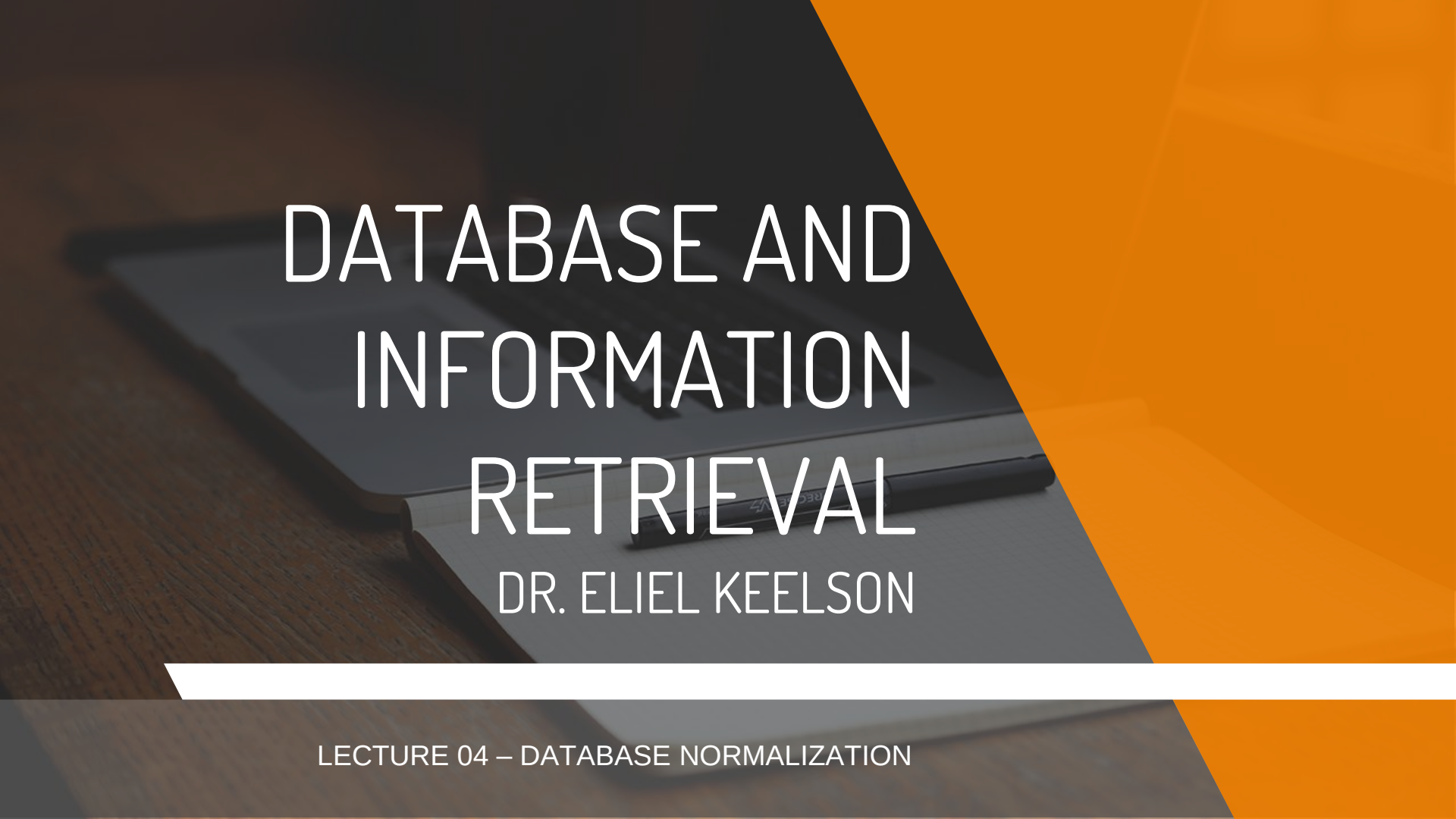

DATABASE AND INFORMATION RETRIEVAL

DR. ELIEL KEELSON

LECTURE 04 – DATABASE NORMALIZATION

Database Normalization

- Database Normalization is a technique of organizing the data in the database.

- Normalization is a systematic approach of decomposing tables to eliminate data redundancy and undesirable characteristics like Insertion, Update and Deletion Anomalies.

- It is a multi-step process that puts data into tabular form by removing duplicated data from the relation tables.

Database Normalization

- Normalization is used for mainly two purposes
  - Eliminating redundant (useless) data.
  - Ensuring data dependencies make sense i.e data is logically stored.

Problems That Occur Without Normalization

- Without Normalization, it becomes difficult to handle and update the database, without facing data loss.

- Insertion, Updating and Deletion Anomalies are very frequent if Database is not Normalized.

- To understand these anomalies let us take an example of **Student** table.

Problems That Occur Without Normalization

S_id	S_Name	S_Address	Subject_opted
401	Adam	Noida	Bio
402	Alex	Panipat	Maths
403	Stuart	Jammu	Maths
404	Adam	Noida	Physics

Problems That Occur Without Normalization

- **Updating Anomaly :** To update address of a student who occurs twice or more than twice in a table, we will have to update **S_Address** column in all the rows, else data will become inconsistent.

- **Insertion Anomaly :** Suppose for a new admission, we have a Student id(S_id), name and address of a student but if student has not opted for any subjects yet then we have to insert **NULL** there, leading to Insertion Anomaly.

- **Deletion Anomaly :** If (S_id) 401 has only one subject and temporarily he drops it, when we delete that row, entire student record will be deleted along with it.

GUIDELINES FOR NORMALIZATION

Guidelines For Normalization

- There are basically three steps of normalization.
- They are First Normal Form (1NF), Second Normal Form (2NF) and Third Normal Form (3NF).
- Each form has its own set of rules/guidelines.
- After a database conforms to a level, it is considered normalized to that form.

Guidelines For Normalization

- There are other levels of normalization that would not been discussed. They are
  - Boyce-Codd Normal Form (BCNF)
  - Fourth Normal Form (4NF)
  - Fifth Normal Form (5NF)
  - Strong Join-Protection Normal Form
  - Over-Strong Join-Protection Normal Form
  - Domain Key Normal Form (DK/NF).

- These forms of normalization may take things further than they need to go. They exist to make a database truly relational. They mostly deal with multiple dependencies and relational keys.

Guidelines For Normalization – 1NF

- As per First Normal Form, no two Rows of data must contain repeating group of information i.e each set of column must have a unique value, such that multiple columns cannot be used to fetch the same row.

- Each table should be organized into rows, and each row should have a primary key that distinguishes it as unique.

Guidelines For Normalization – 1NF

- For example consider a table which is not in First normal form

Student Table :

Student	Age	Subject
Adam	15	Biology, Maths
Alex	14	Maths
Stuart	17	Maths

Guidelines For Normalization – 1NF

- In First Normal Form, any row must not have a column in which more than one value is saved, like separated with commas.

- Rather than that, we must separate such data into multiple rows.

Student Table following 1NF will be :

Student	Age	Subject
Adam	15	Biology
Adam	15	Maths
Alex	14	Maths
Stuart	17	Maths

Guidelines For Normalization – 1NF

- Using the First Normal Form, data redundancy increases, as there will be many columns with same data in multiple rows but each row as a whole will be unique.

Guidelines For Normalization – 2NF

- As per the Second Normal Form there must not be any partial dependency of any column on primary key.

- It means that for a table that has concatenated primary key, each column in the table that is not part of the primary key must depend upon the entire concatenated key for its existence.

- If any column depends only on one part of the concatenated key, then the table fails **Second normal form**.

Guidelines For Normalization – 2NF

- In example of First Normal Form there are two rows for Adam, to include multiple subjects that he has opted for.

- While this is searchable, and follows First normal form, it is an inefficient use of space.

- Also in the above Table in First Normal Form, while the candidate key is {**Student**, **Subject**}, **Age** of Student only depends on Student column, which is incorrect as per Second Normal Form.

Guidelines For Normalization – 2NF

- To achieve second normal form, it would be helpful to split out the subjects into an independent table, and match them up using the student names as foreign keys.

New Student Table following 2NF will be :

Student	Age
Adam	15
Alex	14
Stuart	17

Guidelines For Normalization – 2NF

- In Student Table the candidate key will be **Student** column, because all other column i.e **Age** is dependent on it.

New Subject Table introduced for 2NF will be :

Student	Subject
Adam	Biology
Adam	Maths
Alex	Maths
Stuart	Maths

Guidelines For Normalization – 2NF

- In Subject Table the candidate key will be {**Student**, **Subject**} column.

- Now, both the above tables qualifies for Second Normal Form and will never suffer from Update Anomalies.

- Although there are a few complex cases in which table in Second Normal Form suffers Update Anomalies, and to handle those scenarios Third Normal Form is there.

Guidelines For Normalization – 3NF

- **Third Normal form** applies that every non-prime attribute of table must be dependent on primary key, or we can say that, there should not be the case that a non-prime attribute is determined by another non-prime attribute.

- So this *transitive functional dependency* should be removed from the table and also the table must be in **Second Normal form**.

Guidelines For Normalization – 3NF

- For example, consider a table with following fields.

Student_Detail Table :

Student_id	Student_name	DOB	Street	city	State	Zip

- In this table Student_id is Primary key, but street, city and state depends upon Zip.

- The dependency between zip and other fields is called **transitive dependency**.

- Hence to apply **3NF**, we need to move the street, city and state to new table, with **Zip** as primary key.

Guidelines For Normalization – 3NF

New Student_Detail Table :

Student_id	Student_name	DOB	Zip

Address Table :

Zip	Street	city	state

Guidelines For Normalization – 3NF

- The advantage of removing transitive dependency is,
  - Amount of data duplication is reduced.
  - Data integrity achieved.

THANKS!

Any questions?

You can find me at elielkeelson@gmail.com &
ekeelson@knust.edu.gh