perfect database design example -> **website example**

if I delete anyone doesn’t active in a year / give bonus 100$ to 3 employe

search manually each employee is difficult, so to make it easy we use database.

----------------------------------------------------------------------------------------------------

**Relations** is connection between data.

relation in database comes from mathematical concept, but in database rather than combining number **we're combining attributes.**

2 4

| |

6 62 64

| |

8 82 84

----------------------------------------------------------------------------------------------------

**entity** is what (anything) we store data about.

**attribute** is what (thing) we store about entity.

so, if entity is a person then attributes are username, password, address, phone number

**we have attribute types and attribute values.**

row contains all attribute values for specific entity / row is taking about one entity.

column value for specific attribute type.

attributes are columns.

entities are individual rows.

entity type of table is can be user or sale or …..

if every single row within the table should be a user, then entity type of table is user.

----------------------------------------------------------------------------------------------------

**NULL**: doesn’t enter a value within a column on a table.

ID | fax\_num

7 30

6 -> this emptiness is called NULL

4 20

(Not everyone has a fax number)

---------------------------------------------------------------------------------------------------

**entity type / attribute type**

type means category.

-entity type is user if every single entity (every row) should be a user

-attribute type such as username-password-name

no one's username is username / no one's name is name.

----------------------------------------------------------------------------------------------------

**RDBMS**: Relational Data Base Management System

-specific type of DBMS is designed to work with relational databases.

DBMS: manage data, control data, manipulate data, allow us to view it in a human friendly way.

- allow search for values/change appearance of data

-for an example we have 6 billion users, we want to delete anyone who hasn't been active

\*DBMS is what allows us to easily run a query

---------------------------------------------------------------------------------------------------

**View mechanism**: allow us to change the way our data is presented.

view mechanism: allow us to create different views

-with a view mechanism -> we're able to get different views of data

ID |Username | Password | Email --> attributes in user table

if we had two people accessing this data one is jim and other one is jamie.

-jim is doesn't care about email, because all he's working on is to find information about username and password

So jim will create a new view of username and password, this is a specific view (selecting username and password)

-jamie cares about ID and Email So jaime will create a new view of (ID and Email)

security feature ->who can access and change.

not every person who use database has the privilege of creating a new view

-access data base doesn't mean you can update data only the administrator or the owner of the business is able to do.

---------------------------------------------------------------------------------------------

**Security feature**

-allow users can only access their username and password, they can't access everybody's username and password only theirs. But jim who is hired by the company can access the username and password of every single user.

This is a security, not allowing the user of the website to get too much information and at the same time not limiting jim too little information.

--------------------------------------------------------------------------------------------

RDMS allow us to do **transaction**.

when you do something with the data, it is completed all the way or it doesn't work at all

-if we have multiple step thing, such as transferring money to an account, if any time power goes out and the server crashes and it doesn't complete --> then the transaction is canceled, nothing is saved

-Examples of RDBMS -->SQL (Microsoft server), MYSQL, oracle

--------------------------------------------------------------------------------------------------

Difference between Database/DBMS

Database is actual files are stored on your computer or hard drive.

RDBMS takes the data on the hard drive and put into presentable tables.

----------------------------------------------------------------------------------------------------

on a website, someone put his full name in one box, and that can be stored on a database as a two separate columns -> first name - last name, if we decided to make it one column->full name, the front end didn't change because the only thing that changed is the back end

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Introduction to SQL**

SQL: is programming language used for RDBMS

SQL 1) **DDL**->Defines database structure (Table)

2) **DML**->Manipulate data (insert-delete-search-update)

main structure of the database->DDL

data in the database ->DML

DDL-> CREATE (create table)

UPDATE->DML (because I change data not in table structure)

بدل ما يكون عندي جدول واحد في الاف المعلومات

break table into separate tables.

**JOIN**: collect data from multiple tables to get a new view of larger table-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Naming Conventions** مش قاعدة براحتك

anything I name -> use small letters.

don’t use space and use underscore

user\_id | name | address

**Database Design**

What is Database Design?

Why do we have to design a database?

How do you measure whether a database is good or bad?

we don't want our database have data integrity or security issues

has to do with **Data integrity**: all data in database is correct.

good design database, we prevent data integrity problems.

-we don't have repeating data

-we want the database to always be up to dateالداتا يحصلها update علي طول

if you have repeating data, then database design badly

for example, we have two address for one person, which should just be one address -> data integrity problem

-address is on the database twice for one person, that's a problem because when I update my address, only one may update

--------------------------------------------------------------------------------------------

**Database Design schemas:** Schema tells you the way the data is structured.

1) **conceptual Database Design**

just talks about how data is related / relationships between tables(general)

connection or relation between two entities is type of conceptual design

2) **logical Database Design**

we are structuring(drawing) columns, data types, relationship.

3) **physical** **Database Design**

(More specific) what kind of RDBMS will use or will work best?

what are our table types? what are server?

how are people going to access this? what are different views going to be?

when we start implementing database?

-it's a bad design to put all data in only one table why?

|  |  |  |  |
| --- | --- | --- | --- |
| name | ID | Phone | address |
| zahraa | 1 | 300 | alex |
| zahraa | 1 | 200 | alex |

if one person has two numbers, we can put only one so we need to create a new row and this will lead to duplicate data this is data integrity issue, that's a bad database design

now if I decided to update my address, which one will be updated.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Data Integrity**

if we don't have data integrity, then we will have errors.

if data is integrity, then no repeating values or breaking relationships between tables

if we have two tables, user table and sale table.

-sale is when a user buys product, that means a sale needs a user, we can't have a sale if no one bought it,then we have a relationship between sale and user

user <----- sale

-let's say sale was bought by user with ID of 7 and we have a user of ID of seven, and there is a connection between two tables -> that's data integrity

-if connection between user and sale is broken or user of ID of 7 is removed, that is an example of integrity issue

relational database doesn't come from relationship, it comes from relations.

relational database has relationships between tables.

-------------------------------------------------------------------------------------------

-**Three main type of data integrity** 1) Entity integrity

2)Referential integrity

3)Domain integrity

**1)entity integrity**: unique entity with every row in the same table

|  |  |
| --- | --- |
| name | Phone |
| cabel | 800 |
| cabel | 800 |

we have two people with the same phone number.

ممكن يكون شخصين عايشين مع بعض في نفس البيت

or we have two rows talking about the same person

->away to solve this to add another colomn where the row will be unique, such as national number or ID

----------------------------------------------------------------------------------------------------

2)**Referential integrity**: keeps the connections, the keys, foreign keys, primary keys

known as foreign key.

when we reference the user\_id of a table and user\_id of the another table

user <------------- comment

user

|  |  |
| --- | --- |
| user\_id | name |
| 6 | joe |

Comment

|  |  |
| --- | --- |
| User\_id | comment |
| 6 | thanks |

user put a comment->user is parent.

child can’t exist without the parent.

comment->user\_id know me person who posted that comment so, we have a relational integrity.

----------------------------------------------------------------------------------------------------

3)**Domain integrity**

**domain**: acceptable values to store for a column for an example a phone number phone should be 11 digits and should be numbers.

if I put cake in the phone number column->this is going to cause an error if we have domain integrity because it isn't 11 digits and it isn't numbers

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Data types

numbers, text, or dates

-data type put limits on what we're allowed to store

char (20)->this is domain integrity.

-foreign key constrains->if user is removed, also remove all comments by this user

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Database Terms (terms review)**

data: anything we store in a database.

database: what we store our data in

relational database: store data in table

DBMS: control database

RDBMS: control relational database

control tables and values in table

NULL: no value

Anomaly: error in data integrity

for example, when we update something instead of updating column 1, it updates column 10.

integrity: we implement database integrity to protect against anomalies

\*entity integrity: uniqueness in the table

\*Relational integrity: connection among multiple tables(relationships)

\*Domain integrity: column has its expected value for example phone number not a text or not a date

--------------------------------------------------------------------------------------------------

**(Design review)**

Entity -> anything we store data about.

user/credit card/comments/sales

attributes -> things we store about entity.

attribute of comment->date/number of characters/who posted.

\*Relation->another name for a table

tuple: all attributes about specific entity

name | phone | fav\_color |city

cabel 1800 red Mansoura => all these attributes about one entity is tuple

table: to organize data.

record =tuple=row=entry

column: attribute

file: another name for a table

table=relation=file

record: another name for a row

field: another name for a column

column =field

value: information that we put in a specific column.

entry: another name for a row

Database Design: remove anomalies.

**schema:** drawn out the structure of database.

**normalization:** bunch of steps help us to get the best database design/process of building the best database design.

naming convention: just a consistency تناسق

**keys**: everything unique in the table

how we make connections among tables

\*Entity can be individual row or a table type

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Frontend: what the user sees

Backend: what behind the scene (server side)

client side & sever side.

sever side scripting language.

different privilege for different people->view

I can access my password and email but that might be private to other->**view**.

\* When we use join, we get just a new view, it's not actually stored in the database

->views often use joins

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Atomic values**: store one thing

everything we store in a database should be one thing.

every column should be about one thing

if we have a column called name, its value zahraa ragab kamal-> it's not atomic because we're storing three names within column name

to make it atomic, the best thing to break it up into first\_name middle\_name last\_name

address to be atomic ->

|  |  |  |  |
| --- | --- | --- | --- |
| street | city | state | area\_code |

-if we have a column called favorite movies, this isn't going to work

every single column header should be singular ->favorite movie not favorite movies

not to store multiple things within a cell

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**relationships**: everything is connected.

one table break into two tables should be there a relationship.

relationships talk about entities.

database set of tables or entities that should be related.

three types of relationships 1) one to one

2)one to many

3)many to many

----------------------------------------------------------------------------------------------------

**one-to-one relationships**

one entity has a collection with one other entity.

for an example -husband has one wife

every husband has a one wife and wife has a one husband.

-every person has a one national number and national number assigned to one person

----------------------------------------------------------------------------------------------------

**one-to-many relationships**

one entity has a relationship with multiple other entities.

for examples -> user and comments -> one to many relationship

user can make many comments, but comment is owned by one person.

-------> comment

user ------> comment

-------> comment

one to many

user + comment -> one to many

comment + user-> one to one

husband has multiple wives and wife has one husband

husband and wife => one to many لو متجوز اكتر من واحدة

wife and husband => one to one

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**many-to-many Relationships**

one to many من الطرفين

husband can marry many wives -> one to many

wife can marry many husbands-> one to many

course and student-> many to many

student + course ->one to many

course + student-> one to many

class and student-> many to many

->**many to many relationships doesn't work in relational database**

**relational database deals only with** 1) one to one

2)one to many

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*