Normalization

Database II

Chea Daly

Normalization

- Normalization is a database design technique that correct table structures in the relational database to minimize the data redundancy and eliminate data anomalies (insertion, update, and deletion anomalies).
- Normalization divides larger tables into smaller tables and links them using relationship.

Insertion Anomaly

- Example:

StuID	StuName	Department	Head	OfficeNo
1	A	CS	Mr. X	101

Insertion Anomaly

- Example:

StuID	StuName	Department	Head	OfficeNo
1	A	CS	Mr. X	101
2	B	CS	Mr. X	101

Insertion Anomaly

- Example:

StuID	StuName	Department	Head	OfficeNo
1	A	CS	Mr. X	101
2	B	CS	Mr. X	101
3	C	CS	Mr. X	101

Insertion Anomaly

- Example:

StuID	StuName	Department	Head	OfficeNo
1	A	CS	Mr. X	101
2	B	CS	Mr. X	101
3	C	CS	Mr. X	101
4	D	CS	Mr. X	101

The same Deparment, Head and OfficeNo. are repeated when inserting a new student.

Update Anomaly

- Example:

StuID	StuName	Department	Head	OfficeNo
1	A	CS	Mr. X	101
2	B	CS	Mr. X	101
3	C	CS	Mr. X	101
4	D	CS	Mr. X	101

- Mr. X leaves, and Mr. Y joins as the new Head of CS Department.

Update Anomaly

- Example:

StuID	StuName	Department	Head	OfficeNo
1	A	CS	~~Mr. X~~ Mr. Y	101
2	B	CS	Mr. X	101
3	C	CS	Mr. X	101
4	D	CS	Mr. X	101

Update Anomaly

- Example:

StuID	StuName	Department	Head	OfficeNo
1	A	CS	~~Mr. X~~ Mr. Y	101
2	B	CS	~~Mr. X~~ Mr. Y	101
3	C	CS	Mr. X	101
4	D	CS	Mr. X	101

Update Anomaly

- Example:

StuID	StuName	Department	Head	OfficeNo
1	A	CS	~~Mr. X~~ Mr. Y	101
2	B	CS	~~Mr. X~~ Mr. Y	101
3	C	CS	~~Mr. X~~ Mr. Y	101
4	D	CS	Mr. X	101

Update Anomaly

- Example:

StuID	StuName	Department	Head	OfficeNo
1	A	CS	~~Mr. X~~ Mr. Y	101
2	B	CS	~~Mr. X~~ Mr. Y	101
3	C	CS	~~Mr. X~~ Mr. Y	101
4	D	CS	~~Mr. X~~ Mr. Y	101

- We have to update every single record with the new head of department´s name.

Update Anomaly

- Example:

StuID	StuName	Department	Head	OfficeNo
1	A	CS	~~Mr. X~~ Mr. Y	101
2	B	CS	~~Mr. X~~ Mr. Y	101
3	C	CS	Mr. X	101
4	D	CS	~~Mr. X~~ Mr. Y	101

- If even a single row is missed out, it will lead to inconsistent data.

Deletion Anomaly

- Example:

StuID	StuName	Department	Head	OfficeNo
1	A	CS	Mr. X	101
2	B	CS	Mr. X	101
3	C	CS	Mr. X	101
4	D	CS	Mr. X	101

Deletion Anomaly

- Example:

StuID	StuName	Department	Head	OfficeNo
1	A	CS	Mr. X	101
2	B	CS	Mr. X	101
3	C	CS	Mr. X	101

Deletion Anomaly

- Example:

StuID	StuName	Department	Head	OfficeNo
1	A	CS	Mr. X	101
2	B	CS	Mr. X	101

Deletion Anomaly

- Example:

StuID	StuName	Department	Head	OfficeNo
1	A	CS	Mr. X	101

Deletion Anomaly

- Example:

StuID	StuName	Department	Head	OfficeNo

When we delete student information, department information is also deleted. And when we delete the last row of the student information, we also unintentionally delete the department information. Now, we do not have any student and we also do not have any department information.

Need for Normalization

- Used while designing a new database structure
  - Analyzes the relationship among the attributes within each entity
  - Determines if the structure can be improved
- Improves the existing data structure and creates an appropriate database design.

Normal Forms

- The normal form is used to reduce redundancy from the database table.
- Normal forms
  - First Normal Form (1NF)
  - Second Normal Form (2NF)
  - Third Normal Form (3NF)
  - …
- Structural point of view of normal forms
  - Higher normal forms are better than lower normal forms

Objectives of Normalization

- To ensure that each table conforms to the concept of well-formed relations:
  - Each table represents a single subject
  - No data will be unnecessarily stored in more than one table
  - All non-key attributes in a table are dependent on the primary key or composite primary key.
  - Each table is void of data anomalies (insertion, update, and deletion anomalies)

Objectives of Normalization

- Ensures that all tables are in at least 3NF
  - Higher forms are not likely to be encountered in business environment.
  - The Theory of Data Normalization is still being developed further. For example, there are discussions even on 6th Normal Form. However, in most practical applications, normalization achieves its best in 3NF.

Functional Dependence

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

$$A \rightarrow B$$

- The left side of FD is known as a determinant, the right side of the production is known as a dependent.

Functional Dependence

Example: we have a table, Employee (EmpID, EmpName, EmpAddress)

EmpID attribute can uniquely identify the EmpName attribute of Employee table because if we know the EmpID, we can tell employee name associated with it. Same for EmpAddress.

Functional dependencies can be written as:

EmpID → EmpName

EmpID → EmpAddress

This can be summarized as:

EmpID → EmpName, EmpAddress

Note: Determinant can be formed by multiple attributes.

Types of Functional Dependencies

- Fully-Functional Dependency
- Partial Dependency
- Transitive Dependency

Fully-Functional Dependency

- Functional dependence in which the determinant is the primary key or composite primary key.
- An attribute is fully functional dependent on another attribute, if it is functionally dependent on that attribute and not on any of its proper subset.

Fully-Functional Dependency

- Example 01:

ProjectID	ProjectName	Cost
1	Geo Location	1000
2	Cluster Exploration	5000

ProjectID can determine the ProjectName and Cost. In other words, ProjectName and Cost are fully dependent on Project.

ProjectID → ProjectName, Cost

Fully-Functional Dependency

- Example 02:

ProjectID	EmployeeID	DurationInDay
1	11	30
2	12	50
1	13	45
2	11	25

{EmployeeID, ProjectID} can determine the DurationInDay, while ProjectID alone or EmployeeID alone cannot determine the DurationInDay. In other words, DurationInDay is fully dependent on {EmployeeID, ProjectID}.

EmployeeID, ProjectID → DurationInDay

Partial Dependency

- Functional dependence in which the determinant is only part of the composite primary key.
- Example:

ProjectID	StudentID	ProjectName	StudentName
1	11	Geo Location	Mary
2	12	Cluster Exploration	Smith
1	13	Geo Location	Lucy
2	11	Cluster Exploration	Mary

- In the above table:
  - The key attributes are ProjectID and StudentID.
  - The non-key attributes are ProjectName and StudentName.

Partial Dependency

ProjectID, StudentID → ProjectName, StudentName

- However, the ProjectName and StudentName can be Partially Dependent.
  - The ProjectName can be determined by ProjectID.
  - The StudentName can be determined by StudentID.

Transitive Dependency

- An attribute functionally depends on another non-key attribute.
- A → C is a transitive dependency if A → B and B → C.
- Example:

Book	Author	AuthorAge
Game of Thrones	George R. R. Martin	66
Harry Potter	J. K. Rowling	49

- If we know the Book, we knows the AuthorAge because Book → Author and Author → AuthorAge => Book → AuthorAge

Unnormalized Form

- Example:

StuID	StuName	Subject
1	Sok	OS, CN
2	Pisey	Java
3	Dara	C, C++

Repeating Group

- **Repeating group**: Group of multiple entries of same type can exist for any single key attribute occurrence.

Normal Forms

Normal Form	Description
1NF	In a table format, has no repeating group, All key attributes are defined, the primary key is identified.
2NF	In a 1NF, and has no partial dependency - all non-key attributes are fully functional dependent on the primary key.
3NF	In a 2NF, and has no transitive dependency.

Normalization Example

Normalize a report below to 3NF:

INVOICE

No: 810002

Customer:

Date: 10/12/2020

Name: Mr. Siwon Hong

Address: #23, 102, Jongno-gu

Phone: 010 2342 1123

No.	Item	Quantity	UnitPrice	Amount
1	Rice	10	$2	$20
2	Water	5	$1	$5

Sub Total	:	$25
Discount	:	10%
Total	:	$22.5

Cashier: Ms. Park Soyeon

Conversion to UNF

InvoiceNo	InvoiceDate	CusName	Gender	CusPhone	CusAddress
810002	10/12/2020	Siwon Hong	M	010 2342 1123	#23, 102, Jongno-gu

StaffName	Gender	StaffPosition	SubTotal	Discount	Total
Park Soyeon	F	Cashier	$25	0.10	$22.5

ItemName	SaleQty	UnitPrice	Amount
Rice, Water	10, 5	$2, $1	$20, $5

Conversion to 1NF

- 1NF:
  - All key attributes are defined
  - There is no repeating group in the table
  - All attributes are dependent on the primary key or composite primary key

Conversion to 1NF

InvoiceNo	InvoiceDate	CusID	CusName	CusGender	CusPhone	CusAddress
810002	10/12/2020	1	Siwon Hong	M	010 2342 1123	#23, 102, Jongno-gu
810002	10/12/2020	1	Siwon Hong	M	010 2342 1123	#23, 102, Jongno-gu

StaffID	StaffName	StaffGender	StaffPosition	SubTotal	Discount	Total
1	Park Soyeon	F	Cashier	$25	0.10	$22.5
1	Park Soyeon	F	Cashier	$25	0.10	$22.5

ItemID	ItemName	SaleQty	UnitPrice	Amount
1	Rice	10	$2	$20
2	Water	5	$1	$5

Conversion to 1NF

- {InvoiceNo, ItemID} is the composite primary key of the table.

- Therefore, we got one table in 1NF:

InvoiceDetail({InvoiceNo, ItemID}PK, InvoiceDate, CusID, CusName,
CusGender, CusPhone, CusAddress, StaffID, StaffName,
StaffGender, StaffPosition, SubAmount, Discount, Total,
ItemName, SaleQty, UnitPrice, Amount)

Conversion to 2NF

- Table is in 2NF when it:
  - Is in 1NF
  - No partial dependencies

Conversion to 2NF

InvoiceDetail({InvoiceNo, ItemID}PK, InvoiceDate, CusID, CusName, CusGender, CusPhone, CusAddress, StaffID, StaffName, StaffGender, StaffPosition, SubAmount, Discount, Total, ItemName, SaleQty, UnitPrice, Amount)

In the 1NF table above, there exist partial dependencies:

InvoiceNo → InvoiceDate, CusID, CusName, CusGender, CusPhone, CusAddress, StaffID, StaffName, StaffGender, StaffPosition, SubAmount, Discount, Total

ItemID → ItemName, UnitPrice

Conversion to 2NF

Therefore, we got three tables in 2NF:

Item({ItemID}PK, ItemName, UnitPrice)

InvoiceDetail({InvoiceNo, ItemID}PK, SaleQty, Amount)

Invoice({InvoiceNo}PK, InvoiceDate, CusID, CusName,
CusGender, CusPhone, CusAddress, StaffID, StaffName,
StaffGender, StaffPosition, SubAmount, Discount, Total)

Conversion to 3NF

- Table is in 3NF when it:
  - Is in 2NF
  - Contains no transitive dependencies

Conversion to 3NF

- Item and InvoiceDetail tables has no transitive dependency, so they are already in 3NF.
- Invoice table, there exist transitive dependencies:

InvoiceNo → CusID

CusID → CusName, CusGender, CusPhone, CusAddress

=> InvoiceNo → CusName, CusPhone, CusAddress

InvoiceNo → StaffID

StaffID → StaffName, StaffGender, StaffPosition

=> InvoiceNo → StaffName, StaffGender, StaffPosition

Conversion to 3NF

- Therefore, we got five tables in 3NF:

Item({ItemID}PK ItemName, UnitPrice)

InvoiceDetail({InvoiceNo, ItemID}PK, SaleQty, Amount)

Invoice({InvoiceNo}PK, InvoiceDate, CusID, StaffID, SubAmount, Discount, Total)

Customer({CusID}PK, CusName, CusGender, CusPhone, CusAddress)

Staff({StaffID}PK, StaffName, StaffGender, StaffPosition)

Normalization and Database Design

- Normalization should be part of the database design process.
- Proposed entities must meet required the normal form (3NF) before table structures are created.

References

- **Carlos M. Coronel**, “Database Systems Design, Implementation, & Management” – 2018
- javatpoint.com
- tutorialspoint.com