Databases I

Etelka Szendrői Dr. (PhD)
associate professor
System and Software Technology Department

szendroi@mik.pte.hu

Well-structured relations

- Well-structured relations contain minimal redundancy and allow insertion, modification, and deletion without errors or inconsistencies
- Anomalies are errors or inconsistencies resulting from redundancy
- Insertion anomaly
- Deletion anomaly
- Modification anomaly

2

Functional dependencies and keys

- In a database, we often have the case for which one attribute defines the other. For example, we can say that Social Security Number (SSN) defines or identifies a name.

  What does this mean? It means that if I have a database with SSNs and names, and if I know someone's SSN, then I can find the person's name.

Functional dependencies and keys

- Functional dependency: the value of one attribute (the determinant) determines the value of another attribute
- A -> B, for every valid instance of A, that value of A uniquely determines the value of B
- Candidate key: an attribute or combination of attributes that uniquely identifies an instance
- Uniqueness: each non-key field is functionally dependent on every candidate key
- Non-redundancy

Functional dependency

- Suppose that a company assigned each employee a unique employee number. Each employee has one employee number and one name.
- Names might be the same for two different employees, but for two employees their employee numbers would always be different and unique because the company defined them that way.
- We write an FD with an arrow like this: EmpNo -> Name
- The expression EmpNo -> Name is read “Empno defines Name” or “Empno implies Name.”

5

Functional Dependencies

- Let R be a relation schema

$$\alpha \subseteq R \text{ and } \beta \subseteq R$$

- The **functional dependency**

$$\alpha \rightarrow \beta$$

holds on R if and only if for any legal relations r(R), whenever any two tuples t_1 and t_2 of r agree on the attributes α, they also agree on the attributes β. That is,

$$t_1[\alpha] = t_2[\alpha] \Rightarrow t_1[\beta] = t_2[\beta]$$

- Example: Consider $r(A, B)$ with the following instance of r.

1	4
1	5
3	7

- On this instance, $A \rightarrow B$ does **NOT** hold, but $B \rightarrow A$ does hold.

Functional Dependencies (Cont.)

- K is a superkey for relation schema R if and only if $K \rightarrow R$
- K is a candidate key for R if and only if
  - $K \rightarrow R$, and
  - for no $\alpha \subset K$, $\alpha \rightarrow R$
- Functional dependencies allow us to express constraints that cannot be expressed using superkeys. Consider the schema:

  inst_dept (*ID, name, salary, dept_name, building, budget*).

  We expect these functional dependencies to hold:

  $$dept\_name \rightarrow building$$

  and $$ID \rightarrow building$$

  but would not expect the following to hold:

  $$dept\_name \rightarrow salary$$

Closure of a Set of Functional Dependencies

- Given a set F of functional dependencies, there are certain other functional dependencies that are logically implied by F.
  - For example: If $A \rightarrow B$ and $B \rightarrow C$, then we can infer that $A \rightarrow C$
- The set of **all** functional dependencies logically implied by F is the **closure** of F.
- We denote the *closure* of F by $\mathbf{F^+}$.
- F^+ is a superset of F.

Closure of a Set of Functional Dependencies

- We can find $F^{+,}$ the closure of F, by repeatedly applying **Armstrong's Axioms:**
  - if $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$ **(reflexivity)**
  - if $\alpha \rightarrow \beta$, then $\gamma\, \alpha \rightarrow \gamma\, \beta$ **(augmentation)**
  - if $\alpha \rightarrow \beta$, and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$ **(transitivity)**
- These rules are
  - **sound** (generate only functional dependencies that actually hold), and
  - **complete** (generate all functional dependencies that hold).

Example

- $R = (A, B, C, G, H, I)$
  $F = \{ A \rightarrow B$
  $A \rightarrow C$
  $CG \rightarrow H$
  $CG \rightarrow I$
  $B \rightarrow H\}$
- some members of F^+
  - $A \rightarrow H$
    - by transitivity from $A \rightarrow B$ *and* $B \rightarrow H$
  - $AG \rightarrow I$
    - by augmenting $A \rightarrow C$ with G, to get $AG \rightarrow CG$ and then transitivity with $CG \rightarrow I$
  - $CG \rightarrow HI$
    - by augmenting $CG \rightarrow I$ to infer $CG \rightarrow CGI$, and augmenting of $CG \rightarrow H$ to infer $CGI \rightarrow HI$, and then transitivity

Procedure for Computing F^+

- To compute the closure of a set of functional dependencies F:

$F^+ = F$
repeat
  for each functional dependency f in F^+
    apply reflexivity and augmentation rules on f
    add the resulting functional dependencies to F^+
  for each pair of functional dependencies f_1 and f_2 in F^+
    if f_1 and f_2 can be combined using transitivity
      then add the resulting functional dependency to F^+
until F^+ does not change any further

NOTE: We shall see an alternative procedure for this task later

Closure of Functional Dependencies (Cont.)

- Additional rules:
  - If $\alpha \rightarrow \beta$ holds *and* $\alpha \rightarrow \gamma$ holds, then $\alpha \rightarrow \beta\gamma$ holds **(union)**
  - If $\alpha \rightarrow \beta\gamma$ holds, then $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds **(decomposition)**
  - If $\alpha \rightarrow \beta$ holds *and* $\gamma\beta \rightarrow \delta$ holds, then $\alpha\gamma \rightarrow \delta$ holds **(pseudotransitivity)**

  The above rules can be inferred from Armstrong's axioms.

Closure of Attribute Sets

- Given a set of attributes α, define the ***closure*** of α **under** *F* (denoted by α^+) as the set of attributes that are functionally determined by α under *F*

- Algorithm to compute α^+, the closure of α under *F*

> *result* := α;
> **while** (changes to *result*) **do**
>   **for each** $\beta \rightarrow \gamma$ **in** *F* **do**
>   **begin**
>   **if** $\beta \subseteq$ *result* **then** *result* := *result* $\cup\ \gamma$
>   **end**

Example of Attribute Set Closure

- $R = (A, B, C, G, H, I)$
- $F = \{A \rightarrow B$
  $A \rightarrow C$
  $CG \rightarrow H$
  $CG \rightarrow I$
  $B \rightarrow H\}$
- $(AG)^+$
  1. *result = AG*
  2. *result = ABCG* ($A \rightarrow C$ and $A \rightarrow B$)
  3. *result = ABCGH* ($CG \rightarrow H$ and $CG \subseteq AGBC$)
  4. *result = ABCGHI* ($CG \rightarrow I$ and $CG \subseteq AGBCH$)
- Is *AG* a candidate key?
  1. Is AG a super key?
     1. Does $AG \rightarrow R$? == Is $(AG)^+ \supseteq$ R
  2. Is any subset of AG a superkey?
     1. Does $A \rightarrow R$? == Is $(A)^+ \supseteq$ R
     2. Does $G \rightarrow R$? == Is $(G)^+ \supseteq$ R

Uses of Attribute Closure

There are several uses of the attribute closure algorithm:

- Testing for superkey:
  - To test if α is a superkey, we compute $\alpha^{+,}$ and check if α^+ contains all attributes of *R*.
- Testing functional dependencies
  - To check if a functional dependency $\alpha \rightarrow \beta$ holds (or, in other words, is in F^+), just check if $\beta \subseteq \alpha^+$.
  - That is, we compute α^+ by using attribute closure, and then check if it contains β.
  - Is a simple and cheap test, and very useful
- Computing closure of F
  - For each $\gamma \subseteq R$, we find the closure γ^+, and for each $S \subseteq \gamma^+$, we output a functional dependency $\gamma \rightarrow S$.

ER Model and Normalization

- When an E-R diagram is carefully designed, identifying all entities correctly, the tables generated from the E-R diagram should not need further normalization.
- However, in a real (imperfect) design, there can be functional dependencies from non-key attributes of an entity to other attributes of the entity
  - Example: an *employee* entity with attributes *department_name* and *building*, and a functional dependency *department_name* $\rightarrow$ *building*
  - Good design would have made department an entity
- Functional dependencies from non-key attributes of a relationship set possible, but rare --- most relationships are binary

Data normalization

- Normalization is a formal process for deciding which attributes should be grouped together in a relation
- Objective: to validate and improve a logical design so that it satisfies certain constraints that avoid unnecessary duplication of data
- Definition: the process of decomposing relations with anomalies to produce smaller, well-structured relations

17

First Normal Form

A relation is in **first normal form (1NF)** if and only if all attributes are atomic.

Atomic attributes are single valued, and cannot be composite, multi-valued or nested relations.

Example:
Customer(CID, Name: First + Last, Phones, Address)

CID	Name: First + Last	Phones	Address
111	Joe Jones	111-2223 111-3393 112-4582	123 Main

18

Second Normal Form

A relation is in **second normal form (2NF)** if it is in 1NF and each non-key attribute is fully functionally dependent on the primary key.

K → Ai for each non-key attribute Ai
That is, there is no subset K' such that K' → Ai

Example:
OrderProduct(OrderID, ProductID, Quantity, Description)

OrderID	ProductID	Quantity	Description
32	15	1	Blue Hose
32	16	2	Pliers
33	15	1	Blue Hose

19

Transitive Dependency

Given functional dependencies: X → Y and Y → Z, the **transitive dependency** X → Z must also hold.

Example:
There is an FD between OrderID and CustomerID. Given the OrderID key attribute, you always know the CustomerID.

There is an FD between CustomerID and the other customer data, because CustomerID is the primary key. Given the CustomerID, you always know the corresponding attributes for Name, Phone, and so on.

Consequently, given the OrderID (X), you always know the corresponding customer data by transitivity.

20

Third Normal Form

A relation is in **third normal form** if and only if it is in 2NF and no non-key attributes are transitively dependent on the primary key.

That is, K → Ai for each attribute, (2NF) and
There is no subset of attributes X such that K → X → Ai

Example:
Order(OrderID, OrderDate, CustomerID, Name, Phone)

OrderID	OrderDate	CustomerID	Name	Phone
32	May-05	1	Jones	222-3333
33	May-05	2	Hong	444-8888
34	May-05	1	Jones	222-3333

Boyce-Codd Normal Form

A relation is in Boyce-Codd Normal Form (BCNF) if and only if it is in 3NF and every determinant is a candidate key (or K is a superkey).
That is, K → Ai for every attribute, and there is no subset X (key or nonkey) such that X → Ai where X is different from K.

Example: Employees can have many specialties, and many employees can be within a specialty. Employees can have many managers, but a manager can have only one specialty: Mgr → Specialty
EmpSpecMgr(EID, Specialty, ManagerID)

EID	Speciality	ManagerID
32	Drill	1
33	Weld	2
34	Drill	1

FD ManagerID → Specialty is not currently a key.

Multi-Valued Dependency

A multi-valued dependency (MVD) exists when there are at least three attributes in a relation (A, B, and C; and they could be sets), and one attribute (A) determines the other two (B and C) but the other two are independent of each other.

That is, A →B and A → C but B and C have no FDs

Example:
Employees have many specialties and many tools, but tools and specialties are not directly related.

23

Fourth Normal Form

A relation is in **fourth normal form 4NF** if and only if it is in BCNF and there are no multi-valued dependencies.
That is, all attributes of R are also functionally dependent on A.
If A → → B, then all attributes of R are also functionally dependent on A: A → Ai for each attribute.

Example:
EmpSpecTools(<u>EID</u>, <u>Specialty</u>, <u>ToolID</u>)

EmpSpec(<u>EID</u>, <u>Specialty</u>)
EmpTools(<u>EID</u>, <u>ToolID</u>)

24

Definitions

- Relational database: A collection of tables.
- Table: A collection of columns (attributes) describing an entity. Individual objects are stored as rows of data in the table.
- Property (attribute): a characteristic or descriptor of a class or entity.
- Every table has a primary key.
  - The smallest set of columns that uniquely identifies any row
  - Primary keys can span more than one column (concatenated keys)
  - We often create a primary key to insure uniqueness (e.g., CustomerID, Product#, . . .) called a surrogate key.

Primary key *Properties* *Class: Employee*

Rows/Objects

Employee

<u>**EmployeeID**</u>	TaxpayerID	LastName	FirstName	HomePhone	Address
12512	888-22-5552	Cartom	Abdul	(603) 323-9893	252 South Street
15293	222-55-3737	Venetiaan	Roland	(804) 888-6667	937 Paramaribo Lane
22343	293-87-4343	Johnson	John	(703) 222-9384	234 Main Street
29387	837-36-2933	Stenheim	Susan	(410) 330-9837	8934 W. Maple

26

Keys

- Primary key
  - Every table (object) must have a primary key
  - Uniquely identifies a row (one-to-one)
- Composite key
  - Multiple columns needed for primary key
  - Identify repeating relationships (1 : M or M : N)
- Key columns are underlined
- First step
  - Collect user documents
  - Identify possible keys: unique or repeating relationships

27

Notation

Table name

Table columns

Customer(CustomerID, Phone, Name, Address, City, State, ZipCode)

Primary key is underlined

CustomerID	Phone	LastName	FirstName	Address	City	State	Zipcode
1	502-666-7777	Johnson	Martha	125 Main Street	Alvaton	KY	42122
2	502-888-6464	Smith	Jack	873 Elm Street	Bowling Green	KY	42101
3	502-777-7575	Washington	Elroy	95 Easy Street	Smith's Grove	KY	42171
4	502-333-9494	Adams	Samuel	746 Brown Drive	Alvaton	KY	42122
5	502-474-4746	Rabitz	Victor	645 White Avenue	Bowling Green	KY	42102
6	616-373-4746	Steinmetz	Susan	15 Speedway Drive	Portland	TN	37148
7	615-888-4474	Lasater	Les	67 S. Ray Drive	Portland	TN	37148
8	615-452-1162	Jones	Charlie	867 Lakeside Drive	Castalian Springs	TN	37031
9	502-222-4351	Chavez	Juan	673 Industry Blvd.	Caneyville	KY	42721
10	502-444-2512	Rojo	Maria	88 Main Street	Cave City	KY	42127

28

Identifying Key Columns

Orders

OrderID	Date	Customer
8367	5-5-10	6794
8368	5-6-10	9263

Each order has only one customer. So Customer is **not** part of the key.

OrderItems

OrderID	Item	Quantity
8367	229	2
8367	253	4
8367	876	1
8368	555	4
8368	229	1

Each order has many items. Each item can appear on many orders. So OrderID and Item are **both** part of the key.

29

Sample Database for Sales

Sale ID					Date
Customer First Name Last Name Address City, State ZIPCode					
ItemID	**Description**	**List Price**	**Quantity**	**QOH**	**Value**
					Total

30

Initial Objects

Initial Object	Key	Sample Properties
Customer	Assign CustomerID	Name Address Phone
Item	Assign ItemID	Description List Price Quantity On Hand
Sale	Assign SaleID	Sale Date
SaleItems	SaleID + ItemID	Quantity

31

Initial Form Evaluation

SaleForm(<u>SaleID</u>, SaleDate, CustomerID, FirstName, LastName,
Address, City, State, ZIPCode,
(<u>ItemID</u>, Description, ListPrice, Quantity, QuantityOnHand))

Identify potential keys.
Identify repeating groups.

Sale ID					Date
Customer First Name Last Name Address City, State ZIPCode					
ItemID	**Description**	**List Price**	**Quantity**	**QOH**	**Value**
					Total

32

Problems with Repeating Sections

SaleForm(<u>SaleID</u>, SaleDate, CustomerID, FirstName, LastName, Address, City, State, ZIPCode,
(<u>ItemID</u>, Description, ListPrice, Quantity, QuantityOnHand))

Repeating section

Duplication

Not atomic

<u>SaleID</u>	Date	CID	FirstName	LastName	Address	City	State	ZIP	ItemID	Description	ListPrice	Quantity	QOH
11851	7/15	15023	Mary	Jones	111 Elm	Chicago	IL	60601	15 27 32	Air Tank Regulator Mask 1557	192.00 251.00 65.00	2 1 1	15 5 6
11852	7/15	63478	Miguel	Sanchez	222 Oro	Madrid			15 33	Air Tank Mask 2020	192.00 91.00	4 1	15 3
11853	7/16	15023	Mary	Jones	111 Elm	Chicago	IL	60601	41 75	Snorkel 71 Wet suit-S	44.00 215.00	2 1	15 3
11854	7/17	94552	Madeline	O'Reilly	333 Tam	Dublin			75 32 57	Wet suit-S Mask 1557 Snorkel 95	215.00 65.00 83.00	2 1 1	3 6 17

33

First Normal Form

SaleForm(SaleID, SaleDate, CustomerID, FirstName, LastName, Address, City, State, ZIPCode,
(ItemID, Description, ListPrice, Quantity, QuantityOnHand))

SaleForm2(SaleID, SaleDate, CustomerID, FirstName, LastName, Address, City, State, ZIPCode)

SaleLine(SaleID, ItemID, Description, ListPrice, Quantity, QuantityOnHand)

34

Multiple Repeating: Independent Groups

FormA(Key1, Simple Columns, (Group1, A, B, C), (Group2, X, Y))

MainTable(Key1, Simple Columns)

Group1(Key1, Group1, A, B, C)

Group2(Key1, Group2, X, Y)

35

First Normal Form Problems (Data)

SaleLine(<u>SaleID</u>, <u>ItemID</u>, Description, ListPrice, Quantity, QuantityOnHand)

Duplication for columns that depend only on ItemID

<u>SaleID</u>	<u>ItemID</u>	Description	ListPrice	Quantity	QOH
11851	15	Air Tank	192.00	2	15
11851	27	Regulator	251.00	1	5
11851	32	Mask 1557	65.00	1	6
11852	15	Air Tank	192.00	4	15
11852	33	Mask 2020	91.00	1	3
11853	41	Snorkel 71	44.00	2	15
11853	75	West suit-S	215.00	1	3
11854	75	Wet suit-S	215.00	2	3
11854	32	Mask 1557	65.00	1	6
11854	57	Snorkel 95	83.00	1	17

36

Second Normal Form Definition

Depends on both SaleID and ItemID

SaleLine(<u>SaleID</u>, <u>ItemID</u>, Description, ListPrice, Quantity, QuantityOnHand)

Depend only on ItemID

- Each non-key column must depend on the entire key.
  - Only applies to concatenated keys
  - Some columns only depend on part of the key
  - Split those into a new table.
- Dependence (definition)
  - If given a value for the key you always know the value of the property in question, then that property is said to depend on the key.
  - If you change part of a key and the questionable property does not change, then the table is **not** in 2NF.

37

Second Normal Form Example

SaleLine(SaleID, ItemID, Description, ListPrice, Quantity, QuantityOnHand)

SaleItems(SaleID, ItemID, Quantity)

Item(ItemID, Description, ListPrice, QuantityOnHand)

Second Normal Form Example (Data)

SaleItems(SaleID, ItemID, Quantity)

SaleID	ItemID	Quantity
11851	15	2
11851	27	1
11851	32	1
11852	15	4
11852	33	1
11853	41	2
11853	75	1
11854	75	2
11854	32	1
11854	57	1

Item(ItemID, Description, ListPrice, QuantityOnHand)

ItemID	Description	ListPrice	QOH
15	Air Tank	192.00	15
27	Regulator	251.00	5
32	Mask 1557	65.00	6
33	Mask 2020	91.00	3
41	Snorkel 71	44.00	15
57	Snorkel 95	83.00	17
75	Wet suit-S	215.00	3
77	Wet suit-M	215.00	7

Second Normal Form Problems (Data)

SaleForm2(<u>SaleID</u>, SaleDate, CustomerID, FirstName, LastName, Address, City, State, ZIPCode)

<u>SaleID</u>	Date	CustomerID	FirstName	LastName	Address	City	State	ZIP
11851	7/15	15023	Mary	Jones	111 Elm	Chicago	IL	60601
11852	7/15	63478	Miguel	Sanchez	222 Oro	Madrid		
11853	7/16	15023	Mary	Jones	111 Elm	Chicago	IL	60601
11854	7/17	94552	Madeline	O'Reilly	333 Tam	Dublin		

Duplication

Third Normal Form Definition

Depend on SaleID

SaleForm2(<u>SaleID</u>, SaleDate, CustomerID, FirstName, LastName, Address, City, State, ZIPCode)

Depend on CustomerID

Third Normal Form Example

SaleForm2(SaleID, SaleDate, CustomerID, FirstName, LastName, Address, City, State, ZIPCode)

Sale(SaleID, SaleDate, CustomerID)

SaleID	Date	CustomerID
11851	7/15	15023
11852	7/15	63478
11853	7/16	15023
11854	7/17	94552

Customer(CustomerID, FirstName, LastName, Address, City, State, ZIPCode)

CustomerID	FirstName	LastName	Address	City	State	ZIP
15023	Mary	Jones	111 Elm	Chicago	IL	60601
63478	Miguel	Sanchez	222 Oro	Madrid		
94552	Madeline	O'Reilly	333 Tam	Dublin		

42

Third Normal Form Tables

Customer(CustomerID, FirstName, LastName, Address, City, State, ZIPCode)

Sale(SaleID, SaleDate, CustomerID)

SaleItems(SaleID, ItemID, Quantity)

Item(ItemID, Description, ListPrice, QuantityOnHand)

43

3NF Rules/Procedure

- Split out repeating sections
  - Be sure to include a key from the parent section in the new piece so the two parts can be recombined.
- Verify that the keys are correct
  - Is each row uniquely identified by the primary key?
  - Are one-to-many and many-to-many relationships correct?
  - Check “many” for keyed columns and “one” for non-key columns.
- **Make sure that each non-key column depends on the whole key and nothing but the key.**
  - No hidden dependencies.

Boyce-Codd Normal Form (BCNF)

c

Employee-Specialty(EID, Specialty, Manager)

b a d

- Business rules.
- Each employee may have many specialties.
- Each specialty has many managers.
- Employee has only one manager for each specialty.
- Each manager has only one specialty.

Employee(EID, Manager)
Manager(Manager, Specialty)

Fourth Normal Form (Keys)

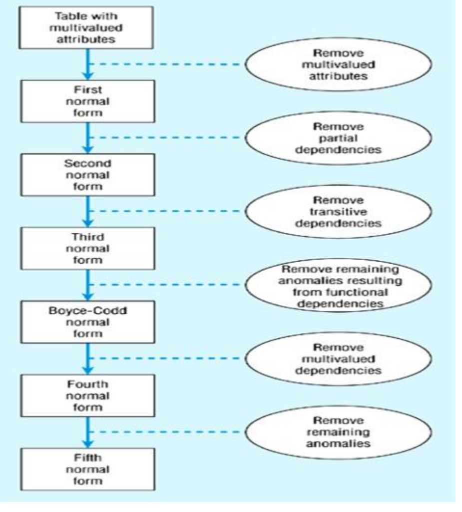
EmployeeTasks(EID, Specialty, ToolID)

- Business rules.
- Each employee has many specialties.
- Each employee has many tools.
- Tools and specialties are unrelated.

EmployeeSpecialty(EID, Specialty)
EmployeeTools(EID, ToolID)

46

Steps in normalization



47

Thank you for your attention!