ADVANCED DB

Week4 presentation by A.M.Esfar-E-Alam

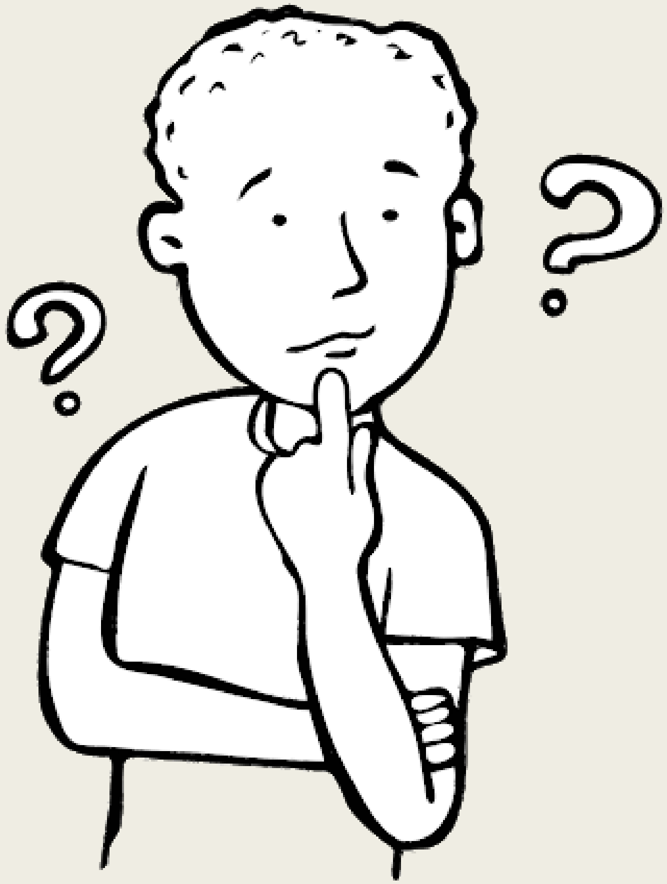
Topics

Normal forms in Database

Atomic and normal form relation

Introducing Complex Data Type

Normalization



- **NORMALIZATION** is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies.
- Normalization rules divides larger tables into smaller tables and links them using relationships.
- The purpose of Normalization in SQL is to eliminate redundant (repetitive) data and ensure data is stored logically.

BRAC UNIVERSITY
Inspiring Excellence

Database Normal Forms

- Here is a list of Normal Forms
- 1NF (First Normal Form)
- 2NF (Second Normal Form)
- 3NF (Third Normal Form)
- BCNF (Boyce-Codd Normal Form)
- There are 5th 6th normal forms but these exists in theory mostly

DB normalization Example

- Database **NORMALIZATION EXAMPLE** can be easily understood with the help of a case study. Assume, a video library maintains a database of movies rented out. Without any **normalization**, all information is stored in one table as shown.
- Here you see **Movies Rented column has multiple values.** Now let's move into 1st Normal Forms:

FULL NAMES	PHYSICAL ADDRESS	MOVIES RENTED	SALUTATION
Janet Jones	First Street Plot No 4	Pirates of the Caribbean, Clash of the Titans	Ms.
Robert Phil	3rd Street 34	Forgetting Sarah Marshal, Daddy's Little Girls	Mr.
Robert Phil	5th Avenue	Clash of the Titans	Mr.

Table 1

BRAC UNIVERSITY
Inspiring Excellence

1st Normal Form

- **1NF (First Normal Form) Rules**
- Each table cell should contain a single value.
- Each record needs to be unique.
- The above table in 1NF-

FULL NAMES	PHYSICAL ADDRESS	MOVIES RENTED	SALUTATION
Janet Jones	First Street Plot No 4	Pirates of the Caribbean	Ms.
Janet Jones	First Street Plot No 4	Clash of the Titans	Ms.
Robert Phil	3rd Street 34	Forgetting Sarah Marshal	Mr.
Robert Phil	3rd Street 34	Daddy's Little Girls	Mr.
Robert Phil	5th Avenue	Clash of the Titans	Mr.

BRAC UNIVERSITY
Inspiring Excellence

2^{nd} NF

- Rule 1- Be in 1NF
- Rule 2- Single Column Primary Key
- It is clear that we can't move forward to make our simple database in 2^{nd} Normalization form unless we partition the table above.

MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION
1	Janet Jones	First Street Plot No 4	Ms.
2	Robert Phil	3^{rd} Street 34	Mr.
3	Robert Phil	5^{th} Avenue	Mr.

MEMBERSHIP ID	MOVIES RENTED
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans

Table 2

BRAC UNIVERSITY
Inspiring Excellence

What are transitive functional dependencies?

- A transitive functional dependency is when changing a non-key column, might cause any of the other non-key columns to change

MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION
1	Janet Jones	First Street Plot No 4	Ms.
2	Robert Phil	3rd Street 34	Mr.
3	Robert Phil	5th Avenue	Mr.

Change in Name

May Change Salutation

BRAC UNIVERSITY
Inspiring Excellence

3NF (Third Normal Form) Rules

- Rule 1- Be in 2NF
  Rule 2- Has no transitive functional dependencies
- We have again divided our tables and created a new table which stores Salutations.
- There are no transitive functional dependencies, and hence our table is in 3NF
- In Table 3 Salutation ID is primary key, and in Table 1 Salutation ID is foreign to primary key in Table 3
- **BCNF (Boyce-Codd Normal Form)**
- Even when a database is in 3rd Normal Form, still there would be anomalies resulted if it has more than one **Candidate** Key.
- Sometimes is BCNF is also referred as **3.5 Normal Form.**
-

Membership ID	Full Names	Physical Address	Salutation ID
1	Janet Jones	First Street Plot No 4	2
2	Robert Phil	3rd Street 34	1
3	Robert Phil	5th Avenue	1

Membership ID	Movies Rented
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans

Salutation ID	Salutation
1	Mr.
2	Ms.
3	Mrs.
4	Dr.

BRAC UNIVERSITY
Inspiring Excellence

Takeaway

Database designing is critical to the successful implementation of a database management system that meets the data requirements of an enterprise system.

Normalization in DBMS helps produce database systems that are cost-effective and have better security models.

Functional dependencies are a very important component of the normalize data process

Most database systems are normalized database up to the third normal forms.

A primary key uniquely identifies are record in a Table and cannot be null

A foreign key helps connect table and references a primary key

What is Atomic Relation in First Normal Form

- Atomic means data which cannot be divided further.
- Rule of atomicity:
- rule 1: a column with atomic data can't have several values of the same type of data in the same column.
- rule2: a table with atomic data can't have several columns with the same datatype.
- Like fullname column can't say that it could be atomic because it can be further divded into lastname, firstname. A column with interest could also be divided further, so a column which can't be divided is known as atomic.
- In 1st normal form we try to make our dat atomic

BRAC UNIVERSITY
Inspiring Excellence

Complex Data Type

Why we need it?

- Permit non-atomic domains .
- Example of non-atomic domain: set of integers,or set of tuples (arrays,map etc) as datatype
- Allows more intuitive modeling for applications with complex data
- Traditional database applications in data processing had
- conceptually simple data types
  Relatively few data types, first normal form holds
- Complex data types have grown more important in recent years.E.g. Addresses(flat/ road/ area) can be viewed as a
- Single string, or
  Separate attributes for each part, or
  Composite attributes (which are not in first normal form)
- E.g. it is often convenient to store multivalued attributes as-is, without creating a separate relation to store the values in first normal form

Applications

- computer-aided design, computer-aided software engineering
- multimedia and image databases, and document/hypertext databases.

Defining Complex data type

- Intuitive definition:
  - *It is a datatype that allows relations(table) to violate 1st normal form*
  - *That means we don't need the values to be atomic, we can have array map tuple etc through that we can have relation inside a relation*
  - *Retains mathematical foundation of relational model*
  - *Violates first normal form.*
- Any data that does not fall into the traditional field structure (alpha, numeric, dates) of a relational DBMS. Examples of complex data types are bills of materials, word processing documents, maps, time-series, images and video.
- In a relational DBMS, complex data types are stored in a LOB, but either the client application or some middleware is required to process the data.
- In an object DBMS or an object-relational DBMS, complex data types are stored as objects that are integrated into and activated by the DBMS.

Nested Relation

- Example of a Nested Relation Example:
  - *library information system*
  - *Each book has title,*
  - *a set of authors,*
  - *Publisher, and*
  - *a set of keywords*
- See a figure of a Non-1NF relation ‘books ’

title	*author-set*	*publisher*	*keyword-set*
		(name, branch)	
Compilers	{Smith, Jones}	(McGraw-Hill, New York)	{parsing, analysis}
Networks	{Jones, Frick}	(Oxford, London)	{Internet, Web}

- Now lets Remove awkwardness of books table by assuming that the following multivalued dependencies hold:
- title author
- title keyword
- title pub-name, pub-branch
- This is how we can Decompose it into 4NF using the schemas: (title, author) (title, keyword) (title, pub-name, pub-branch)

title	*author*
Compilers	Smith
Compilers	Jones
Networks	Jones
Networks	Frick

authors

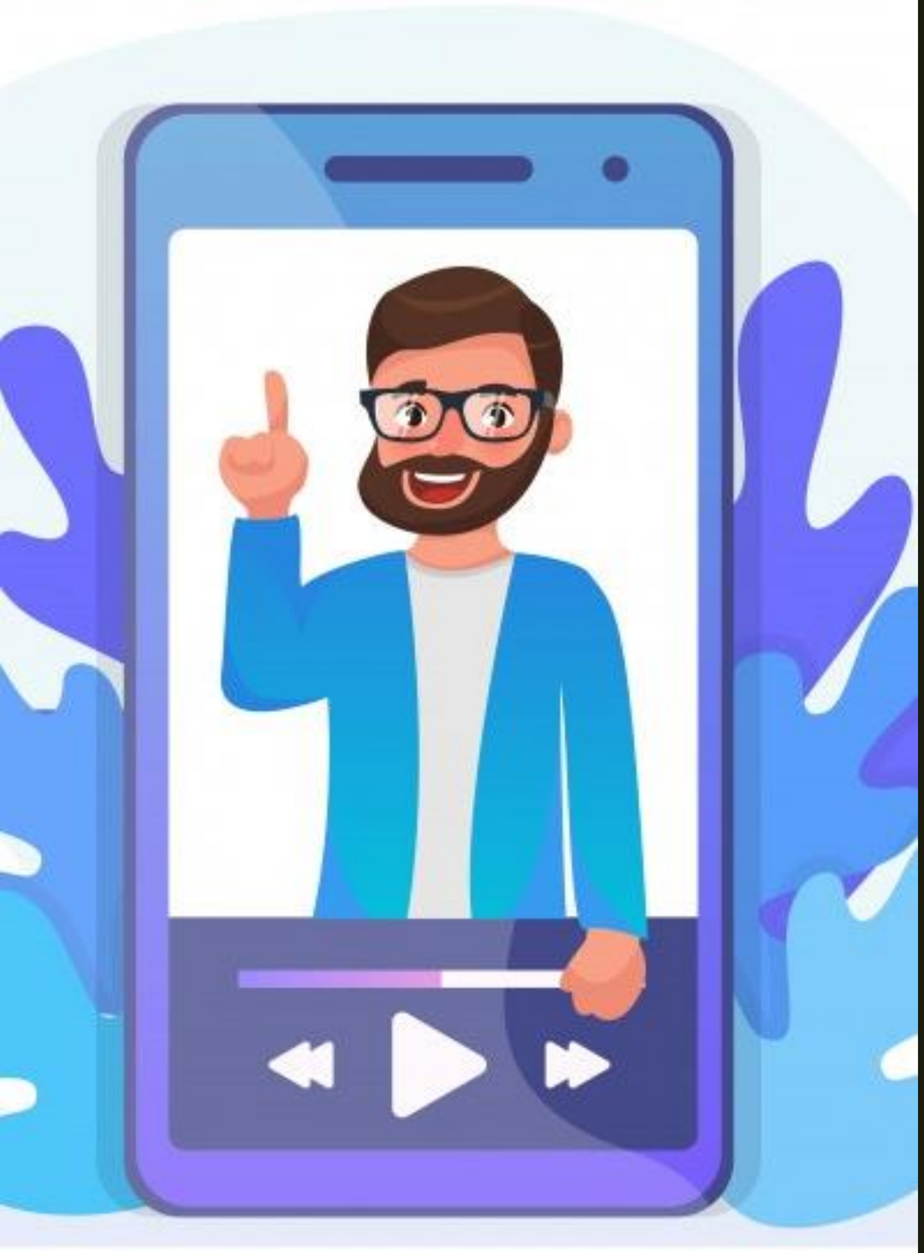
title	*keyword*
Compilers	parsing
Compilers	analysis
Networks	Internet
Networks	Web

keywords

title	*pub-name*	*pub-branch*
Compilers	McGraw-Hill	New York
Networks	Oxford	London

books4

- As you can see Problems with 4NF Schema 4NF design requires users to include joins in their queries.(if we want to query a book or doc we will need to write a join query for these three tables)
- So if we can have a 1NF relational view flat-books defined by join of 4NF relations:
- eliminates the need for users to perform joins, but loses the one-to-one correspondence between tuples and documents.
- And has a large amount of redundancy
- So a Nested relations representation is much more natural here.
- That means going for more and more normalization is not always good and it goes against our intuitive design



- To resolve it SQL extended its support for data type by introducing complex data type.
- Introducing collection data type like map or array can solve it
- alternatively if we can use objects which can include multiple data types inside it will also solve our issue.
- So these were introduced:
- Collection and large object types
  - ☐ *Nested relations are an example of collection types*
- Structured types
  - ☐ *Nested record structures like composite attributes*
- Inheritance
- Object orientation
  - ☐ *Including object identifiers and references*

Structured Types and Inheritance in SQL

- Lets see how we can implement these
- One of these ways are implementing structure type and inheritance
- We know what a structure is if we know C/C++. We also had that in java
- Structured types can be declared and used in SQL
- Here’s an example

- create type Name as

  (firstname varchar(20),

  lastname varchar(20))

  final

- create type Address as

  (street varchar(20),

  city varchar(20),

  zipcode varchar(20))

  not final

- Note: final and not final indicate whether subtypes can be created
  Structured types can be used to create tables with composite attributes

- create table customer

(name Name, address Address, dateOfBirth date)

- Dot notation used to reference components: name.firstname

- As you can see we have structures with multiple data inside it and then we combined that in customer table

- Now we can have our own type like name or address and we can use them to create or define a table
- User-defined types
- create type *CustomerType as*

    (name Name,

    address Address,

    dateOfBirth date) not final

- Can then create a table whose rows are a user-defined type

create table customer of CustomerType

Methods

- We can add a method declaration with a structured type.

  method ageOnDate (onDate date)

  returns interval year

- Method body is given separately.

  create instance method ageOnDate (onDate date)

  returns interval year

  for CustomerType

  begin

  return onDate - self.dateOfBirth;

  end

- We can now find the age of each customer:

  select name.lastname, ageOnDate (current_date) from customer

Type Inheritance

- Suppose that we have the following type definition for people:

create type Person (name varchar(20), address varchar(20))

- Using inheritance to define the student and teacher types

create type Student under Person (degree varchar(20), department varchar(20))

create type Teacher under Person (salary integer, department varchar(20))

- Subtypes can redefine methods by using overriding method in place of method in the method declaration

BRAC UNIVERSITY

Inspiring Excellence

Type Inheritance in SQL

- It does not support multiple inheritance
- As in most other languages, a value of a structured type must have exactly one most-specific type
- Example: an entity has the type Person as well as Student. I
- The most specific type of the entity is Student
- We can also have Table Inheritance Subtables in SQL

Array and Multiset Types in Oracle NoSQL

- An instance of a complex data type contains multiple values and provides access to its nested values. Currently, Oracle NoSQL Database supports the following kinds of complex value

Data Type	Example
ARRAY (T)	Type: ARRAY (INTEGER) Type Instance: [600004,560076,01803]
MAP (T)	Type: MAP(INTEGER) Type Instance: { "Chennai":600004, "Bangalore":560076, "Boston":01803 }
RECORD (k1 T1 n1, k2 T2 n2, ……., kn Tn nn)	Type: RECORD(country STRING, zipcode INTEGER, state STRING, street STRING) Type Instance: { "country":"US", "zipcode":600004, "state":"Arizona", "street":"4th Block" }

BRAC UNIVERSITY
Inspiring Excellence

Next Topics

- Querying collection and unesting
- Object Identity
- Class reference