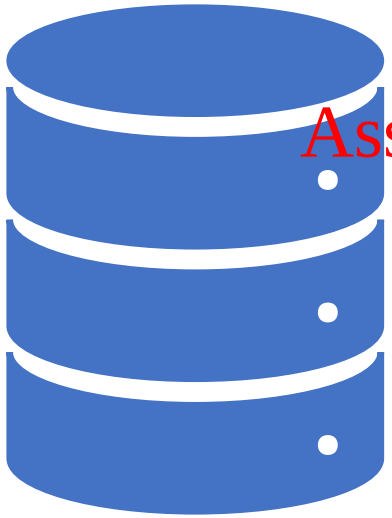
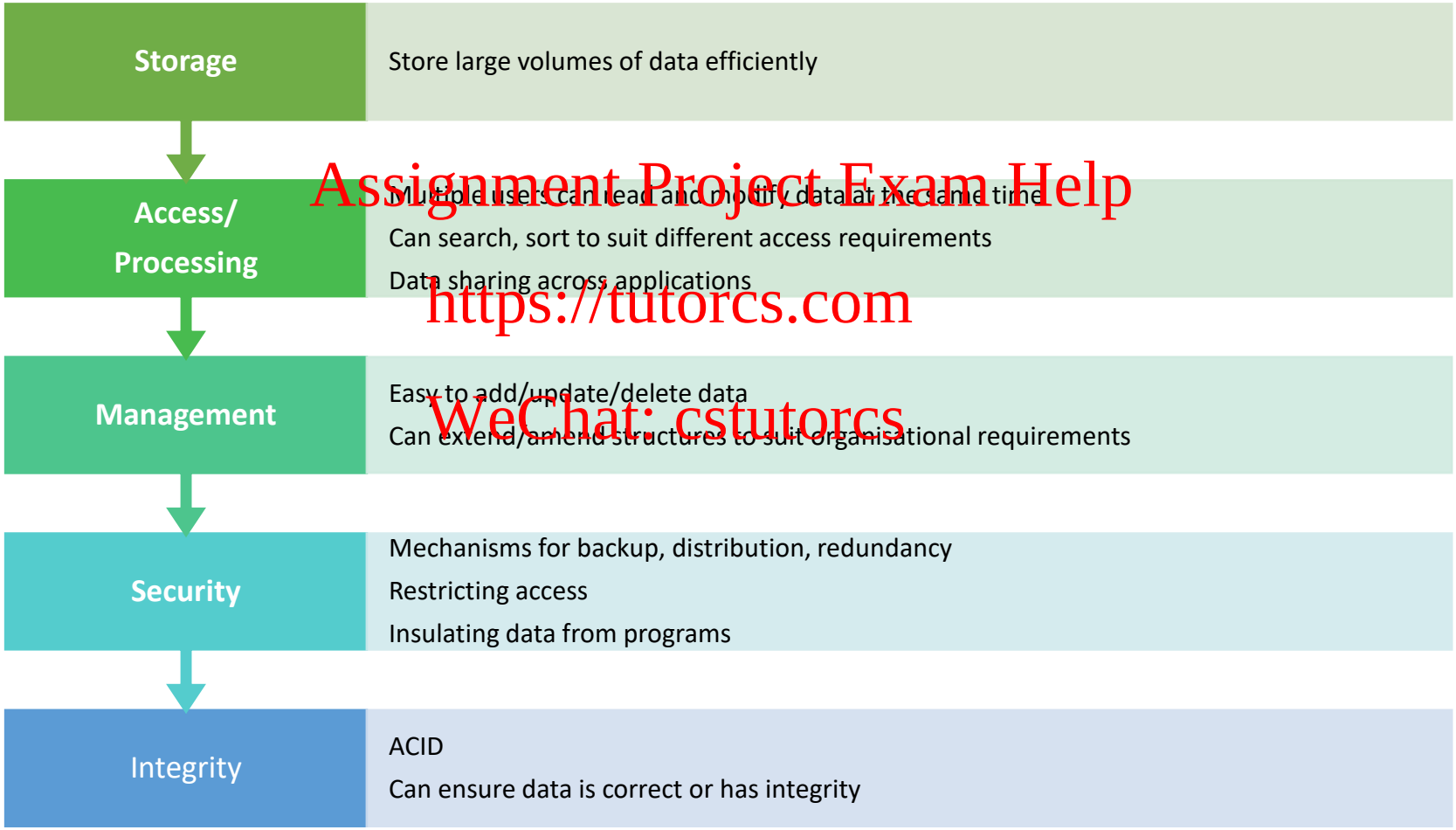
Advanced Databases

DB Design: Normalization



Why are databases/DBMS useful?

Why are databases/DBMS useful?

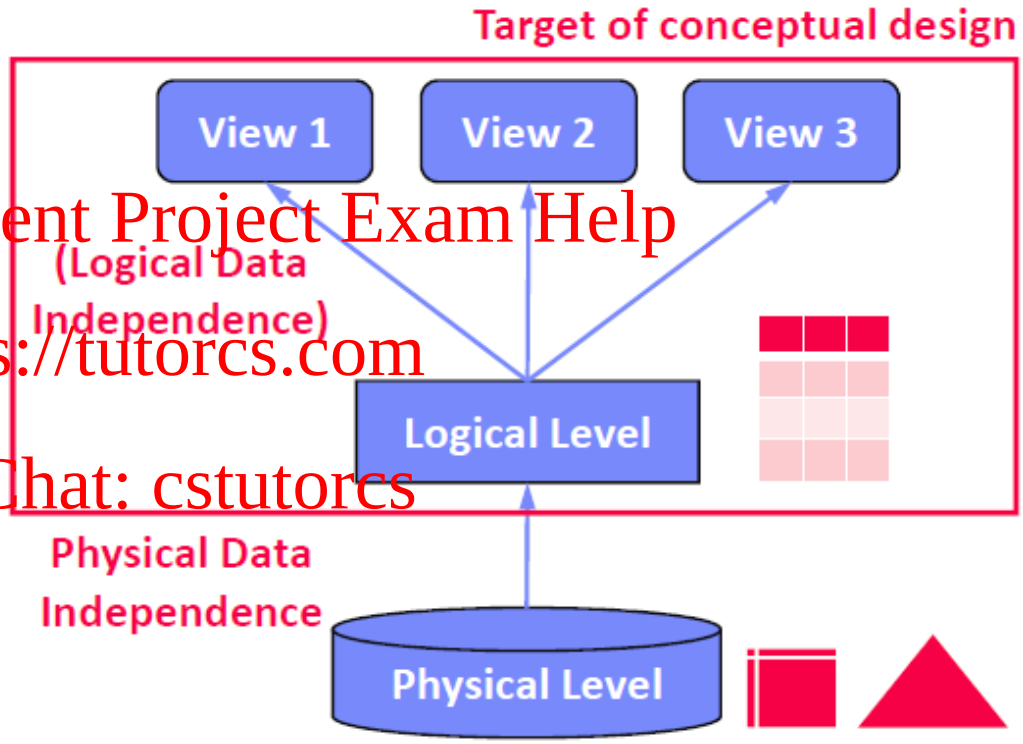


Why are databases/DBMS useful?

- Integrity
  - **A**tomicity
    - When changing data, if any part of the change fails, the whole change will fail.
    - Therefore the data will remain as it was before the change was attempted
    - Prevents partial records being created
  - **C**onsistency:
    - Before data can be changed in a database, it must be validated against a set of rules
  - **I**solation:
    - Multiple changes are possible at the same time, but each change is isolated from others
  - **D**urability:
    - Once a change has been made, the data is safe, even in the event of system failure

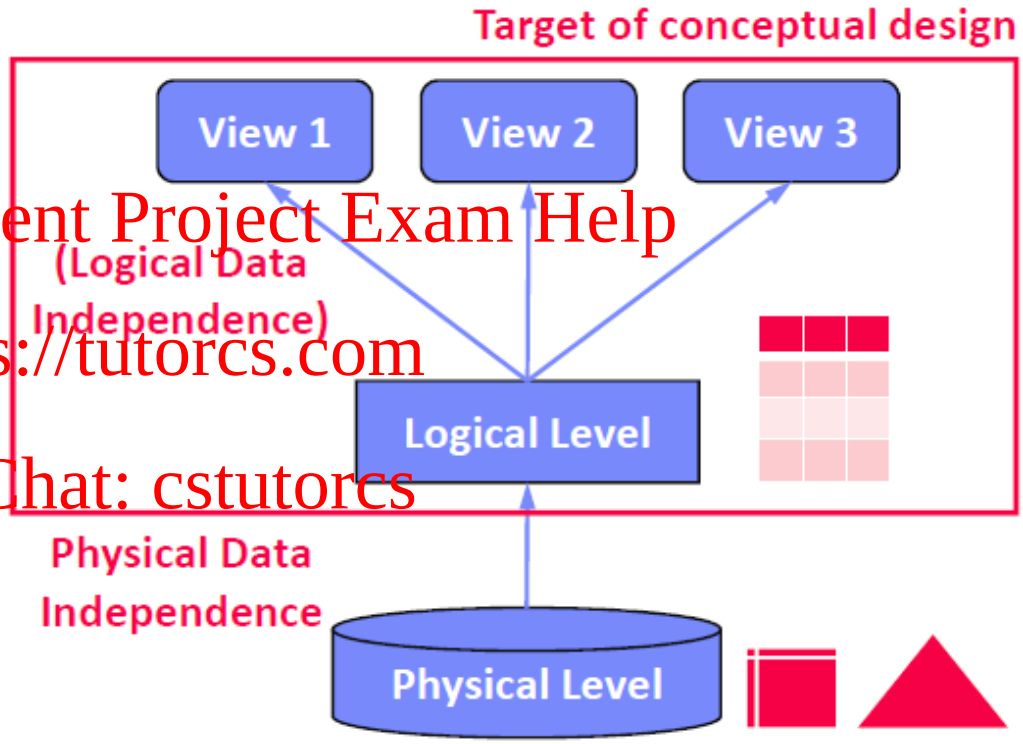
Quick Recap

- Three Layer ANSI-SPARC Architecture
  - 3 levels of abstraction
  - aims to separate each user's view of the database from the way the database is physically represented
- External schemas
  - (external level)
- Conceptual schema
  - (logical level)
- Internal schema
  - (physical level)



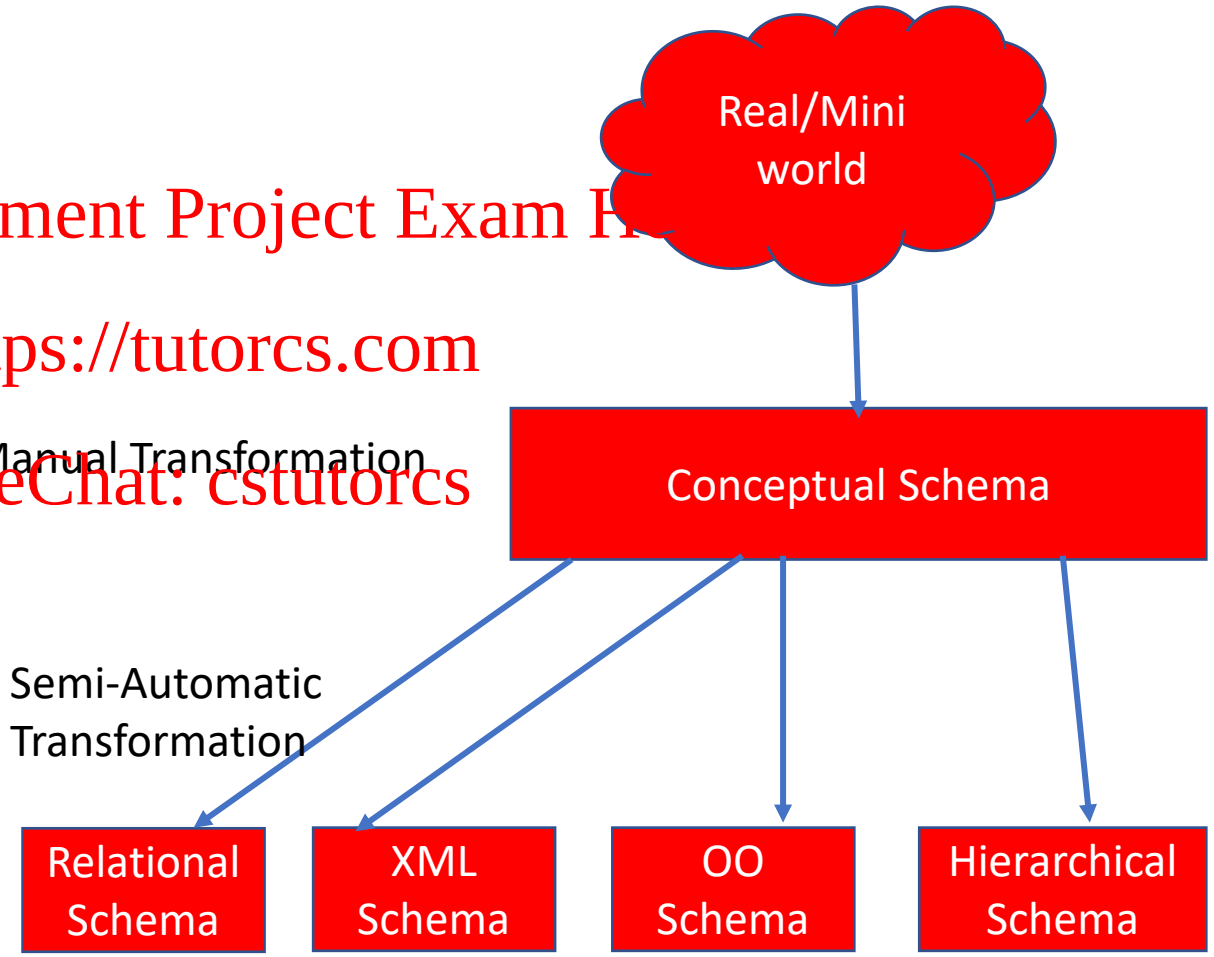
Quick Recap

- Types of Data Independence
- Logical data independence
  - (external views and applications independent of logical data model)
- Physical data independence
  - (logical data model independent of underlying data organization)



Quick Recap

- **Data Model:**
  - Concepts for describing data objects and their relationships (meta model)
- **Schema:**
  - Description (structure, semantics) of specific data collection



Quick Recap

- Conceptual Data Models
  - Entity-Relationship Model (ERM), focus on data, ~1975
  - Unified Modeling Language (UML), focus on data and behaviour, ~1990
- Logical Data Models
  - **Relational**
  - **Key-value**
  - **Graph**
  - **Document (XML, JSON)**
  - Matrix/Tensor
  - Object-oriented
  - Network
  - Hierarchical

Quick Recap DB Design

1. Requirements engineering
   - Collect and analyze data and application requirements
   - Specification documents
2. Conceptual Design
   - Model data semantics and structure, independent of logical data model
   - ER model / diagram
3. Logical Design
   - Model data with implementation primitives of concrete data model
   - e.g., relational schema + integrity constraints, views, permissions, etc
4. Physical Design
   - Model user-level data organization in a specific DBMS (and data model)
   - Account for deployment environment and performance requirements

Database Normalization

Normalization is a design technique

- *"..a very important ingredient in database design",* Coronel, C., & Morris, S. (2016). *Database systems: design, implementation, & management*. Cengage Learning.

Objectives

- Eliminate redundant data (storing the same data in more than one table)
- Ensure data dependencies make sense (only storing related data in a table)

Focus

- Correct assignment of attributes to tables.

How

- Considering the rules of the real world
- Examining the actual values attributes can take

Why is it important?

- If you don't normalize, databases can be inaccurate, slow, and inefficient and they might not produce the data you expect.

Anomalies

emp_id	emp_name	emp_address	emp_dept
101	Rick	Dublin 6	D001
101	Rick	Dublin 6	D002
123	Maggie	Dublin 7	D890
166	Glenn	Dublin 8	D900
166	Glenn	Dublin 8	D004

- Update anomaly
  - We have two rows for employee Rick as he works in two departments of the company.
  - If we want to update Rick's address then we have to update two rows or the data will become inconsistent.
  - If somehow, the correct address gets updated for one department but not the other then as far as the database is concerned Rick has two different addresses – this is not correct or consistent with the real world we are modelling.

Anomalies

emp_id	emp_name	emp_address	emp_dept
101	Rick	Dublin 6	D001
101	Rick	Dublin 6	D002
123	Maggie	Dublin 7	D890
166	Glenn	Dublin 8	D900
166	Glenn	Dublin 8	D004

- Insert anomaly
  - Suppose a new employee joins the company
  - They are not assigned to any department while they are training and there is no official training department
  - At the moment we would not be able to insert the data into the table if emp_dept field doesn't allow nulls.

Anomalies

emp_id	emp_name	emp_address	emp_dept
101	Rick	Dublin 6	D001
101	Rick	Dublin 6	D002
123	Maggie	Dublin 7	D890
166	Glenn	Dublin 8	D900
166	Glenn	Dublin 8	D004

- Delete anomaly
  - Suppose the company closes the department D890
  - Deleting the rows that have emp_dept as D890 would also delete the information of employee Maggie since she is assigned only to this department

Database Tables and Normalization

- Normalization works through a series of stages called normal forms:
  - 1NF (First Normal Form)
  - 2NF (Second Normal Form)
  - **3NF (Third Normal Form)**
  - BCNF (Boyce-Codd Normal Form)
  - 4NF (Fourth Normal Form)
  - 5NF (Fifth Normal Form)
  - 6NF (Sixth Normal Form)
- The higher levels of normalization are not always advisable.

How do you decide?

- Objectives of DB Design
  - Arrange data into logical groups such that each group describes a small part of the world represented by the data
  - Minimize the amount of duplicated data stored in a database
  - Build a database which allows you to access and manipulate the data quickly and efficiently balanced with maintaining the integrity of the data stored
  - Organise the data so that, when you modify it, you make the changes in only one place
- Normalization: Helps achieve data integrity, referential integrity, or keyed data access.

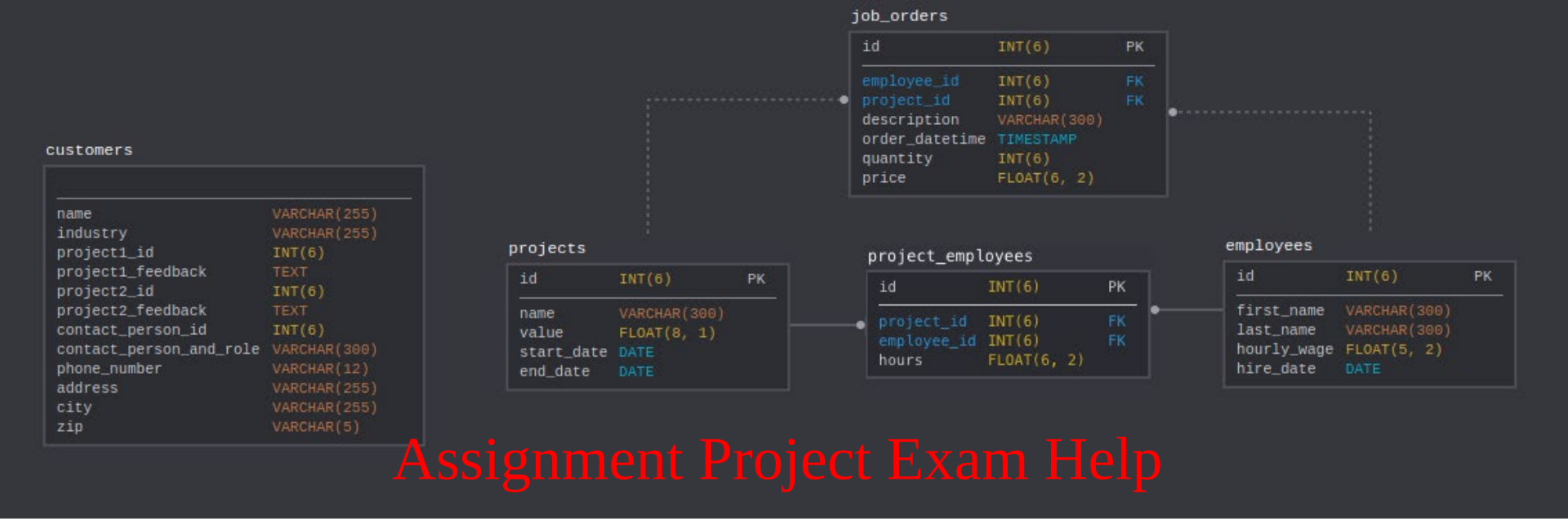
Entity v Relation v Table

- Entity:
  - During the very first step in the design process, when you're creating a conceptual design you have a bunch of entities and relationships that represent the various types of data you'll want to store (and express it as an ERD).
- Relation:
  - Once you've finalized your conceptual design you'll convert that ERD into a logical schema (and express it as an ERD).
  - This schema will be a list of relations.
  - The relations are all your entities and relationships from the previous step .
- Table:
  - The final step is to actually create the database with all of its tables based on your schema from the previous step (which was based on your ERD from the first step).
  - The tables are fully defined and usable objects in your database.

- Codey's Construction Company
- Database contains the tables: projects, job_orders, employees, and project_employees
- Customers table has just been added

Example

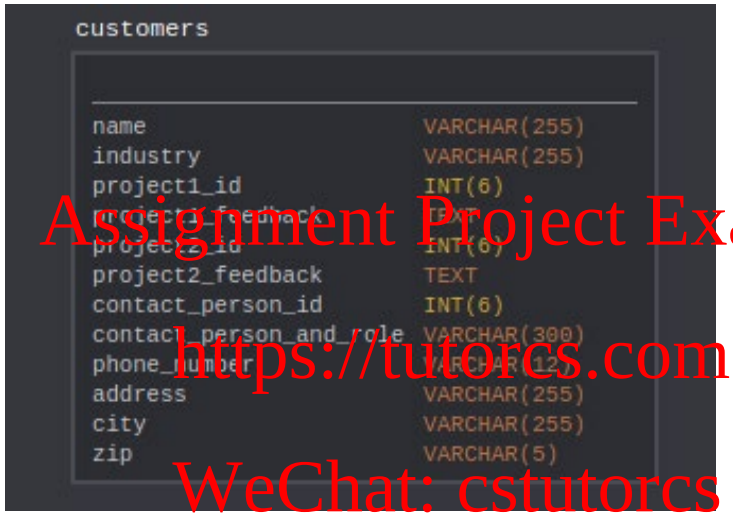
Example adapted from Next Tech Database Fundamentals https://next.tech/



- Objective:
  - Data is stored in tables with rows uniquely identified by a primary key
  - Data within each table is stored in individual columns in its most reduced form
  - There are no repeating groups

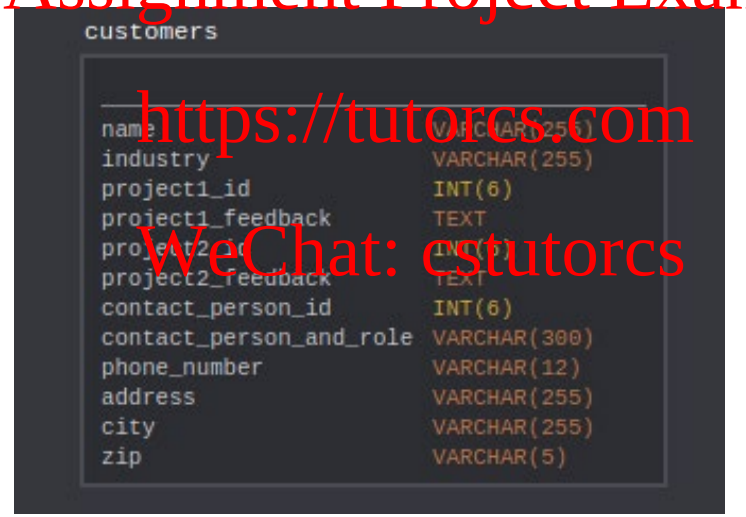
First Normal Form (1 NF)

- Codey's Construction's table customers violates all three rules of 1NF.
  - There is no primary key
    - Need to look up by name which is not guaranteed to be unique



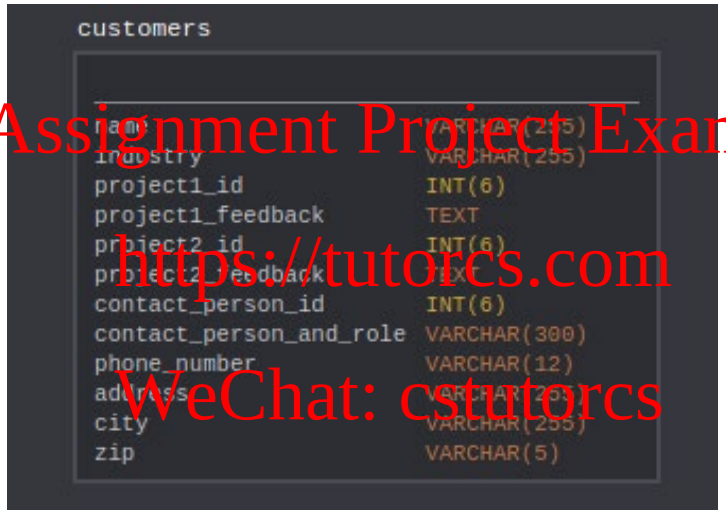
First Normal Form (1 NF)

- Codey's Construction's table customers violates all three rules of 1NF.
  - The data is not in its most reduced form.
    - The column contact_person_and_role can be further divided into two columns, such as contact_person and contact_role.



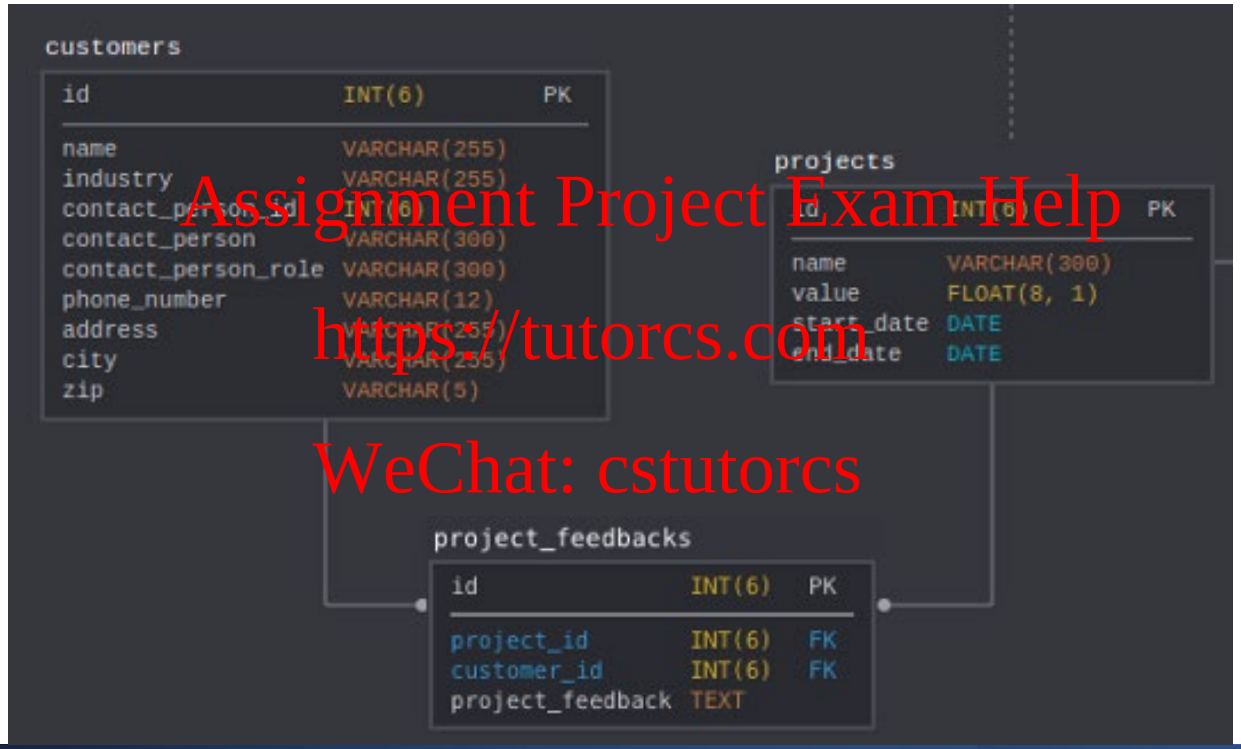
First Normal Form (1 NF)

- Codey's Construction's table customers violates all three rules of 1NF.
  - There are two repeating groups of columns
    - (project1_id, project1_feedback) and (project2_id, project2_feedback).



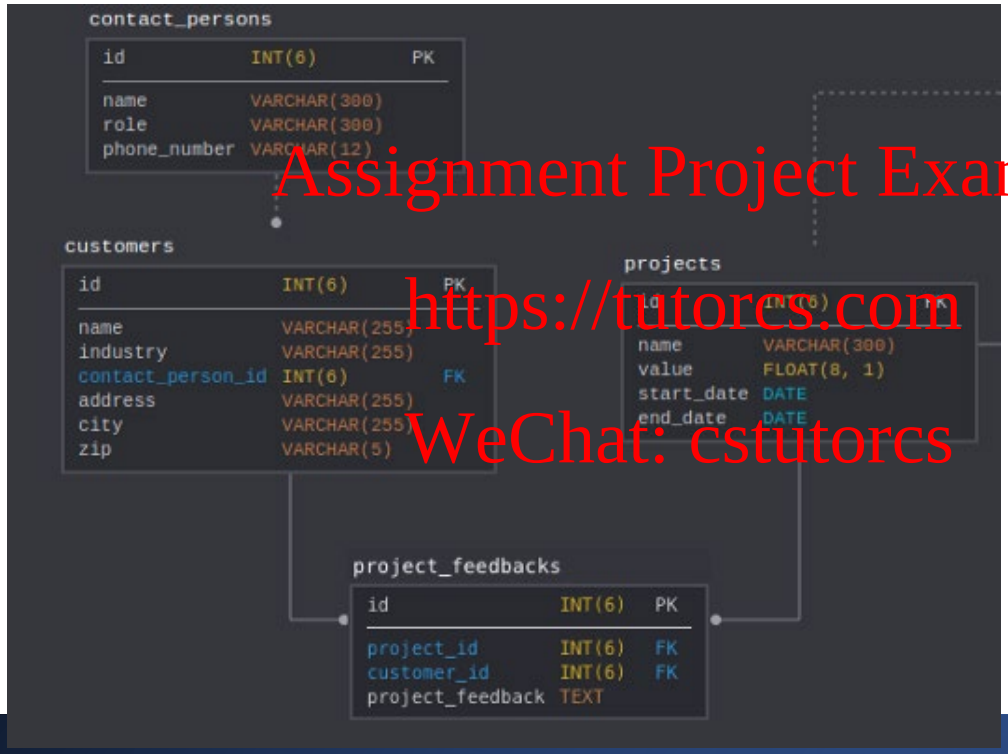
First Normal Form (1 NF)

- To bring to 1NF:
  - We need to add a primary key column called ID
  - We will split contact person and role into two separate columns : contact_person, contact_person_role
  - Move the repeating groups (project id and feedback) to a new table call project_feedbacks (taking account of the project table that already exists)



First Normal Form (1 NF)

- All data (non-prime attributes) in each table must relate directly to the record that the primary key of the table (prime attribute)
  - contact_person, contact_person_role and phone number in this organisation are dependent on the contact_person_id
  - So we should extract these into a new table contact_persons (and give this a primary key to fulfil 1NF)



Second Normal Form (2 NF)

- This is how our customers table looks after 1NF and 2NF:

id	name	industry	contact_id	address	city	zip
000001	Next University	Education	000001	1 Coding Lane	Next	99999
000002	XYZ Health Center	Healthcare	000002	88 Hospital Avenue	Healersville	55555
000003	ArchiTECHS	Architecture	000003	77 Tower Street	Gridlock	12345

- To be in 3NF we need to remove **transitive dependencies:**
  - When one non-prime attribute is dependent on another non-prime attribute

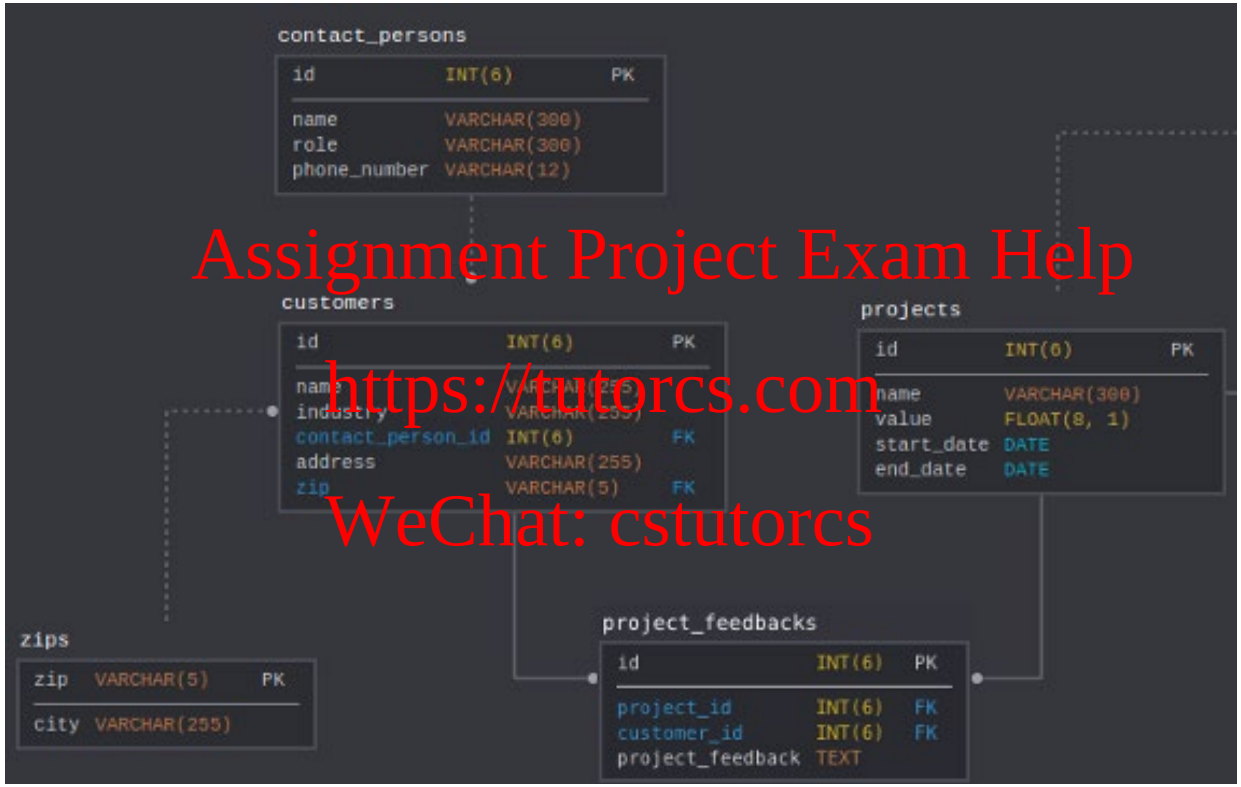
Third Normal Form (3 NF)

id	name	industry	contact_id	address	city	zip
000001	Next University	Education	000001	1 Coding Lane	Next	99999
000002	XYZ Health Center	Healthcare	000002	88 Hospital Avenue	Healersville	55555
000003	ArchiTECHS	Architecture	000003	77 Tower Street	Gridlock	12345

- Look at city and zip
  - City relies on the customer but also on the zip
- We could, potentially, if a customer moves city update one column but not the other.

Third Normal Form (3 NF)

- To overcome this we create a new table (zips) to handle the dependency between city and zip



Third Normal Form (3 NF)

Entity v Relation v Table

- Entity:
  - During the very first step in the design process, when you're creating a conceptual design you have a bunch of entities and relationships that represent the various types of data you'll want to store (and express it as an ERD).
- Relation:
  - Once you've finalized your conceptual design you'll convert that ERD into a logical schema (and express it as an ERD). This schema will be a list of relations (the relations are all your entities and relationships from the previous step).
- Table:
  - The final step is to actually create the database with all of its tables based on your schema from the previous step (which was based on your ERD from the first step).
  - The tables are fully defined and usable objects in your database.

Dependencies

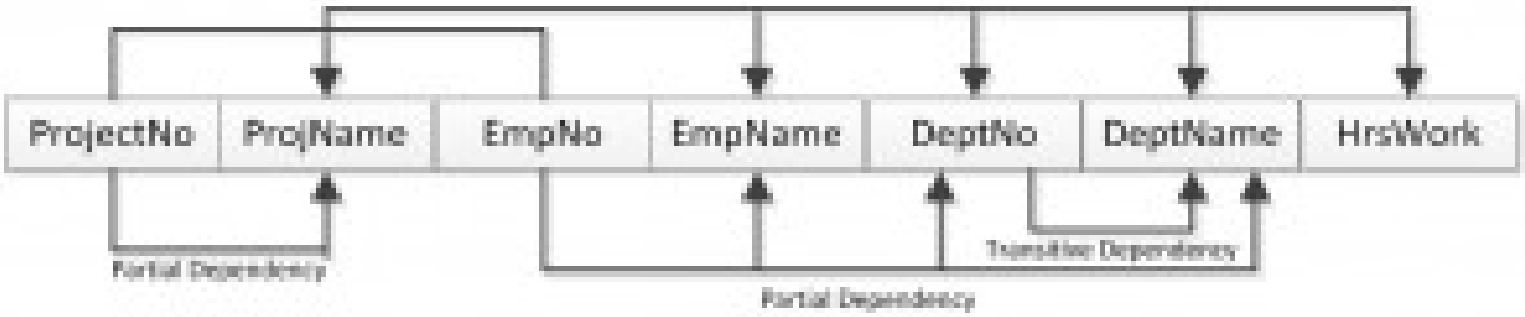
- If you say there is a dependency between attributes in a relation this is the same as saying that there is a functional dependency between those attributes.
- For any relation R, attribute Y is functionally dependent on attribute X (usually the PK), if for every valid instance of X, that value of X uniquely determines the value of Y.
  - E.g. If there is a dependency in a database such that attribute B is dependent upon attribute A, you would write this as:
    - A ->B
    - E.g. Student ID -> Student Name (a student's name can be uniquely determined from their ID)

Dependencies

- Transitive dependencies occur when there is an indirect relationship that causes a functional dependency.
- A transitive dependency exists when you have the following functional dependency pattern:
  - A->B, B->C so A ->C
  - A -> C is a transitive dependency when it is true only because both A -> B and B -> C are true.
- Suppose we have a non-normalised table for a warehouse:
  - For each item in the warehouse we store the item number, number assigned to the distributor of that item (distrib number) and the phone number of that distributor (distrib_phone_number)
  - The only reason that the distributor phone number is functionally dependent on the item number is because the distributor is functionally dependent on the item number and the phone number is functionally dependent on the distributor. The functional dependencies are really:
    - Item_numb –> distrib_numb
    - Distrib_numb –> distrib_phone_number

Dependencies

- Dependency Diagram
  - Suppose we have a non-normalised table (one where dependencies may exist)
  - We can depict it as a diagram
  - The following dependencies are identified in this table:
    - ProjectNo and EmpNo, combined, are the PK.
    - Partial Dependencies:
      - ProjectNo —> ProjName
      - EmpNo —> EmpName, DeptNo,
      - ProjectNo, EmpNo —> HrsWork
    - Transitive Dependency:
      - DeptNo —> DeptName



Domain Key Normal Forms

- Boyce-Codd normal form (BCNF), fourth normal form (4NF), and fifth normal form (5NF) are examples of Domain Key Normal Forms.
- Each form eliminates a possible modification anomaly but doesn't guarantee prevention of all possible modification anomalies.

When a table has more than one candidate key, anomalies may result even though the relation is in 3NF.

Boyce-Codd normal form is a special case of 3NF (3.5NF)

A table is in BCNF if, and only if:

1. It is in the **Third Normal Form**.
2. And, for any functional dependency A → B, A should be a **super key** (set of attributes that uniquely identifies each tuple of a relation)

Boyce-Codd Normal Form (BCNF)

Student_id	Major	Advisor
111	Physics	Smith
111	Music	Chan
320	Math	Dobbs
671	Physics	White
803	Physics	Smith

1.Each Student may major in several subjects.
2.For each Major, a given Student has only one Advisor.
3.Each Major has several Advisors.
4.Each Advisor advises only one Major.
5.Each Advisor advises several Students in one Major.

Boyce-Codd Normal Form (BCNF)

Student_id	Major	Advisor
111	Physics	Smith
111	Music	Chan
320	Math	Dobbs
671	Physics	White
803	Physics	Smith

The functional dependencies for this table are listed below. The first one is a candidate key; the second is not.

1.Student_id, Major ——> Advisor

2.Advisor ——> Major

Boyce-Codd Normal Form (BCNF)

Student_id	Major	Advisor
111	Physics	Smith
111	Music	Chan
320	Math	Dobbs
671	Physics	White
803	Physics	Smith

Anomalies for this table include:

1.Delete – student deletes advisor info

2.Insert – a new advisor needs a student

3.Update – inconsistencies

Note: No single attribute is a candidate key.

PK can be Student_id, Major or Student_id, Advisor.

Boyce-Codd Normal Form (BCNF)

1. **St_Adv** (Student_id, Advisor)
2. **Adv_Maj** (Advisor, Major)

- This is now in BCNF

Boyce-Codd Normal Form (BCNF)

Person	Mobile	Food_Likes
Jim	9893/9424	Burger/Pizza
Jon	9191	Pizza

Person ->-> mobile (->-> multidetermines)

Person ->-> food_likes

Fourth Normal Form

Fourth normal form (4NF) is a level of database normalization where there are no non-trivial multivalued dependencies other than on a candidate key.

Fourth Normal Form

Student_No	Subject_Enrolled	Club_Enrolled
45	Economics	Painting
45	History	Hockey
33	Physics	Drawing
59	Chemistry	Opera
40	Computer Science	Games

The table has 3 columns.
Our dependencies are:

Student_No —> Subject_Enrolled
Student_No —> Club_Enrolled

Check for multivalue dependency:
For every dependency A-> B, if for every value of A multiple values of B exists then the dependency is referred to as multi-valued dependency.

Student no 45 has enrolled in Economics and History in terms of subjects and Painting and Hockey as activities. Thus for a value of Student_No different values Subject_Enrolled and Activity_Enrolled exist.

Fourth Normal Form

Student_No	Subject_Enrolled	Club_Enrolled
45	Economics	Painting
45	History	Hockey
33	Physics	Drawing
59	Chemistry	Opera
40	Computer Science	Games

Subject_Enrolled and Activity_Enrolled are independent of each other.

Solution:

Split into two separate tables (Subject_Enrolled, Activity_Enrolled)

5th Normal Form

A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.

5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy.

5NF is also known as Project-join normal form (PJ/NF).

6th Normal Form

- A relation is in 6NF if it is in 5NF and every join dependency on the relation is trivial.
- Most simple definition:
  - a table is in 6NF when the row contains the Primary Key, and at most one, attribute.

Exercise (Dental Appointments)

staffNo	dentistName	patientNo	patientName	appointment		surgeryNo
				date	time	
S1011	Tony Smith	P100	Gillian White	12-Aug-03	10.00	S10
S1011	Tony Smith	P105	Jill Bell	13-Aug-03	12.00	S15
S1024	Helen Pearson	P108	Ian MacKay	12-Sept-03	10.00	S10
S1024	Helen Pearson	P108	Ian MacKay	14-Sept-03	10.00	S10
S1032	Robin Plevin	P105	Jill Bell	14-Oct-03	16.30	S15
S1032	Robin Plevin	P110	John Walker	15-Oct-03	18.00	S13

- Provide examples of insert, update and delete anomalies
- Transform to 3NF, make clear any assumptions you make about the data.

Normal Forms and Dependencies

- For a table to be in second normal form (2NF), there must be no case of a nonprime attribute in the table that is functionally dependent upon a subset of a candidate key.
- For a table to be in third normal form (3NF), every nonprime attribute must have a nontransitive functional dependency on every candidate key.
- For a table to be in Boyce-Codd Normal Form (BCNF), every functional dependency (other than trivial dependencies) must be on a **superkey**.
  - Superkey = a set of attributes that uniquely identifies each tuple of a relation

Exercises

Normalisation: Advantages

Increases data consistency as it avoids the duplication of data by storing the data in one place only.

Helps in grouping like or related data under the same schema, thereby resulting in the better grouping of data.

Improves searching faster as indexes can be created faster.

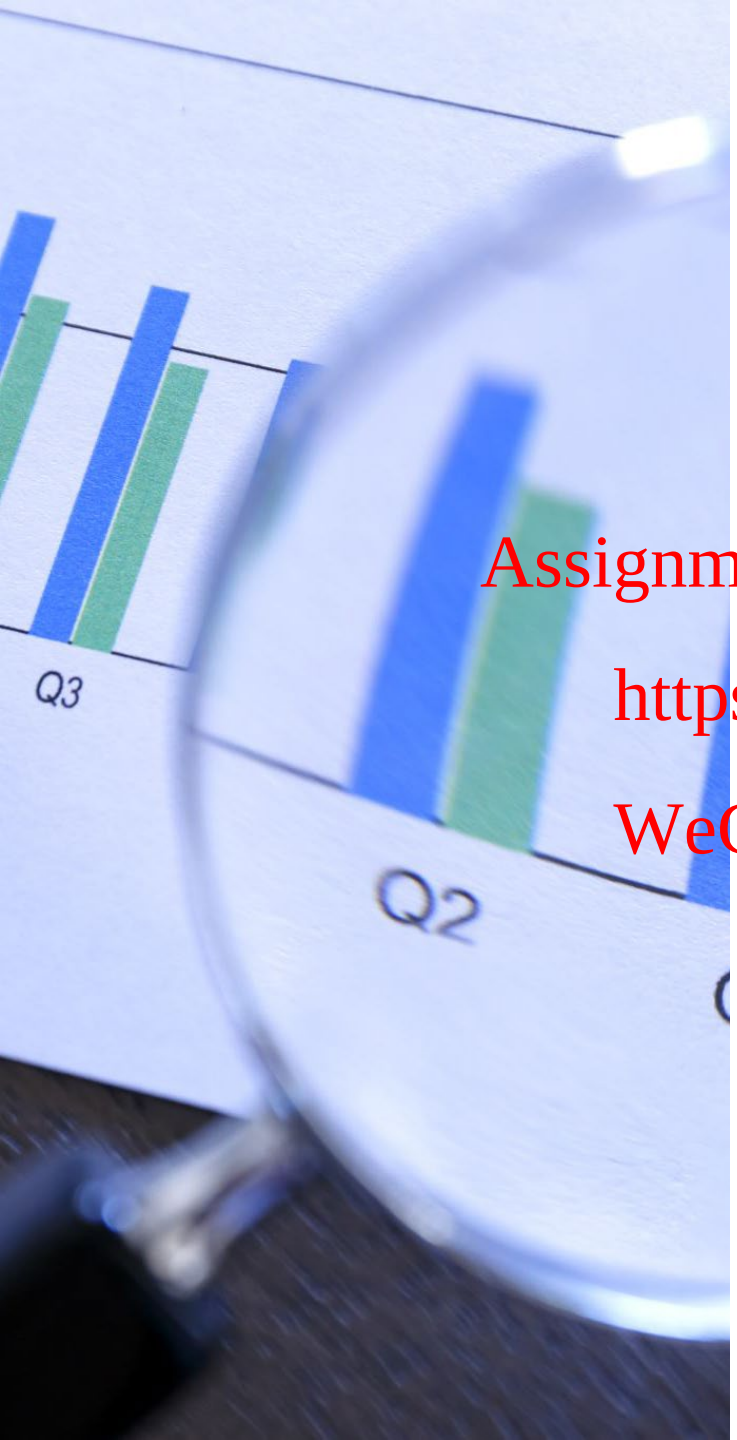
(Very useful for OLTP (online transaction processing)).

Normalisation: Disadvantages

Storing data in one place causes a delay when retrieving

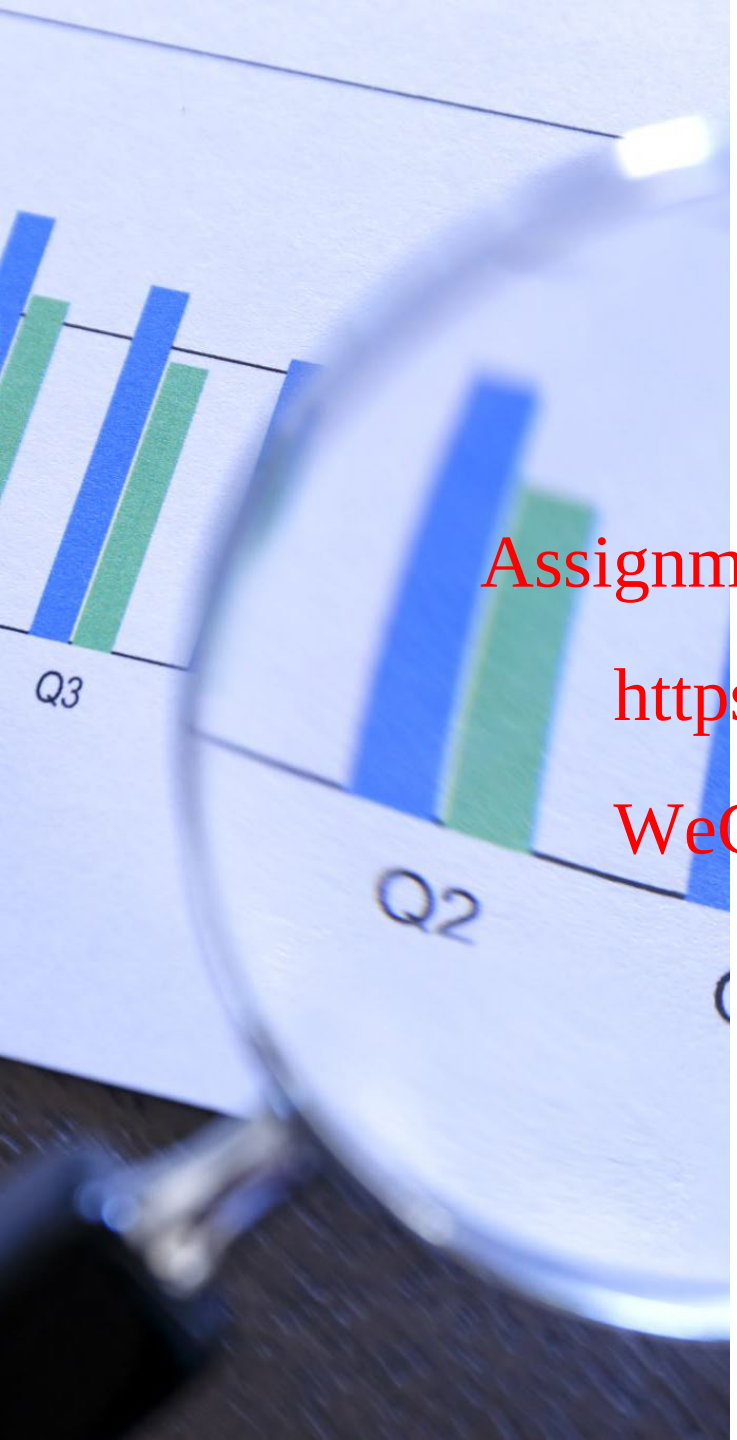
e.g. We cannot find the associated data for, say a product or employee in one place, so we need to join more than one table.

Means that normalization is not a good option in OLAP transactions (online analytical processing).



Denormalization

- Normalized (decomposed) tables require additional processing, reducing system speed.
- Normalization purity is often difficult to sustain in the modern database environment.
- The conflict between design efficiency, information requirements, and processing speed are often resolved through compromises that include **Denormalization:**
  - the process of improving the read performance of a database, at the expense of losing some write performance, by adding redundant copies of data or by grouping it.



Denormalization

- How?
  - Adding precomputed redundant data to a database to improve the read performance
- You cannot denormalise unless you have normalised.
- Can be done as part of design or delegated to the DBMS (handled by the DBA)