of all efforts needed to set up a data warehouse



#### ETL Process

**Extraction**

Full or incremental

**Transformation**

formatting

cleansing

aggregation and merging

enrichment

**Loading**

Fill the fact and dimension tables

**Documentation** and **metadata**

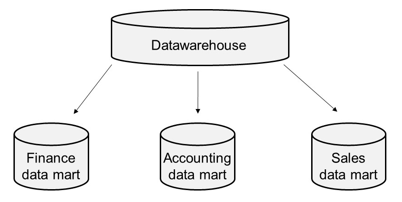
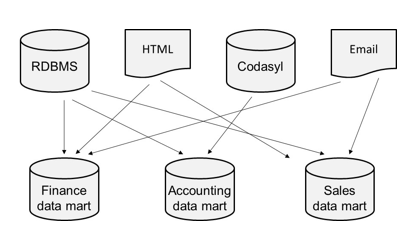
### Data Marts

A **data mart** is a **scaled down version** of a data warehouse aimed at meeting the information needs of a homogeneous small group of end-users such as a department or business unit

Provide **focused content + improve** **query** performance

**Dependent data marts** pull their data directly from a central data warehouse

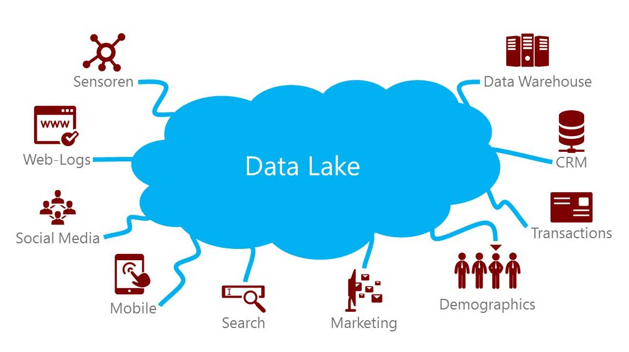
**Independent data marts** are standalone systems drawing data directly from the operational systems, external sources or a combination of both

### Data Lake

A data lake is a centralized repository that allows you to store all your **structured** and **unstructured** data at any scale

You can **store your data as-is,** without having to first structure the data, and run different types of analytics—from dashboards and visualizations to big data processing, real-time analytics, and machine learning to guide better decisions



#### Virtual Data Warehouses and Virtual Data Marts

Idea of virtualization is to use **middleware** to create a logical or virtual data warehouse or virtual data mart which has **no physical data** but **provides a single point of access** to a set of underlying physical data stores

Data is only accessed **(‘pulled’) at query time**

Virtual Data warehouse can be **built as a set of SQL views** directly on the underlying operational data sources

Disadvantages

Extra processing capacity from the underlying (operational) data sources

Not possible to keep track of historical data

### Business Intelligence

**Business intelligence (BI)** is referred to as the set of activities, techniques and tools aimed at **understanding patterns in past data and predicting the future**.

It’s all about **Data**. Garbage In, Garbage Out (**GiGo**)

BI techniques

(Ad hoc) Query and Reporting

Pivot tables

OLAP

Data Mining

#### Query and Reporting

Business user can **graphically and interactively** design a query and corresponding report

**Self Service BI**

Query by Example (**QBE**)

a query is composed in a user-friendly and visual way

**Reports** can be refreshed at any time

Innovative **visualization techniques**

#### Pivot tables

A pivot or cross-table is a popular **data summarization** tool

It essentially **cross-tabulates a set of dimensions** in such a way that multidimensional data can be represented **in a two-dimensional tabular format**

#### On-Line Analytical Processing (OLAP)

**OLAP** allows you interactively analyze the data, summarize it and visualize it in various ways

Provide the business-user with a powerful tool for **ad-hoc querying**

An **OLAP cube** is a multidimensional database that is optimized for data warehouse and online analytical processing (OLAP) applications.

Types

MOLAP

ROLAP

HOLAP

##### MOLAP

Multidimensional OLAP (MOLAP) stores the multidimensional data using a Multidimensional DBMS (MDBMS) whereby the data is stored in a multi-dimensional array-based data structure optimized for efficient storage and quick access.

Fast in terms of data retrieval

More storage space needed

Scales poorly when the number of dimensions increases

No universal SQL-like standard is provided

Not optimized for transaction processing

##### ROLAP

Relational OLAP (ROLAP) stores the data in a relational data warehouse, which can be implemented using a star, snowflake or fact constellation schema

ROLAP **scales** **better** to more dimensions than MOLAP

ROLAP **query** **performance** may however be inferior to MOLAP

##### HOLAP

Hybrid OLAP (HOLAP) tries to combine the best of both MOLAP and ROLAP

RDBMS used to store the detailed data in a relational data warehouse whereas the **pre-computed aggregated data** is kept as a multidimensional array managed by an MDBMS

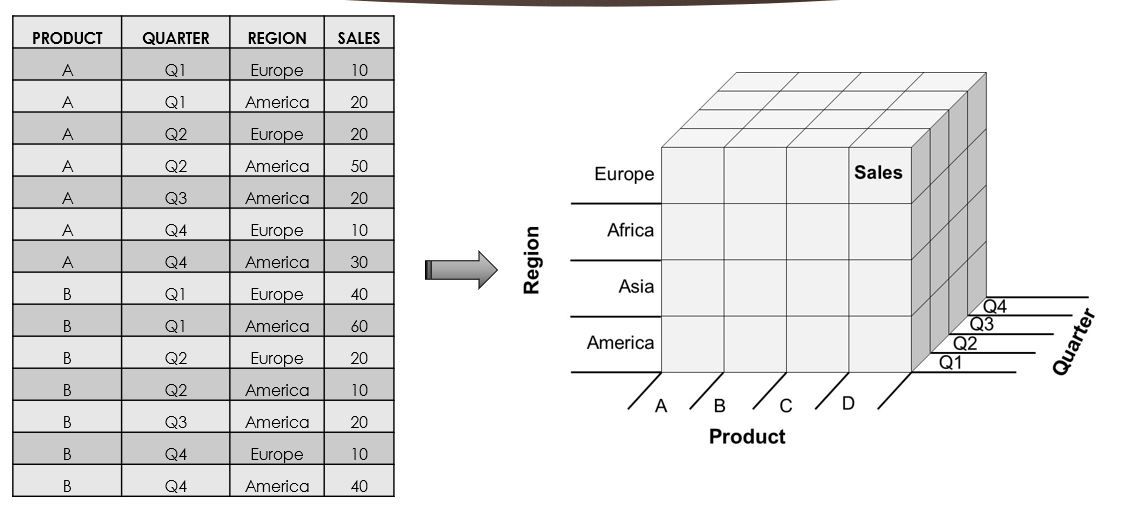
OLAP analysis **first starts** from the multidimensional database

**Combine the performance** of MOLAP with the **scalability** of ROLAP

##### MOLAP, ROLAP and HOLAP



##### OLAP Cube



##### OLAP operators

###### Roll-up & Drill-down

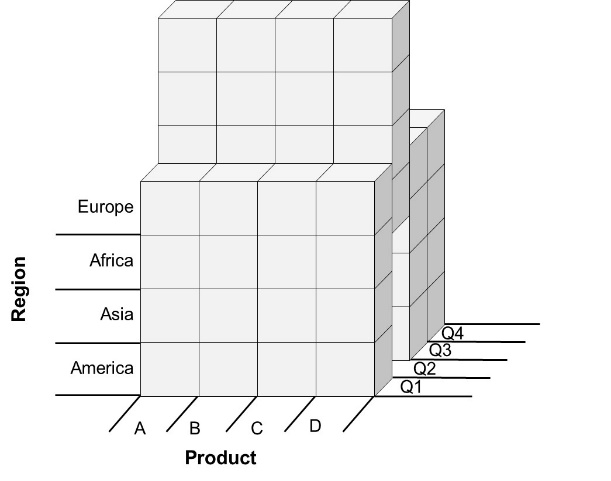
Drill-up (Roll-up) refers to aggregating the current set of fact values within or across one or more dimensions.

Drill Down refers to zoom in to more detailed data.



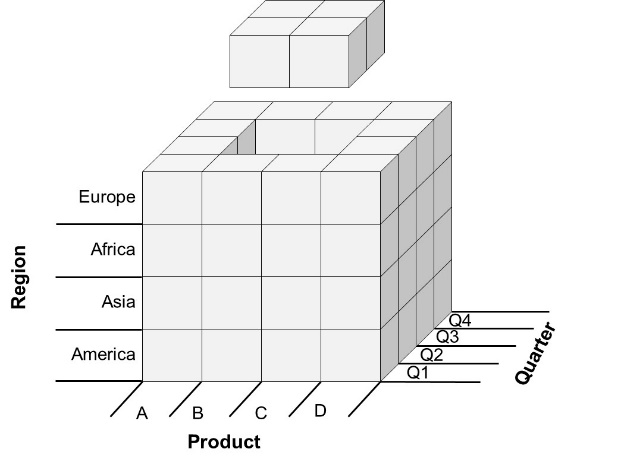
###### Slicing

Slicing represents the operation whereby one of the dimensions is set at a particular value.



###### Dicing

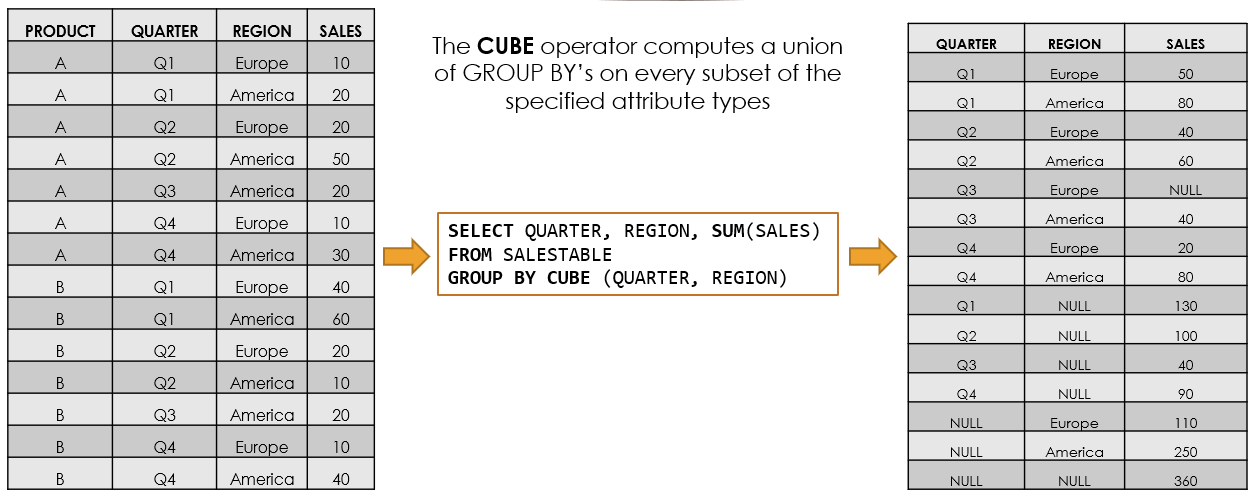
Dicing corresponds to a range selection on one or more dimensions



##### OLAP queries in SQL – GROUP BY

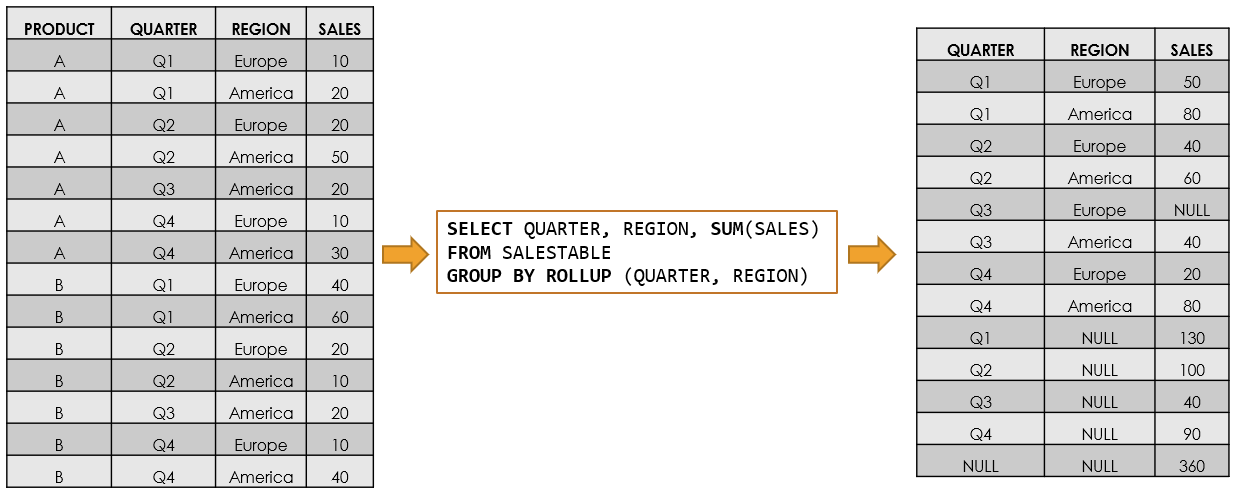
###### GROUP BY CUBE

The CUBE operator computes a union of GROUP BY’s on every subset of the specified attribute types



###### GROUP BY ROLLUP

ROLLUP operator computes the union on every prefix of the list of specified attribute types, from the most detailed up to the grand total

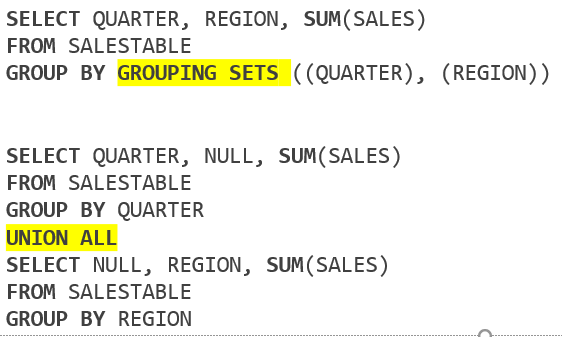


###### Key difference between the ROLLUP and CUBE

Key difference between the ROLLUP and CUBE operator is that the former generates a result set showing the aggregates for a hierarchy of values of the specified attribute types, whereas the latter generates a result set showing the aggregates for all combinations of values of the selected attribute types

###### GROUP BY – GROUPING SETS

GROUPING SETS operator generates a result set **equivalent** to that generated by a UNION ALL of multiple simple GROUP BY clauses



##### OLAP Queries in SQL

**Ranking**

RANK(): returns the rank of each row within the partition of a result set.

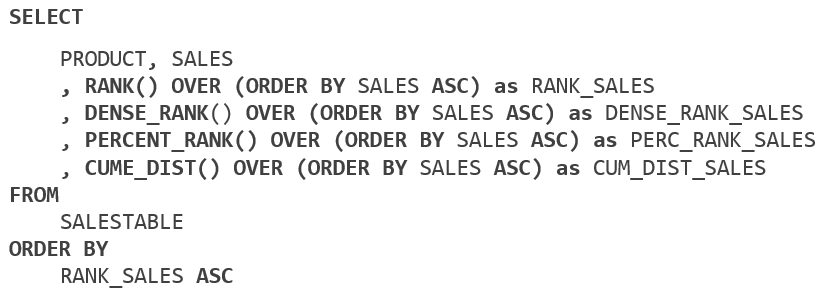
DENSE\_RANK(): returns the rank of each row within a result set partition, with no gaps in the ranking values.

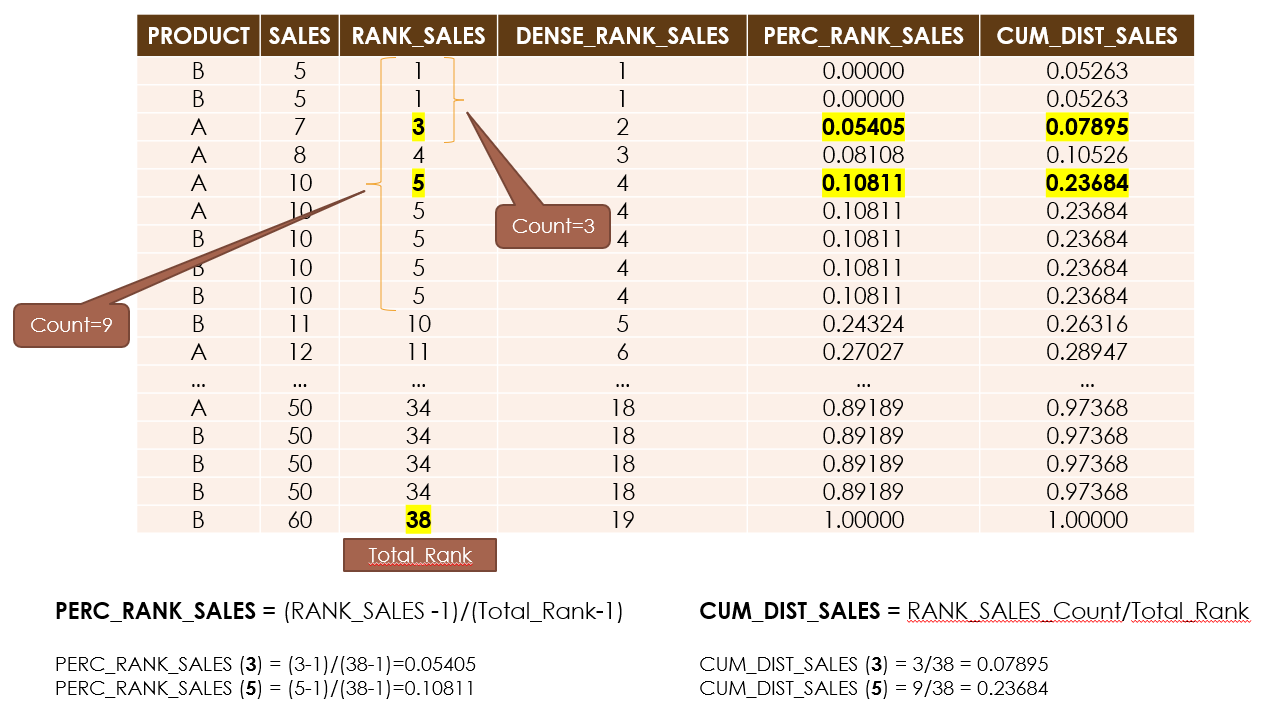
PERCENT\_RANK(): calculates the relative rank of a row within a group of rows.

CUME\_RANK(): calculates the cumulative distribution of a value within a group of values.

**Windowing** (Partition By)

###### RANK()





###### WINDOWING

Windowing allows calculating cumulative totals or running averages based on a specified time window

Windowing functions are a great way to **get different perspectives on a set of data without having to make repeat calls** to the server for that data

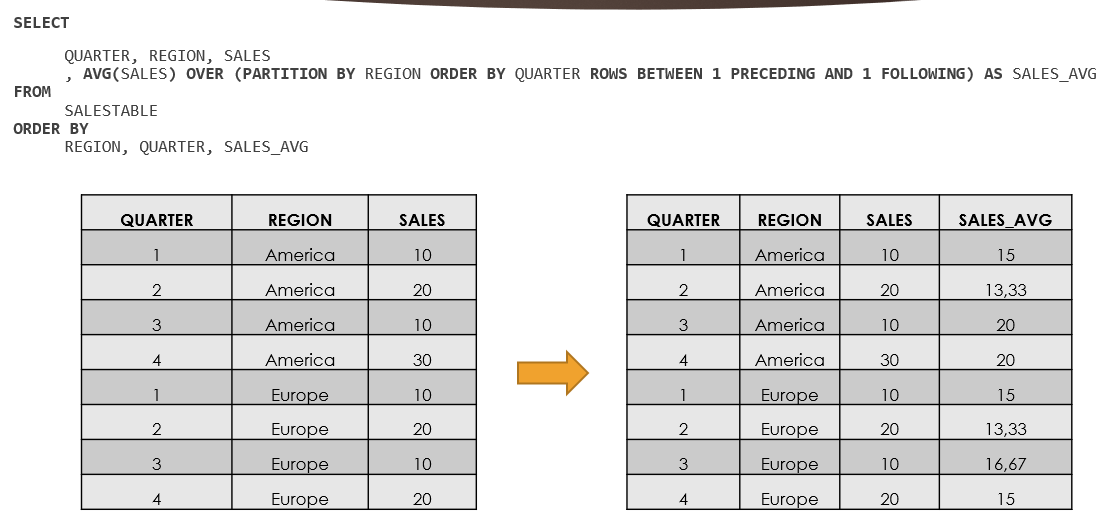
Window functions all use the **OVER**() clause, which is used to define how the function is evaluated

The OVER() clause accepts three different arguments:

**PARTITION BY**: Resets its counter every time the stated column(s) changes values

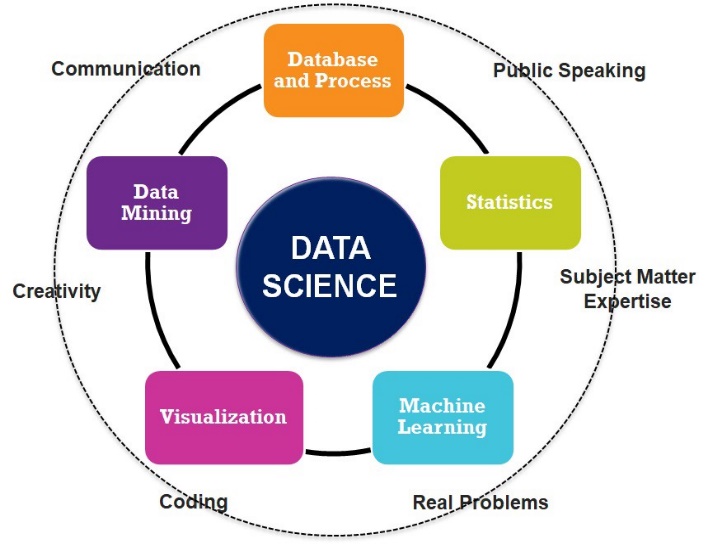
**ORDER BY**: Orders the rows the function will evaluate. This does not order the entire result set, only the way the function proceeds through the rows

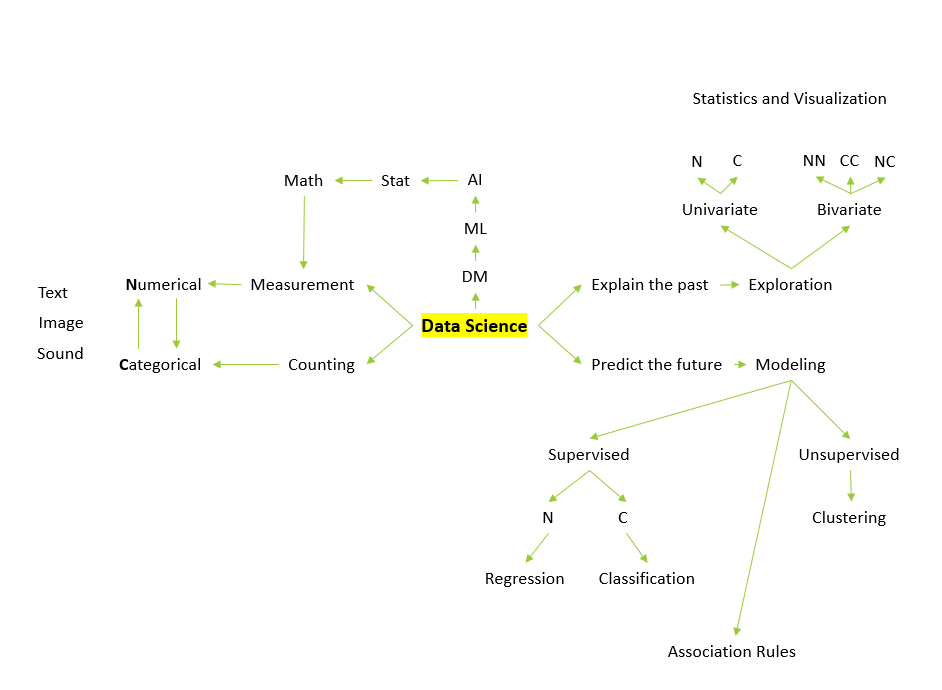
**ROWS BETWEEN**: Specifies how to further limit the rows evaluated by the function



## Data Mining

### Data for Exploration and Prediction





## Quiz

#### An external data model is also called a

view

#### Is a numerical attribute with patterns

Heart sound

#### A description of data in terms of a data model is called a schema. Where dose DBMS save its schemas?

Catalog

#### There are the main benefits using DBMS except

Artificial Intelligence

#### Information about the conceptual, external and physical schemas is stored in the system

Catalog

#### Represents the data in the database at a moment

Database state

#### A is a mechanism use to control access to database objects

lock

#### The DBMS must insure that the changes made by incomplete transactions are removed from the database. To do so, the DBMS maintains a of all writes to the databases.

log

#### When would you not store data in RDBMS

A company with a collection of 1 billion web pages

#### A is any one execution of a user program in a DBMS

transaction

#### RDBMS stands for

Relational database management system

#### Which data type is not valid

Class

#### A is an association among two entity types

Relationship type

#### A specifies a set of values that may be assigned to an attribute

domain

#### Is a part of enhanced entity relationship eer model but not er model

aggregation

#### Which one is the proper steps in database data model design?

Conceptual, logical, physical and external

#### Specify the minimum or maximum number of relationship instances that and individual entity can participate in.

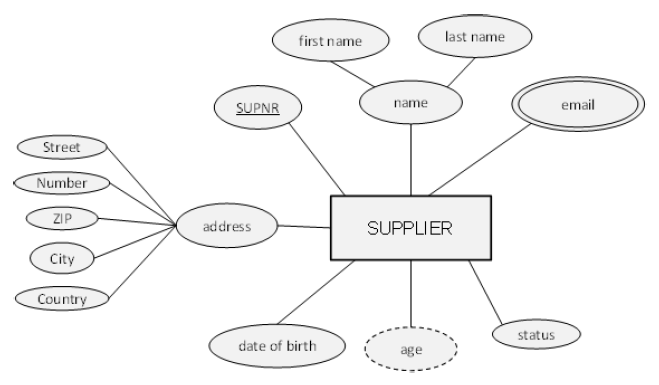
cardinality

#### These are all limitations of er model except

Er model cannot model weak entities

#### A is a rule that forbids duplicate values in one or more columns within an entity and cannot contain NULL values.

Primary key constraint



#### Is an entity type

supllier

#### Is a derived attribute type

age

#### Is a key attribute type

supnr

#### Is a multi-valued attribute type

email

#### Is a composite attribute type

Address

#### These are all related to integrity constraint except

Restrict key

#### Intended to guarantee validity of data in a database even in the event of errors, power failures, media failure and more

ACID

#### Which normal form takes care of dependency of two non-prime attributes transitive dependency

3NF

#### Which normal form takes care of dependency of a non-prime attribute to a partial prime attribute partial dependency

2NF

#### Which normal form takes care of dependency of a prime attribute to a non-prime attribute

BCNF

#### Means that the results of applying a transaction are permanent, even in the presence of failures

durability

#### Is not a DDL statement

DELETE TABLE

#### Is a process of organizing the data in database to avoid data redundancy and insertion, update and deletion anomaly

Normalization

#### Is not a DML statement

DROP VIEW

#### Which join method is valid

Left/right/self/inner/full

#### A non-prime attribute is

An attribute which is not part of any candidate key

#### Fill the blank for [SELECT…FROM…WHERE…GROUP BY…]

Attributes, entities, filters, attributes

#### Tests for the existence of any record in a subquery

EXISTS

#### Gives a column or table a temporary name (alias)

AS

#### Returns rows that have matching values in both tables

Inner join

#### Is a DDL statement

DROP TABLE

#### A constraint that limits the value that can be placed in a column

CHECK

#### Is a filter for SQL aggregate statements

HAVING

#### Copies data from one table into a new table. The new table will be created with the column-names and types as defined in the old table

SELECT INTO

#### Searches for a specified pattern in a column

LIKE

#### Is not a DML statement

ALTER VIEW

#### Combines the result set of two or more SELECT statements

UNION

#### Copies data from one table and inserts it into an existing table

INSERT INTO

#### Returns all rows from the right table, and the matching rows from the left table

RIGHT OUTER JOIN

#### All the following SQL statements should be executed by users except

triggers

#### AVG is

System aggregate function

#### Defines TIMING in a trigger

AFTER

#### Opening a on a result set allows processing the result set one row at a time

cursor

#### Is not an TCL statement

END TRANSACTION

#### Encloses a series of SQL statements so that a group of SQL statements can be executed

BEGIN…END

#### The SQL statement or statements that follow are skipped and processing continues at the label

GOTO

#### Provides a structured method of evaluating a list of options and then returning a single value

CASE-[WHEN-THEN]-ELSE-END

#### You can run from the database’s command environment using the exec command

Stored procedures

#### That cause triggers to be activated include insert, updata, delete, create, alter and drop

Events

#### Is not a TRIGGER statement

DENY

#### Which one is the proper sequence of statements for a SQL Cursor

DECLARE > OPEN > FETCH > CLOSE > DEALLOCATE

#### NoSQL is about all the following items except

Fixed schema

#### Means that the system continues to function even the communication among the servers is unreliable

Partition tolerance

#### Means all operations in a transaction succeed or every operation is rolled backc

Atomicity

#### Means that the system is always on (service guarantee availability), no downtime

availability

#### Means that the results of applying a transaction are permanent, even in the presence of failures

durability

#### They are all NoSQL databases except

List-oriented

#### They are the typical Key-value API functions except

create

#### Which DBMS is column-oriented

Redshift

#### Copies the entire database across all nodes in the distributed system

replication

#### Tables in a mysql is equal to in mongodb

collections

#### Relationships in a mysql is equal to in neo4j a graph dbms

relationships

#### Divides the data inside the database and partitions pieces of it to different nodes

sharding