Information And Database Management Systems (CSE 227)

Vikas Bajpai

Normalization:

- Set of rules to allocate data to tables.
- Purpose of Normalization:
  - To minimize the redundancy.
  - To minimize anomalies* related to insert, delete and update of data.
  - improve storage efficiency, data integrity, and scalability

Normalization:

- In the relational model, methods exist for quantifying how efficient a database is.
- These classifications are called normal forms (or NF), and there are algorithms for converting a given database between them.
- Normalization generally involves splitting existing tables into multiple ones, which must be re-joined or linked each time a query is issued.

Normalization:

- In simple words:
  - Normalization is a process of decomposing unsatisfactory "bad" relations by breaking up their attributes into smaller relations.
- The objective of normalization:
  - To create relations where every dependency is on the key, the whole key, and nothing but the key.

Insertion anomaly:

- It is a failure to place information into all the places in database.
- In a properly normalized database, information about a new entry needs to be inserted into only one place in the databse.

Deletion anomaly:

- It is a failure to remove information about an existing database entry from all places.
- In a properly normalized database, the entry needs to be deleted from only one place.

Update anomaly:

- Is a failure to update information about an existing database entry from all places.

History:

- Edgar F. Codd first proposed the process of normalization and what came to be known as the **1st normal form** in his paper *A Relational Model of Data for Large Shared Data Banks* Codd stated:

  “There is, in fact, a very simple elimination procedure which we shall call normalization. Through decomposition non-simple domains are replaced by ‘*domains whose elements are atomic (non-decomposable) values.*’”

Normal Form

- Edgar F. Codd originally established three normal forms: 1NF, 2NF and 3NF. There are now others that are generally accepted, but 3NF is widely considered to be sufficient for most applications.
- Most tables when reaching 3NF are also in BCNF (Boyce-Codd Normal Form).

Normalization:

There is a sequence to normal forms:

1NF is considered the weakest,

2NF is stronger than 1NF,

3NF is stronger than 2NF, and

BCNF is considered the strongest

Also,

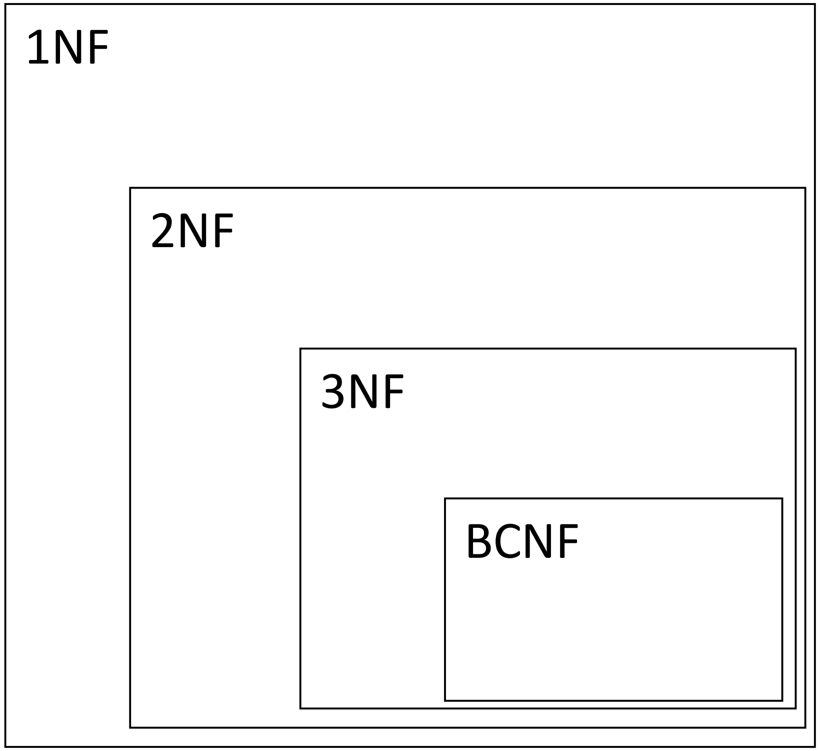
any relation that is in BCNF, is in 3NF;

any relation in 3NF is in 2NF; and

any relation in 2NF is in 1NF.

*We consider a relation in BCNF to be fully normalized.

Normalization:



a relation in BCNF, is also in 3NF

a relation in 3NF is also in 2NF

a relation in 2NF is also in 1NF

Functional Dependency:

- Functional dependency is an Integrity Constraint (some not explicitly expressed in ER diagram).
- A functional dependency occurs when one attribute in a relation uniquely determines another attribute.

Functional Dependencies:

- We say an attribute, B, has a functional dependency on another attribute, A, if for any two records, which have the same value for A, then the values for B in these two records must be the same.
- We illustrate this as:

  A → B

Functional Dependencies: Example

- Suppose we keep track of employee email addresses, and we only track one email address for each employee. Suppose each employee is identified by their unique employee number. We say there is a functional dependency of email address on employee number:

employee number → email address

Functional Dependencies: Example

EmpNum	EmpEmail	EmpFname	EmpLname
123	jdoe@abc.com	John	Doe
456	psmith@abc.com	Peter	Smith
555	alee1@abc.com	Alan	Lee
633	pdoe@abc.com	Peter	Doe
787	alee2@abc.com	Alan	Lee

If EmpNum is the PK then the FDs:

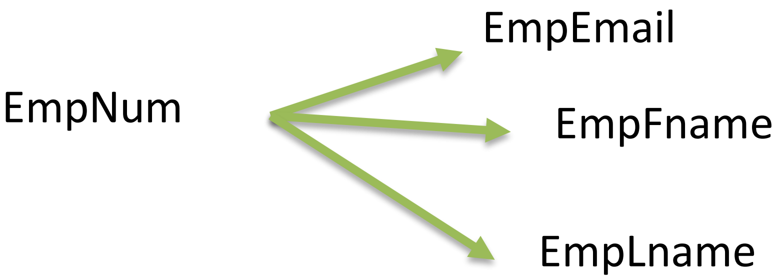
EmpNum → EmpEmail
EmpNum → EmpFname
EmpNum → EmpLname

must exist.

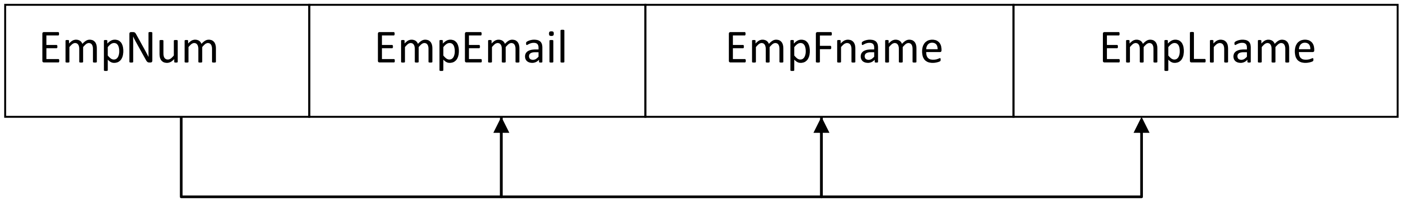
Functional Dependencies: Example

EmpNum → EmpEmail
EmpNum → EmpFname
EmpNum → EmpLname

3 different ways you might see FDs depicted



EmpEmail
EmpNum
EmpFname
EmpLname



EmpNum	EmpEmail	EmpFname	EmpLname

Determinant:

Functional Dependency:

EmpNum → EmpEmail

- Attribute on the LHS is known as the **determinant**
- EmpNum is a determinant of EmpEmail

Two types of Functional Dependencies:

1. Transitive dependency
2. Partial dependency

1. Transitive dependency

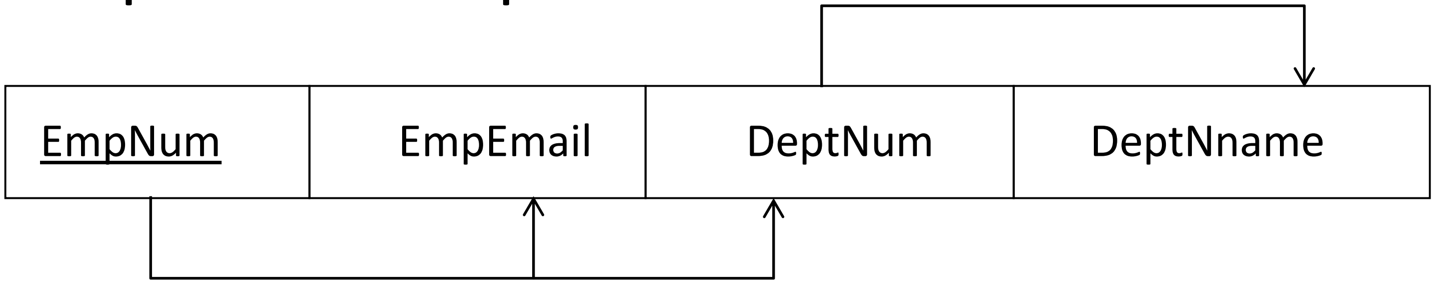
- Consider attributes A, B, and C, and where

  A → B and B → C.

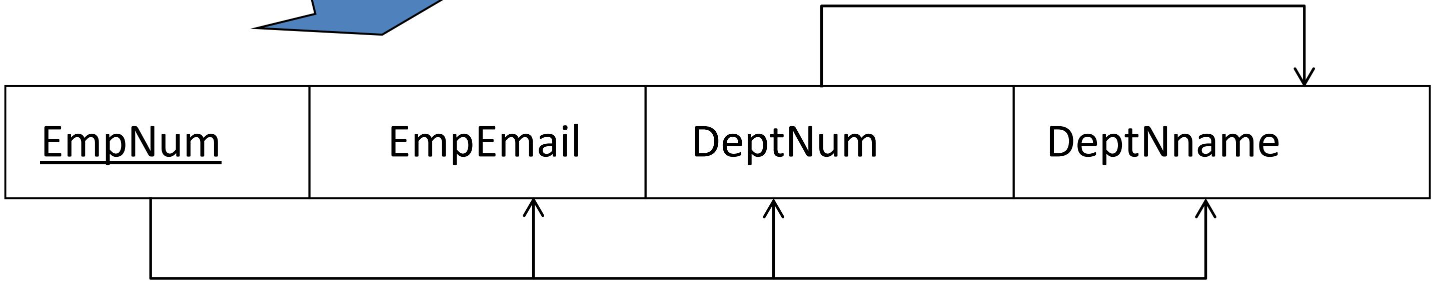
- Functional dependencies are transitive, which means that we also have the functional dependency A → C
- We say that C is transitively dependent on A through B.

Transitive dependency :Example

EmpNum → DeptNum



DeptNum → DeptName

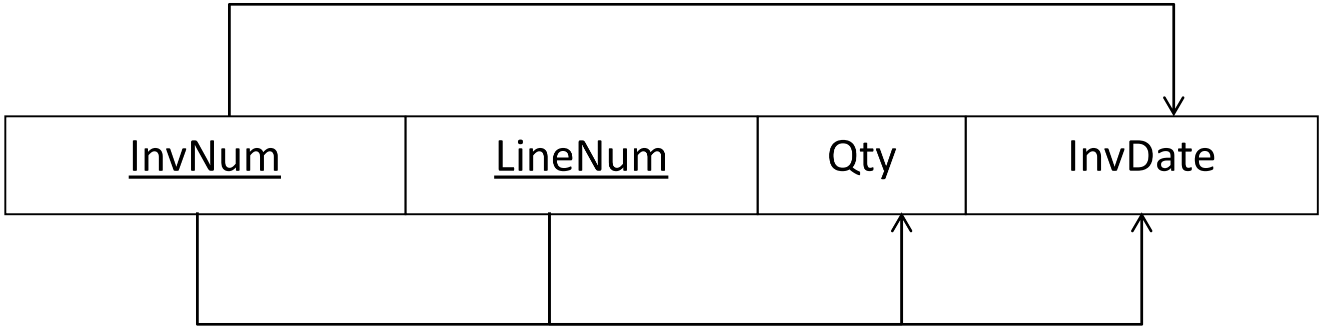


DeptName is transitively dependent on EmpNum via DeptNum

EmpNum → DeptName

2. Partial dependency

A partial dependency exists when an attribute B is functionally dependent on an attribute A, and A is a component of a multipart candidate key.



InvNum LineNum Qty InvDate

Candidate keys: {InvNum, LineNum} InvDate is *partially dependent* on {InvNum, LineNum} as InvNum is a determinant of InvDate and InvNum is part of a candidate key

First Normal Form:

- We say a relation is in **1NF** if all values stored in the relation are single-valued and atomic.
- 1NF places restrictions on the structure of relations.
- Values must be simple.
- Disallows composite attributes, multivalued attributes, and nested relations; attributes whose values for an individual tuple are non-atomic
- Considered to be part of the definition of relation

First Normal Form: Example 1

The following in **not** in 1NF:

EmpNum	EmpPhone	EmpDegrees
123	233-9876	
333	233-1231	BA, BSc, PhD
679	233-1231	BSc, MSc

- EmpDegrees is a multi-valued field:

  employee 679 has two degrees: *BSc* and *MSc*

  employee 333 has three degrees: *BA, BSc, PhD*

First Normal Form: Example 1

EmpNum	EmpPhone	EmpDegrees
123	233-9876	
333	233-1231	BA, BSc, PhD
679	233-1231	BSc, MSc

To obtain 1NF relations we must, without loss of information, replace the above with two relations

First Normal Form: Example 1

Employee

EmpNum	EmpPhone
123	233-9876
333	233-1231
679	233-1231

EmployeeDegree

EmpNum	EmpDegree
333	BA
333	BSc
333	PhD
679	BSc
679	MSc

An outer join between Employee and EmployeeDegree will produce the information we saw before

First Normal Form: Example 2

Table 1

Title	**Author1**	**Author2**	**ISBN**	**Subject**	**Pages**	**Publisher**
Database System Concepts	Abraham Silberschatz	Henry F. Korth	0072958863	MySQL, Computers	1168	McGraw-Hill
Operating System Concepts	Abraham Silberschatz	Henry F. Korth	0471694665	Computers	944	McGraw-Hill

Table 1 problems

- This table is not very efficient with storage.

- This design does not protect data integrity.

- This table does not scale well.

First Normal Form:

- In our Table 1, we have two violations of First Normal Form:
  - First, we have more than one author field,
  - Second, our subject field contains more than one piece of information.
- With more than one value in a single field, it would be very difficult to search for all books on a given subject.

First Normal Form: Example 2

Table 1

Title	Author1	Author2	ISBN	Subject	Pages	Publisher
Database System Concepts	Abraham Silberschatz	Henry F. Korth	0072958863	MySQL, Computers	1168	McGraw-Hill
Operating System Concepts	Abraham Silberschatz	Henry F. Korth	0471694665	Computers	944	McGraw-Hill

First Normal Table

- Table 2

Title	Author	ISBN	Subject	Pages	Publisher
Database System Concepts	Abraham Silberschatz	0072958863	MySQL	1168	McGraw-Hill
Database System Concepts	Henry F. Korth	0072958863	Computers	1168	McGraw-Hill
Operating System Concepts	Henry F. Korth	0471694665	Computers	944	McGraw-Hill
Operating System Concepts	Abraham Silberschatz	0471694665	Computers	944	McGraw-Hill

- We now have two rows for a single book. Additionally, we would be violating the Second Normal Form…
- A better solution to our problem would be to separate the data into separate tables- an Author table and a Subject table to store our information, removing that information from the Book table:

Subject Table

Subject_ID	Subject
1	MySQL
2	Computers

Author Table

Author_ID	Last Name	First Name
1	Silberschatz	Abraham
2	Korth	Henry

Book Table

ISBN	Title	Pages	Publisher
0072958863	Database System Concepts	1168	McGraw-Hill
0471694665	Operating System Concepts	944	McGraw-Hill

- Each table has a primary key, used for joining tables together when querying the data.
- A primary key value must be unique with in the table (no two books can have the same ISBN number), and a primary key is also an index, which speeds up data retrieval based on the primary key.
- Now to define relationships between the tables:

Relationships

Book_Author Table

ISBN	Author_ID
0072958863	1
0072958863	2
0471694665	1
0471694665	2

Book_Subject Table

ISBN	Subject_ID
0072958863	1
0072958863	2
0471694665	2

Example for Practice

Product_ID	Product_Color	Price
101	Black, Brown	$20
102	Yellow	$40
103	Red	$50
104	Pink, Violet	$60
105	White	$30

First Normal Table:

Product_Price

Product_ID	Price
101	$20
102	$40
103	$50
104	$60
105	$30

Product_ID_Color

Product_ID	Product_Color
101	Black
101	Brown
102	Yellow
103	Red
104	Pink
104	Violet
105	White

Second Normal Form:

- As the First Normal Form deals with redundancy of data across a horizontal row, Second Normal Form (or 2NF) deals with redundancy of data in vertical columns.
- As stated earlier, the normal forms are progressive, so to achieve Second Normal Form, the tables must already be in First Normal Form.

Second Normal Form: Example 1

Item	Colors	Price	Tax
T-shirt	Red, Blue	$12	$0.60
Jeans	Red, Yellow	$12	$0.60
T-shirt	Red, Blue	$12	$0.60
Sweatshirt	Blue, Black	$25	$1.20

- Table is not in first normal form because:
  - Multiple items in color field
  - Duplicate records
  - No primary key

In first normal form:

Item	Colors	Price	Tax
T-shirt	Red	$12	$0.60
T-shirt	Blue	$12	$0.60
Jeans	Red	$12	$0.60
Jeans	Yellow	$12	$0.60
Sweatshirt	Blue	$12	$1.20
Sweatshirt	Black	$12	$1.20

- For Second Normal Form (2NF):
  - All non-key field depend on all components primary key
  - Guaranteed when primary key is a single field.

Item	Colors	Price	Tax
T-shirt	Red	$12	$0.60
T-shirt	Blue	$12	$0.60
Jeans	Red	$12	$0.60
Jeans	Yellow	$12	$0.60
Sweatshirt	Blue	$12	$1.20
Sweatshirt	Black	$12	$1.20

- Table is not in second normal form because:
  - Price and tax depends on Item but not on colors

Table in second normal form:

Item	Color
T-shirt	Red
T-shirt	Blue
Jeans	Red
Jeans	Yellow
Sweatshirt	Blue
Sweatshirt	Black

Item	Price	Tax
T-shirt	$12	$0.60
Jeans	$12	$0.60
Sweatshirt	$25	$1.20

Example for Practice

Customer_ID	Store_ID	Purchase Location
1	1	Ganpati Plaza
1	3	Gaurav Tower
2	1	Ganpati Plaza
3	2	Raja Park
4	3	Gaurav Tower

Second Normal Form

Customer_ID	Store_ID
1	1
1	3
2	1
3	2
4	3

Store_ID	Location
1	Ganpati Plaza
2	Raja Park
3	Gaurav Tower

Third Normal Form

- Third normal form (3NF) requires that there are no functional dependencies of non-key attributes on something other than a candidate key.
- A table is in 3NF if all of the non-primary key attributes are mutually independent
- There should not be transitive dependencies

Boyce-Codd Normal Form

- BCNF requires that the table is 3NF and only determinants are the candidate keys

Differences:

- **1NF:** A table is set to be in first NF if we identify the functional dependency. It has no multivalued attributes.
- **2NF:** A table is set to be in first NF if we identify and delete partial functional dependency. Every non key attribute should depend on key attributes
- **3NF:** A table is set to be in 3rd NF when we identify and delete transitive dependency.
- **Functional dependency:** identify a non key attribute which is depends on key attribute.
- **Partial functional dependency:** identify an attribute which is partially depends on key attribute.
- **Transitive:** identify an key attribute which is independent itself.