CSC 365

Introduction to Database Systems

Normalization - Why?

- *Performance* - can we get the data we need quickly?
- *Integrity* - does our data remain accurate and consistent over time?
- *Maintainability* - can we easily extend or revise the structure of our database?

- The normalization process eliminates **anomalies**

Normalize to Avoid Anomalies

Anomaly: An error or inconsistency that may result when a user attempts to change data in a table that contains redundant or poorly-structured data. The three types of anomalies are:

1. Insertion anomaly
2. Deletion anomaly
3. Modification (or update) anomaly

Anomalies - Insertion

CAL POLY

STUDENT_ENROLLMENT(Student_ID, Student_Name, Course_ID, Date_Completed)

Student_ID	Student_Name	Course_ID	Date_Completed
3874	Boris Medina	CSC-349	12/8/2018
4059	Patricia Jones	CSC-366	3/21/2017
4059	Patricia Jones	CSC-365	12/8/2018
3490	Sebastian Martin	*null*	*null*

Since the primary key is (Student_ID, Course_ID) *both* values are required to insert a new student record (recall that primary key values may not be null) This is an **insertion anomaly**. The user should be allowed to enter student data without supplying course data.

Anomalies - Deletion

CAL POLY

Student_ID	Student_Name	Course_ID	Date_Complete
3874	Boris Medina	CSC-349	12/8/2017
4059	Patricia Jones	CSC-366	3/21/2018
4059	Patricia Jones	CSC-365	12/8/2018
3490	Sebastian Martin	CSC-365	12/6/2017

If the student with ID 4059 is deleted from the database (rows 2 and 3 above), we would no longer have any record of the course CSC 366; we lose the data about when that course might have been offered. This loss of data is a **deletion anomaly**.

Anomalies - Modification

CAL POLY

Student_ID	Student_Name	Course_ID	Date_Complete
3874	Boris Medina	CSC-349	12/8/2015
4059	Patricia Jones	CSC-365	3/21/2016
4059	Patricia Jones	CSC-366	12/8/2015
3490	Sebastian Martin	CSC-365	12/6/2014

If the student with ID 4059 were to change her name, we would need to make the change for all rows in the table (all courses the student had ever completed) If we miss one, the database would be inconsistent. This is an example of a **modification anomaly** (or update anomaly)

Functional Dependency

$A \rightarrow B$

A determines B

Values of B are determined by values of A

Two records sharing the same values of A will necessarily have the same values of B.

Functional Dependency

Functional dependency: A constraint between two attributes in which the value of one attribute is determined by the value of another attribute or set of attributes

Given the relation:
STUDENT_ENROLLMENT(StudentID, CourseID, DateCompleted)

StudentID, CourseID → DateCompleted

Functional Dependency: Definition

Definition of **functional dependency**:

Given a table R, a set of attributes B is functionally dependent on another set of attributes A if, at each instant in time, each A value is associated with only one B value. We use the notation A → B

Example Functional Dependencies

Sample functional dependencies:

- SSN → Name, CurrentAddress, Birthdate
  - The above notation is shorthand, equivalent to these rules:
    - SSN → Name
    - SSN → CurrentAddress
    - SSN → Birthdate
- VIN → Make, Model, Color, LicensePlate, State
- LicensePlate, State → VIN
- ISBN → Title, Author, Publisher, PublicationYear

Normalization - Functional Dependency

CAL POLY

Employee_ID	Employee_Name	Department_ID	Department_Name
3874	Sebastian	HR-NYC	Human Resources
4059	Ryan	MKTG	Marketing
6584	Monica	HR-NYC	Human Resources
3490	William	SALES	Sales
8736	Monica	HR-CHICAGO	Human Resources

Normalization - Functional Dependency

Functional dependencies that exist on the previous slide:

- Employee_ID → Employee_Name
- Employee_ID → Department_ID
- Department_ID → Department_Name

Normalization - First Normal Form (1NF)

A table is in first normal form (1NF) if and only if all columns contain only *atomic* values. That is, each column can have only one value for each row in the table.

In addition, *a primary key is required for 1NF*

Normalization - Example of 1NF Violation

CAL POLY

Employee_ID	Employee_Name	Phone_Number
3874	Boris Medina	555-345-9020
4059	Patricia Jones	805-555-9839, 555-324-8299
3490	Sebastian Martin	805-555-3000, 555-555-3492

Normalization - Second Normal Form (2NF)

A table is in second normal form (2NF) if and only if it is in 1NF and every non-key attribute is fully dependent on the primary key.

An attribute is fully dependent on the primary key if it is on the right side of a **functional dependency** (FD) for which the left side is either the primary key itself or something that can be derived from the primary key using the transitivity of FDs.

Normalization - Second Normal Form (2NF)

Fully dependent vs. partially dependent.

Partial functional dependency: A functional dependency in which one or more non-key attributes are functionally dependent on part *(but not all)* of any candidate key

Normalization - Functional Dependencies (not in 2NF)

CAL POLY

ORDER(OrderID, OrderDate, CustomerID, CustomerName, ProductID, ProductDescription, Quantity)

OrderID	OrderDate	CustomerID	CustomerName	ProductID	ProductDescription	Quantity
1604	8/1/2019	2	Lillian Kleiner	9203	Olive Oil	2
1604	8/1/2019	2	Lillian Kleiner	0390	Couscous	2
1604	8/1/2019	2	Lillian Kleiner	4909	Pesto Sauce	5
1503	3/6/2018	4	Maximo Drysdell	1103	Chef Hat 20cm	1
1503	3/6/2018	4	Maximo Drysdell	2390	Pasta - Orzo, Dry	5

Partial dependencies? Full Dependencies? Transitive Dependencies?

Normalization - Functional Dependency Example

- Full (entire key on the left side)
  - OrderID, ProductID → Quantity
- Partial (only part of *any candidate key* on left side)
  - OrderID → OrderDate, CustomerID, CustomerName
  - ProductID → ProductDescription
- "Regular" Functional Dependency (neither full nor partial)
  - CustomerID → CustomerName
- Transitive
  - OrderID → CustomerName *(through CustomerID)*

Normalization - Second Normal Form (2NF)

A relation is in second normal form if *any one* of these applies:

1. Primary key consists of just one attribute
2. All attributes in the relation are components of the primary key
3. Every non-key attribute is functionally dependent on the *full set* of key attributes

Normalization - Second Normal Form (2NF)

Formal definition of **2NF**:

A table is in second normal form (2NF) if any only if it is in 1NF and every non-key attribute is fully dependent on the primary key.

An attribute is fully dependent on the primary key if it is on the right side of an FD for which the left side is either the primary key itself or something that can be derived from the primary key using the transitivity of FDs.

Normalization - Second Normal Form (2NF)

Second Normal Form (2NF) Meets 1NF and every non-key attribute is *fully* dependent on the primary key

Non-key refers to an attribute that is not a part of *any candidate key* of the table

Fully dependent means that the attribute depends on the *entire* primary key (important in the case of composite keys)

Normalization - Functional Dependencies and Keys

CAL POLY

CUSTOMER(Name, ZipCode, City, State)

Name	ZipCode	City	State
Patricia	12345	SCHENECTADY	NY
Juliana	23456	VIRGINIA BEACH	VA
Sebastian	87654	SPACEPORT CITY	NM
Jules	12345	SCHENECTADY	NY
Dean	45678	SCOTTOWN	OH
Jules	10001	NEW YORK CITY	NY

- Functional Dependencies
  - Name, ZipCode → City, State
  - ZipCode → City, State
  - City → State *
- Superkeys
  - Name, ZipCode, City, State
  - Name, ZipCode, City
  - Name, ZipCode
  - Name, ZipCode, State
  - Name, City *
  - Name, City, State *
- Candidate Keys
  - Name, ZipCode
  - Name, City *
- Primary Key
  - Name, ZipCode

Normalization - Example of 2NF Violation

CAL POLY

CUSTOMER(Name, Zip_Code, City, State)

Name	**Zip_Code**	**City**	**State**
Patricia	12345	SCHENECTADY	NY
Juliana	23456	VIRGINIA BEACH	VA
Sebastian	87654	SPACEPORT CITY	NM
Jules	12345	SCHENECTADY	NY
Dean	45678	SCOTTOWN	OH
Jules	10001	NEW YORK CITY	NY

- Functional Dependencies
  - Name, Zip_Code → City, State
  - Zip_Code → City, State
  - ~~City → State~~ *
- Candidate Keys
  - Name, Zip_Code
  - ~~Name, City~~ *
- Primary Key
  - Name, Zip_Code
- Non-Key Attributes
  - City
  - State

Normalization - 2NF Example

- Full functional dependencies?
  - Name, Zip_Code → City, State
- Partial functional dependencies?
  - Zip_Code → City, State
  - ~~City → State~~ *(not a true FD; does not hold for all possible data -- cities with the same name may exist in different states)*
- Is this table in Second Normal Form?
  - No
  - Must decompose table (split into multiple tables)

Normalization - 2NF Example

CAL POLY

CUSTOMER(Name, ZipCode, City, State)

Split

CUSTOMER(Name, ZipCode)

Name	ZipCode
Patricia	12345
Juliana	23456
Sebastian	87654
Jules	12345
Dean	45678
Jules	10001

LOCATION(ZipCode, City, State)

ZipCode	City	State
12345	SCHENECTADY	NY
23456	VIRGINIA BEACH	VA
87654	SPACEPORT CITY	NM
45678	SCOTTOWN	OH
10001	NEW YORK CITY	NY

Note that we deleted a duplicate entry for Schenectady, NY

Normalization - Third Normal Form (3NF)

Third Normal Form (3NF): Both of these properties must be true:

1. The relation R (table) is in second normal form (2NF)
2. Every non-key attribute of R is non-transitively dependent on every key of R

Recall that a **non-key attribute** of R is an attribute that does not belong to *any candidate key* of R

Transitive dependency:

$A \rightarrow B$
$B \rightarrow C$

C is *transitively dependent* on A (through B)

Note: For transitive dependency to hold, it *must not* be the case that $B \rightarrow A$

Normalization - 3NF Example

CAL POLY

BOOK(ISBN, Title, Author, Year, Publisher, PublisherCity)

ISBN	Title	Author	Year	Publisher	PublisherCity
0-914894-36-6	Database Systems	Ullman	1982	Pittman Publishing	London
0-201-74128-8	Data Mining	Rogier	2003	Addison-Wesley	Boston
0-321-33025-0	Programming Languages	Sebesta	2005	Addison-Wesley	Boston
0-13-17480-7	Spatial Databases	Shekhar	2003	Prentice Hall	Upper Saddle River
0-13-678012-1	Programming Languages	Pratt	1996	Prentice Hall	Upper Saddle River

Example developed by A. Dekhtyar, http://users.csc.calpoly.edu/~dekhtyar/

Is This Table in 2NF?

CAL POLY

***2NF**: Meets 1NF and every non-key attribute is fully dependent on the primary key*

BOOK(ISBN, Title, Author, Year, Publisher, PublisherCity)

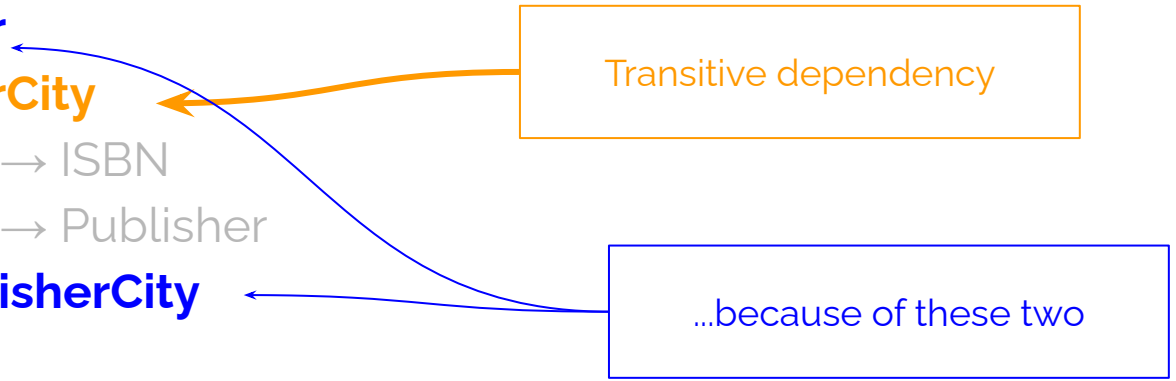
ISBN	**Title**	**Author**	**Year**	**Publisher**	**PublisherCity**
0-914894-36-6	Database Systems	Ullman	1982	Pittman Publishing	London
0-201-74128-8	Data Mining	Rogier	2003	Addison-Wesley	Boston
0-321-33025-0	Programming Languages	Sebesta	2005	Addison-Wesley	Boston
0-13-17480-7	Spatial Databases	Shekhar	2003	Prentice Hall	Upper Saddle River
0-13-678012-1	Programming Languages	Pratt	1996	Prentice Hall	Upper Saddle River

Normalization - 3NF Example - FDs

- ISBN → Title
- ISBN → Author
- ISBN → Year
- ISBN → Publisher
- ISBN → PublisherCity
- Title, Author, Year → ISBN
- Title, Author, Year → Publisher
- Publisher → PublisherCity

Normalization - 3NF Example - FDs

- ISBN → Title
- ISBN → Author
- ISBN → Year
- **ISBN → Publisher**
- **ISBN → PublisherCity**
- Title, Author, Year → ISBN
- Title, Author, Year → Publisher
- **Publisher → PublisherCity**



Normalization - 3NF Example - Transitivity

- ISBN → Publisher
- Publisher → PublisherCity
- It is *not* the case that: Publisher → ISBN

PublisherCity is transitively dependent on ISBN

This violates third normal form, which states:

... every non-key attribute of R is non-transitively dependent on every key of R.

Normalization - 3NF Example

CAL POLY

- To achieve third normal form, we split the relation:
  - BOOK(ISBN, Title, Author, Year, Publisher, PublisherCity)

- Into two relations:
  - BOOK(ISBN, Title, Author, Year, Publisher)
    PUBLISHER(Publisher, PublisherCity)

Normalization - 3NF Stated Another Way

CAL POLY

> “Every non-key attribute must provide a fact about the key, the whole key, and nothing but the key.”

-- Bill Kent (http://www.bkent.net/)

Normalization - Other Normal Forms

Additional normal forms exist (Boyce-Codd, 4NF). Time permitting, we may discuss these briefly later in the course (CSC 366 covers these in more depth)

- Boyce-Codd Normal Form (BCNF) requires that the left side of every nontrivial FD be a superkey.
- 4NF is BCNF applied to multivalued dependencies (MVD)

When to Normalize?

- Certain decisions can be made during initial data modeling. Examples include:
  - Phone number: single field or entity with one-to-many relationship?
  - Customer address: attribute (or attributes) or as a separate entity?

- Other decisions that affect normalization become more clear during logical data modeling (using SQL DDL)

Normalization - Why?

- Performance - can we get the data we need quickly?
- Integrity - does our data remain accurate and consistent over time?
- Maintainability - can we easily extend or revise the structure of our database?
- The normalization process eliminates **anomalies**
  - Insertion
  - Deletion
  - Modification

Normalization - Summary

- Well-defined rules and algorithms
- Typical result is that large tables are split into multiple smaller tables
- Normalization eliminates the chance of data anomalies (insertion, deletion, modification)