* CRUD

Create, read, update, delete

* Database

A **database** is a data structure that stores organized information. Most **databases** contain multiple tables, which may each include several different fields

* Database management system (DBMS)

A **database management system** (**DBMS**) is a software package designed to define, manipulate, retrieve, and manage data in a database.

* Big data

**Big data** is a term that describes the **large** volume of **data** – both structured and unstructured – that inundates a business on a day-to-day basis

* Database query

A **database query** is a similar action that is most closely associated with some sort of CRUD (create, read, update, delete) function. A **database query** is a request to access data from a **database** to manipulate it or retrieve it.

* Update anomaly

Poor normalisation or no normalisation. Insert, update and delete anomalies eg the same data is stored in two tables and we update one(inconsistency) or we delete some information and inadvertently also delete other critical information.

* ACID database properties

Atomicity – succeeds or fails  
consistency – date is calid and conforms to business rules  
isolation – transaction cannot be impacted by another transaction  
durability – transaction does not disappear even if there is crash after rollback and redo logs

* Transaction management

**Transaction management** [1, 2] refers to the tasks of processing multiple **transactions** issued by various clients of a database server in such a way that the ACID contract can be fulfilled

* Concurrency Control

Two processes attempt to update the same row at the same time.

|  |  |
| --- | --- |
| **Relation** | **Table** |
| relation schema | table header |
| relation instance | whole table |
| attribute | column |
| tuple | row |
| tuple component (value) | cell |

* Relation: A relation is a “table” with columns and rows.
* Attribute: An attribute is a named column of a relation.
* Domain: A domain is the set of allowable values for one or more attributes.
* Tuple: A tuple is a row of a relation.
* Degree: The degree of a relation is the number of attributes it contains.
* Cardinality: The cardinality of a relation is the number of tuples it contains.
* Relational database: A collection of relations with distinct relation names.
* Relation Schema: A named relation defined by a set of attribute and domain name pairs.
* Relation Instance: A relation with its schema AND all tuples at any given point in time is defined as the relation instance.
* Relational Database Schema: A set of relation schemas, each with a distinct name
* Atomic attribute

An **attribute** that cannot be divided further into meaningful subcomponents/sub-**attributes** is an **Atomic attribute**. Following **attributes** are some of the examples of **atomic attributes**; First\_name, last\_name of a person, names of things etc. – a person's name may be divided into first name, last name and middle name etc.

* Composite attribute

**Composite attribute** is an **attribute** where the values of that **attribute** can be further subdivided into meaningful sub-parts." Typical examples for **composite attribute** are; Name – may be stored as first name, last name, middle initial.

* Single valued attribute

**Attributes** that can have **single value** at a particular instance of time are called **single valued**. A person can't have more than one age **value**.

* Multi-valued attribute

A **multivalued attribute** can have more than one **value** at a time for an **attribute**. For ex., the skills of a surgeon is a **multivalued attribute** since a surgeon can have more than one skill. Another common example is the address field, which can have **multiple** values like zipcode, street address, state, etc.

* Entity relationship model

The **entity relationship** (**ER**) **data model** has existed for over 35 years. ... **ER models**, also called an **ER** schema, are represented by **ER** diagrams. **ER modelling** is based on two concepts: **Entities**, defined as tables that hold specific information (**data**) **Relationships**, defined as the associations or interactions between **entities**.

* UML notation

The **UML notation** is a **notation** conceived for modeling object of applications and continue and extend, in particular, the **notations** of OMT (Object Modeling Technique) and Booch methods. More precisely, here we describe the principles of the use-case diagrams, classes, objects and sequence diagrams.

* Entities

**Database entity** is a thing, person, place, unit, object or any item about which the data should be captured and stored in the form of properties, workflow and tables. While workflow and tables are optional for **database entity**, properties are required (because **entity** without properties is not an **entity**).

* Relationships

There are three types of relationships between the data you are likely to encounter at this stage in the design: **one-to-one**, one-to-many, and many-to-many. To be able to identify these relationships, you need to examine the data and have an understanding of what business rules apply to the data and tables.

* Attributes

You can think of an attribute as a column in an **entity** table. An attribute **value** is the **value** used to describe a specific member. When you create an **entity** that contains many attributes, you can organize the attributes into attribute groups

* Constraints

SQL constraints are used to specify rules for the data in a table. Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.