NORMALIZATION

Introduction

- A large DB defined as a single relation may result in data duplication. This repetition of data may result in:
  - Large relations, increase search and maintenance time
  - Poor utilization of disk space and resources.
  - The likelihood of errors and inconsistencies increases.
- A solution: analyze and decompose the relations (with redundant data) into smaller, simpler, and well-structured relations that satisfy the desirable properties.
- Normalization is a process of decomposing the relations into relations with fewer attributes.

What is Normalization?

- A process of organizing data in the database.
- It divides the larger table into small tables and links them using relationships.
- It is used to minimize the redundancy from a relation or set of relations.
- It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
  - normalizing the relations will remove these anomalies.
  - Failure to which leads to data redundancy and can cause data integrity and other problems as the database grows.

Insertion anomaly

- When one cannot insert a new tuple into a relationship due to lack of data.
- Ex:- If we try to insert a record in STUDENT_COURSE with STUD_NO =7, it will not allow.

STUDENT

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY	STUD_AGE
1	RAM	9716271721	Haryana	India	20
2	RAM	9898291281	Punjab	India	19
3	SUJIT	7898291981	Rajsthan	India	18
4	SURESH		Punjab	India	21

Table 1

STUDENT_COURSE

STUD_NO	COURSE_NO	COURSE_NAME
1	C1	DBMS
2	C2	Computer Networks
1	C2	Computer Networks

Table 2

Deletion anomaly

- Situation where the deletion of data results in the unintended loss of some other important data.
- Ex:- If we try to delete a record from STUDENT with STUD_NO =1, it will not allow.

STUDENT

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY	STUD_AGE
1	RAM	9716271721	Haryana	India	20
2	RAM	9898291281	Punjab	India	19
3	SUJIT	7898291981	Rajsthan	India	18
4	SURESH		Punjab	India	21

Table 1

STUDENT_COURSE

STUD_NO	COURSE_NO	COURSE_NAME
1	C1	DBMS
2	C2	Computer Networks
1	C2	Computer Networks

Table 2

Update anomaly

- When an update of a single data value requires multiple rows of data to be updated.
- Ex:- If the Course_Name (Table 2) for the Computer Networks changed to “Intro to CN” for STUD_No 2, it would need to be changed in two different records.

STUDENT

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY	STUD_AGE
1	RAM	9716271721	Haryana	India	20
2	RAM	9898291281	Punjab	India	19
3	SUJIT	7898291981	Rajsthan	India	18
4	SURESH		Punjab	India	21

Table 1

STUDENT_COURSE

STUD_NO	COURSE_NO	COURSE_NAME
1	C1	DBMS
2	C2	Computer Networks
1	C2	Computer Networks

Table 2

Normal Forms

Normalization works through a series of stages called Normal forms. The normal forms apply to individual relations. The relation is said to be in particular normal form if it satisfies their constraints.

Normal Form	Description
1NF	A relation is in 1NF if it contains an atomic value.
2NF	A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.
3NF	A relation will be in 3NF if it is in 2NF and no transition dependency exists.
BCNF	A stronger definition of 3NF is known as Boyce Codd's normal form.
4NF	A relation will be in 4NF if it is in Boyce Codd's normal form and has no multi-valued dependency.
5NF	A relation is in 5NF If it is in 4NF and does not contain any join dependency, joining should be lossless.

Functional dependencies (FDs)

- are used to specify *formal measures* of the "goodness" of relational designs
- FDs and keys are used to define **normal forms** for relations
- FDs are **constraints** that are derived from the *interrelationships* of the data attributes

Functional dependencies (FDs)

- FD is a relationship that exists between two attributes. It typically exists between the primary key and non-key attribute within a table.

  $X \rightarrow Y$ (determinant → dependent)

- **Ex:** Assume we have an employee table with attributes: Emp_Id, Emp_Name, Emp_Address.
  - Here Emp_Id attribute can uniquely identify the Emp_Name attribute because knowing Emp_Id, we can tell that employee name associated with it.
  - Functional dependency can be written as:

    $Emp\_Id \rightarrow Emp\_Name$

    We say Emp_Name is functionally dependent on Emp_Id.

Memories

- **Normalization**: Process of decomposing unsatisfactory "bad" relations by breaking up their attributes into smaller relations

- **Normal form**: Condition using keys and FDs of a relation to certify whether a relation schema is in a particular normal form
  - 2NF, 3NF, BCNF based on keys and FDs of a relation schema
  - 4NF based on keys, multi-valued dependencies
  - 5NF based on join dependency

Implication

- What for the attributes that can have multiple values for the same attribute?
  - Phone numbers, email ids, borrowed Books, …?

First Normal From (1NF)

- A table/relation is in the 1NF iff
  - It contains only atomic values at each row and column.

- Example:

User ID	Borrowed books
C45	B33, B44, B55
C12	B56

User ID	Borrowed book
C45	B33
C45	B44
C45	B33
C12	B56

Example

- For an airline

Flight	Weekdays
UA59	Mo We Fr
UA73	Mo Tu We Th Fr

Flight	Weekday
UA59	Mo
UA59	We
UA59	Fr
UA73	Mo
UA73	We
…	…

Functional dependency

Let X and Y be *sets* of attributes in a table *T*

- Y is **functionally dependent** on *X* **iff** for each set x ∈ X there is precisely one corresponding set y∈ Y

- Y is **fully functional dependent** on *X* if
  - *Y* is functional dependent on X and
  - Y is not functional dependent on any proper subset of *X*

Example

- Book table

BookNo	Title	Author	Year
B1	C Programming	Alex	1871
B2	Intro. to Python	Bob	1984

Author attribute is:

- ***functionally dependent*** on the pair { BookNo, Title}
- ***fully functionally dependent*** on BookNo

Why it matters

- table BorrowedBooks

BookNo	User	Address	Due
B1	Alex	101 Main Street	3/5/23
B2	Bob	202 Market Street	2/4/23

Address attribute is

- ***functionally dependent*** on the pair { BookNo, User}
- ***fully functionally dependent*** on User (not on the Key BookNo)

Problems

- Cannot insert new users in the system until they have borrowed books (BookNo)
  - ***Insertion anomaly***
- Must update all rows involving a given user if he or she moves. (rows of all books borrowed)
  - ***Update anomaly***
- Will lose information about users that have returned all the books they have borrowed (if delete a book record)
  - ***Deletion anomaly***

Armstrong inference rules (1974)

- ***Axioms:***
  - Reflexivity: if $Y \subseteq X$, then $X \rightarrow Y$
  - Augmentation: if $X \rightarrow Y$, then $WX \rightarrow WY$
  - Transitivity: if $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$
- ***Derived Rules:***
  - Union: if $X \rightarrow Y$ and $X \rightarrow Z$, the $X \rightarrow YZ$
  - Decomposition: if $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$
  - Pseudotransitivity: if $X \rightarrow Y$ and $WY \rightarrow Z$, then $XW \rightarrow Z$

Armstrong inference rules (1974)

- Three last rules can be derived from the first three (the axioms)
- Let us look at the ***union rule***:
  if $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
- Using the first three axioms, we have:
  - if $X \rightarrow Y$, then $XX \rightarrow XY$ same as $X \rightarrow XY$ (2^{nd})
  - if $X \rightarrow Z$, then $YX \rightarrow YZ$ same as $XY \rightarrow YZ$ (2^{nd})
  - if $X \rightarrow XY$ and $XY \rightarrow YZ$, then $X \rightarrow YZ$ (3^{rd})

Second Normal Form

- A table is in 2NF iff
  - It is in 1NF and
  - No partial, only full dependency

  no non-prime attribute is dependent on any proper subset of any candidate key of the table (no partial, only full dependency, AB->C but not A->C or B->C, ...)

- A ***non-prime attribute*** of a table is an attribute that is not a part of any candidate key of the table
- A ***candidate key*** is a minimal superkey

Example

- Library allows patrons to request books that are currently out
- Book request Table

BookNo	Patron	PhoneNo
B3	J. Fisher	555-1234
B2	J. Fisher	555-1234
B2	M. Amer	555-4321

Example

- Candidate key is {BookNo, Patron}
- We have
  - Patron → PhoneNo
- Table is not 2NF
  - Potential for
    - Insertion anomalies
    - Update anomalies
    - Deletion anomalies

2NF Solution

- Put telephone number in separate Patron table

BookNo	Patron
B3	J. Fisher
B2	J. Fisher
B2	M. Amer

Patron	PhoneNo
J. Fisher	555-1234
M. Amer	555-4321

Third Normal Form

- A table is in 3NF iff
  - it is in 2NF and
  - No transitive dependency

  all its attributes are determined only by its candidate keys and not by any non-prime attributes

  Note: For each dependency

  LHS is candidate or super key OR

  RHS is prime attribute

Example

- Table BorrowedBooks

BookNo	Patron	Address	Due
B1	J. Fisher	101 Main Street	3/2/15
B2	L. Perez	202 Market Street	2/28/15

- Candidate key is BookNo
- Patron → Address

3NF Solution

- Put address in separate Patron table

BookNo	Patron	Due
B1	J. Fisher	3/2/15
B2	L. Perez	2/28/15

Patron	Address
J. Fisher	101 Main Street
L. Perez	202 Market Street

Another example

- Tournament winners

Tournament	Year	Winner	DOB
Indiana Invitational	1998	Al Fredrickson	21 July 1975
Cleveland Open	1999	Bob Albertson	28 Sept. 1968
Des Moines Masters	1999	Al Fredrickson	21 July 1975

- Candidate key is {Tournament, Year}
- Winner →DOB

Boyce-Codd Normal Form

- Stronger form of 3NF
- A table *T* is in BCNF iff
  - LHS must be Candidate key or Super Key

  for every one of its non-trivial dependencies $X \rightarrow Y$, X is a super key for T

- Trivial dependency: $X \rightarrow Y$, Y is a subset of X
- Most tables that are in 3NF also are in BCNF

Example

Manager	Project	Branch
Alice	Alpha	Austin
Alice	Delta	Austin
Carol	Alpha	Houston
Dean	Delta	Houston

- We can assume
  - Manager → Branch
  - {Project, Branch} → Manager

Example

Manager	Project	Branch
Alice	Alpha	Austin
Bob	Delta	Houston
Carol	Alpha	Houston
Alice	Delta	Austin

- Not in BCNF because Manager → Branch and Manager is not a superkey
- Will decomposition work?

A decomposition

Manager	Project
Alice	Alpha
Bob	Delta
Carol	Alpha
Alice	Delta

Manager	Branch
Alice	Austin
Bob	Houston
Carol	Houston

- Two-table solution does not preserve the dependency {Project, Branch} → Manager

3NF and BCNF Example

- R(ABC) F:{AB->C, C->A}

Steps

- 1. Find Keys
- 2. Check FDs for BCNF

- AB->C BCNF
- C->A Not in BCNF but A is prime attribute (in 3NF)

Another Example

- R(ABCD) F:{AB->CD, D->A}

Steps

1. Find Keys AB+=ABCD, CB+=CB, DB+=DBAC
2. Check FDs for 3NF

- AB->CD in 3NF
- D->A in 3NF (Not in BCNF) - A is prime attribute

- Table decomposition ABCD into DA and BCD (CK-BD) – FDs not preserved

Multivalued dependencies

- Assume the column headings in a table are divided into three disjoint groupings *X*, *Y*, and *Z*

- For a particular row, we can refer to the data beneath each group of headings as *x*, *y*, and *z* respectively

Multivalued dependencies

- A ***multivalued dependency*** X =>Y occurs if
  - For any x_c occurring in the table and the list of all the $x_c$yz combinations that occur in the table,

    we will find that x_c is associated with the same y entries regardless of z.

- A **t*rivial multivalued dependency*** *X* =>*Y* is one where either
  - *Y* is a subset of *X*, or

Example from Wikipedia

Restaurant	Pizza	DeliveryArea
Pizza Milano	Thin crust	Ahmedabad
Pizza Milano	Thick crust	Ahmedabad
Pizza Firenze	Thin crust	Gandhinagar
Pizza Firenze	Thick crust	Gandhinagar
Pizza Milano	Thin crust	Gandhinagar
Pizza Milano	Thick crust	Gandhinagar

Fourth Normal Form

- A table is in 4NF iff
  - In BCNF
  - No (or only one) multivalued dependency

  For every non-trivial multivalued dependencies *X => Y*, *X* is either:
    - A candidate key or
    - A superset of a candidate key

X is Unique

Example from Wikipedia

Restaurant	Pizza	DeliveryArea
Pizza Milano	Thin crust	Ahmedabad
Pizza Milano	Thick crust	Ahmedabad
Pizza Firenze	Thin crust	Gandhinagar
Pizza Firenze	Thick crust	Gandhinagar
Pizza Milano	Thin crust	Gandhinagar
Pizza Milano	Thick crust	Gandhinagar

Key is { Restaurant, Pizza, DeliveryArea}

Discussion

- The table has no non-key attributes

- Two non-trivial multivalued dependencies
  - Restaurant --> Pizza
  - Restaurant --> DeliveryArea

4NF Solution

- Two separate tables
- Dependency preserved
  - Restaurant --> Pizza
  - Restaurant --> DeliveryArea
- No multivalued dependency

Restaurant	DeliveryArea
Pizza Milano	Ahmedabad
Pizza Firenze	Gandhinagar
Pizza Milano	Gandhinagar

Restaurant	Pizza
Pizza Milano	Thin crust
Pizza Milano	Thick crust
Pizza Firenze	Thin crust
Pizza Firenze	Thick crust

Join dependency

- A table *T* has ***join dependency*** if it can always be recreated by ***joining*** multiple tables each having a subset of the attributes of T

- The join dependency is said to be ***trivial*** if one of the tables in the join has all the attributes of the table *T*

- *Notation: *{ A, B, …}* on *T* (Join gives all attributes)

Fifth normal form

- A table *T* is in 5NF iff
  - Every non-trivial join dependency in it is implied by its candidate keys

- A join dependency *{A, B, … Z} on T is implied by the candidate key(s) of T iff each of A, B, …, Z is a superkey for T

An example

Store	*Brand*	*Product*
Circuit City	Apple	Tablets
Circuit City	Apple	Phones
Circuit City	Toshiba	Laptops
CompUSA	Apple	Laptops

- Note that Circuit City sells Apple tablets and phones but only Toshiba laptops

A very bad decomposition

Store	*Product*
Circuit City	Tablets
Circuit City	Phones
Circuit City	Laptops
CompUSA	Laptops

Brand	*Product*
Apple	Tablets
Apple	Phones
Apple	Laptops
Toshiba	Laptops

- Lets see what happens when we do a natural join

The result of the join

Store	*Brand*	*Product*
Circuit City	Apple	Tablets
Circuit City	Apple	Phones
Circuit City	Apple	Laptops
Circuit City	Toshiba	Laptops
CompUSA	Apple	Laptops
CompUSA	Toshiba	Laptops

- Introduces two spurious tuples

A different example table

Store	*Brand*	*Product*
Circuit City	Apple	Tablets
Circuit City	Apple	Phones
Circuit City	Apple	Laptops
Circuit City	Toshiba	Laptops
CompUSA	Apple	Laptops

- Assume now that any store carrying a given brand and selling a product that is made by that brand will ***always*** carry that product

The same decomposition

Store	*Product*
Circuit City	Tablets
Circuit City	Phones
Circuit City	Laptops
CompUSA	Laptops

Brand	*Product*
Apple	Tablets
Apple	Phones
Apple	Laptops
Toshiba	Laptops

- Let see what happens when we do a natural join

The result of the join

Store	*Brand*	*Product*
Circuit City	Apple	Tablets
Circuit City	Apple	Phones
Circuit City	Apple	Laptops
Circuit City	Toshiba	Laptops
CompUSA	Apple	Laptops
CompUSA	Toshiba	Laptops

- Still one spurious tuple

The right decomposition

Store	*Product*
Circuit City	Tablets
Circuit City	Phones
Circuit City	Laptops
CompUSA	Laptops

Brand	*Product*
Apple	Tablets
Apple	Phones
Apple	Laptops
Toshiba	Laptops

Store	*Brand*
Circuit City	Apple
Circuit City	Toshiba
CompUSA	Apple

Lossless Decomposition

General Concept

- If R(A, B, C) satisfies $A \rightarrow B$
  - We can project it on A,B and A,C ***without losing information***
  - Lossless decomposition

- $R = \pi_{AB}(R) \bowtie \pi_{AC}(R)$
  - $\pi_{AB}(R)$ is the projection of R on AB
  - $\bowtie$ is the natural join operator

Example

R

Course	*Instructor*	*Text*
4330	Paris	none
4330	Cheng	none
3330	Hillford	Patterson & Hennessy

- Observe that Course $\rightarrow$ Text

A lossless decomposition

$\pi_{\text{Course, Text}}(R)$

Not a candidate key

Course	*Text*
4330	none
3330	Patterson & Hennessy

$\pi_{\text{Course, Instructor}}(R)$

Course	*Instructor*
4330	Paris
4330	Cheng
3330	Hillford

A different case

R

Course	*Instructor*	*Text*
4330	Paris	Silberschatz and Peterson
4330	Cheng	none
3330	Hillford	Patterson & Hennessy

- Now Course $\rightarrow$ Text
- R cannot be decomposed

A lossy decomposition

$\pi_{\text{Course, Text}}(R)$

Course	*Text*
4330	none
4330	Silberschatz & Peterson
3330	Patterson & Hennessy

$\pi_{\text{Course, Instructor}}(R)$

Course	*Instructor*
4330	Paris
4330	Cheng
3330	Hillford

An Example

Normalisation Example

- We have a table representing orders in an online store
- Each row represents an item on a particular order
- Primary key is {Order, Product}

- Columns
  - Order
  - Product
  - Quantity
  - UnitPrice
  - Customer
  - Address

Functional Dependencies

- Columns
  - Order
  - Product
  - Quantity
  - UnitPrice
  - Customer
  - Address

- Each order is for a single customer:
  - Order $\rightarrow$ Customer
- Each customer has a single address
  - Customer $\rightarrow$ Address
- Each product has a single price
  - Product $\rightarrow$ UnitPrice

- As Order $\rightarrow$ Customer and Customer $\rightarrow$ Address
  - Order $\rightarrow$ Address

Normalisation to 2NF

- Second normal form means no partial dependencies on candidate keys
  - {Order} $\rightarrow$ {Customer, Address}
  - {Product} $\rightarrow$ {UnitPrice}

- To remove the first FD we project over

  {Order, Customer, Address} (R1)

  and

  {Order, Product, Quantity, UnitPrice} (R2)

2NF Solution (I)

- ***First decomposition***
  - First table

Order	Product	Quantity	UnitPrice

  - Second table

Order	Customer	Address

Normalisation to 2NF

- R2
  - <u>Order</u>
  - <u>Product</u>
  - Quantity
  - UnitPrice

- R1 is now in 2NF, but there is still a partial FD in R2

  {Product} $\rightarrow$ {UnitPrice}

- To remove this we project over

{Product, UnitPrice} (R3)

and

{Order, Product, Quantity} (R4)

2NF Solution (II)

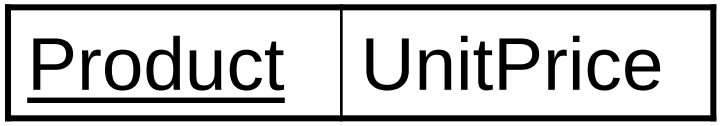
- ***Second decomposition***
  - First table

R4

  - Second table

R1

  - Third table



R3

Normalisation to 3NF

- R has now been split into 3 relations - R1, R3, and R4
  - R3 and R4 are in 3NF
  - R1 has a transitive FD on its key
- To remove

  $\{Order\} \rightarrow \{Customer\} \rightarrow \{Address\}$
- we project R1 over
  - {Order, Customer}
  - {Customer, Address}

3NF

- In second table

Order	Customer	Address

- Split second table into

Order	Customer

Customer	Address

Normalisation

- 1NF:
  - {Order, Product, Customer, Address, Quantity, UnitPrice}
- 2NF:
  - {Order, Customer, Address}, {Product, UnitPrice}, and {Order, Product, Quantity}
- 3NF:
  - {Product, UnitPrice}, {Order, Product, Quantity}, {Order, Customer}, and {Customer, Address}