Chapter 6: Database Normalization

Outline

- Key of a relational schema
- Normalization
- Normal Forms

Keys

- **Key/candidate key:**
  - We say a set of **one or more attributes** {A1,A2 , ... , An} is a key/candidate key for a relation R if:
    - Those attributes functionally determine all other attributes of the relation.
    - No proper subset of { A1,A2 , . . . , An} functionally determines all other attributes of R; i.e., a key must be minimal.
- **Superkey**: A set of attributes that contains a **key** is called a superkey.

The algorithm for computing a candidate key of R.

- **Input**: a relation R = { A1, A2, ..., An}, and F, a set of functional dependencies.
- **Output**: K, a candidate key of R.
- **Algorithm**

```
set K ← R;
for each attribute A ∈ K
 {
   compute {K – A}+ with respect to F
   if {K – A}+ contains all attributes of R then
             set K = {K – A}
 }
 return K
```

The algorithm for computing a candidate key of R.

- **Example**:
  - R = {A, B, C, D, E, F, G}
  - F= {B → A, D → C, D → BE, DF → G}
  - **K= ?**

- B1:

  K = ABCDEFG.

- B2:
  - Lặp 1: $(BCDEFG)_F^+ = BCDEFGA \Rightarrow K = BCDEFG$.
  - Lặp 2: $(CDEFG)_F^+ = CDEFGBA \Rightarrow K = CDEFG$.
  - Lặp 3: $(DEFG)_F^+ = DEFGCBA \Rightarrow K = DEFG$.
  - Lặp 4: $(EFG)_F^+ = EFG$.
  - Lặp 5: $(DFG)_F^+ = DFGCBEA \Rightarrow K = DFG$.
  - Lặp 6: $(DG)_F^+ = DGCBEA$.
  - Lặp 7: $(DF)_F^+ = DFCBEAG \Rightarrow K = DF$.

- B3:

  Khóa là K = DF.

- B1:

  K = ABCDEFG.

- B2:
  - Lặp 1: $(BCDEFG)_F^+ = BCDEFGA \Rightarrow K = BCDEFG$.
  - Lặp 2: $(CDEFG)_F^+ = CDEFGBA \Rightarrow K = CDEFG$.
  - Lặp 3: $(DEFG)_F^+ = DEFGCBA \Rightarrow K = DEFG$.
  - Lặp 4: $(EFG)_F^+ = EFG$.
  - Lặp 5: $(DFG)_F^+ = DFGCBEA \Rightarrow K = DFG$.
  - Lặp 6: $(DG)_F^+ = DGCBEA$.
  - Lặp 7: $(DF)_F^+ = DFCBEAG \Rightarrow K = DF$.
- B3:

  Khóa là K = DF.

- R(A, B, C, D, E),
- F= {AB → C, B → D, CD → E}

Normalization

- What is normalization?
    - Basically, it's the process of efficiently organizing data in a database.
- Normalization theory is based on the observation that relations with certain properties are *more effective in inserting, updating and deleting data than other sets of relations containing the same data*
- Normalization is a multi-step process beginning with an "unnormalized" relation
    - This is the process which allows you to winnow out redundant data within your database.
    - This involves restructuring the tables to successively meeting higher forms of Normalization.

Normalization

- Normalization theory is based on the concepts of **normal forms**.
- A relational table is said to be in a particular normal form if it satisfied a certain set of constraints.
- A properly normalized database should have the following characteristics
  - Scalar values in each fields
  - Absence of redundancy.
  - Minimal use of null values.
  - Minimal loss of information.

Normalization

- *The goal of normalization is to create a set of relational tables that are **free of redundant** data and that can be **consistently** and **correctly modified or updated**.*

Definitions

- ***Partial FD*** – *when an non-key attribute is determined by a part, but not the whole, of a* ***COMPOSITE*** *primary key.*
- **Example**: Key= {Cust_ID, Order_ID}
  - Cust_ID → Name is partial FD

Partial Dependency

CUSTOMER

Cust_ID	Name	Order_ID
101	AT&T	1234
101	AT&T	156
125	Cisco	1250

Definitions

- ***Transitive FD –*** *when a non-key attribute determines another non-key attribute.*
- Example: Key= {Emp_ID}
  - Dept_ID → Dept_Name is transitive FD

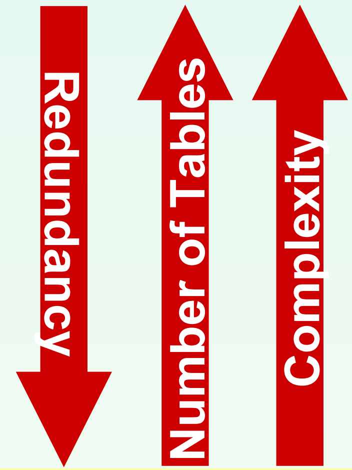
Transitive Dependency

EMPLOYEE

<u>Emp_ID</u>	F_Name	L_Name	Dept_ID	Dept_Name
111	Mary	Jones	1	Acct
122	Sarah	Smith	2	Mktg

Normal Forms

- Levels of normalization based on the amount of redundancy in the database.
- Various levels of normalization are:
  - First Normal Form (1NF)
  - Second Normal Form (2NF)
  - Third Normal Form (3NF)
  - Boyce-Codd Normal Form (BCNF)



Most databases should be 3NF or BCNF in order to avoid the database anomalies.

Each higher level is a subset of the lower level

FDs and keys are used to define normal forms for relations

Normal Forms

- Unnormalized – There are multivalued attributes or repeating groups
- 1 NF – No multivalued attributes or repeating groups.
- 2 NF – 1 NF plus no partial dependencies
- 3 NF – 2 NF plus no transitive dependencies
- BCNF – 3 NF plus all determinants are superkeys

First Normal Form (1NF)

- A table is considered to be in 1NF if **all the fields contain only atomic or scalar values** (as opposed to list of values or there are no sets of values within a column).
  - No repeating groups
  - A column or set of columns is called a Candidate Key when its values can uniquely identify the row in the relation.
- Note: ***All relations* are in 1st Normal Form**

First Normal Form (1NF)

- **Example:** Table with multivalued attributes, not in 1st normal form

Order_ID	Order_ Date	Customer_ ID	Customer_ Name	Customer_ Address	Product_ID	Product_ Description	Product_ Finish	Unit_ Price	Ordered_ Quantity
1006	10/24/2004	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
					5	Writer's Desk	Cherry	325.00	2
					4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2004	6	Furniture Gallery	Boulder, CO	11	4–Dr Dresser	Oak	500.00	4
					4	Entertainment Center	Natural Maple	650.00	3

First Normal Form (1NF)

- **Example:** Table with no multivalued attributes and unique rows, in 1st normal form

INVOICE relation (1NF) (Pine Valley Furniture Company)

Order_ID	Order_ Date	Customer_ ID	Customer_ Name	Customer_ Address	Product_ID	Product_ Description	Product_ Finish	Unit_ Price	Ordered_ Quantity
1006	10/24/2004	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
1006	10/24/2004	2	Value Furniture	Plano, TX	5	Writer's Desk	Cherry	325.00	2
1006	10/24/2004	2	Value Furniture	Plano, TX	4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2004	6	Furniture Gallery	Boulder, CO	11	4–Dr Dresser	Oak	500.00	4
1007	10/25/2004	6	Furniture Gallery	Boulder, CO	4	Entertainment Center	Natural Maple	650.00	3

Note: this is relation, but not a well-structured one

Second Normal Form (2NF)

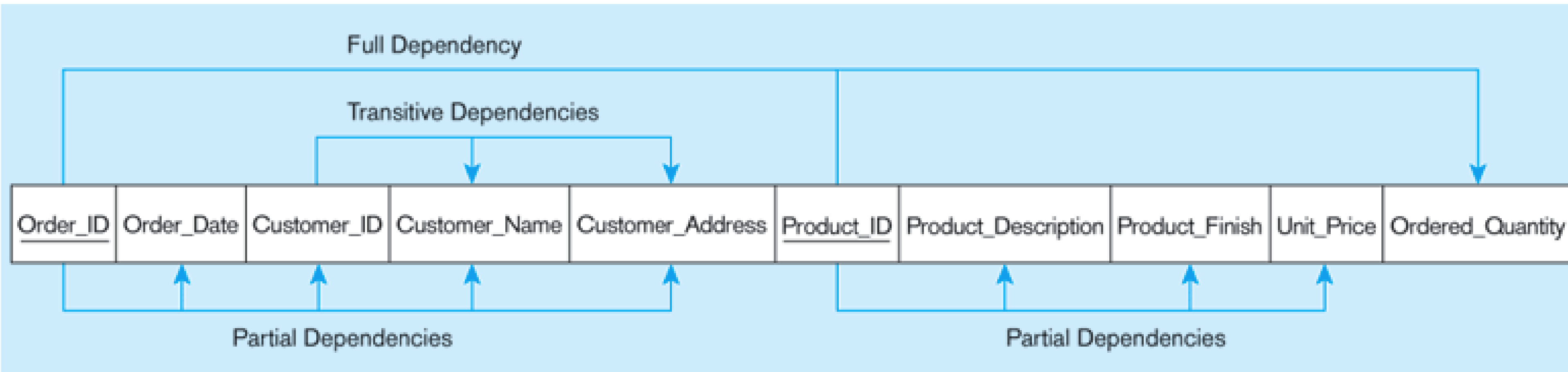
- Where the 1NF deals with atomicity of data, the Second Normal Form (or 2NF) deals with relationships between composite key columns and non-key columns:
  - Meet all the requirements of the first normal form.
  - Any non-key columns must depend on the entire primary key. In the case of a composite primary key, this means that a non-key column cannot depend on only part of the composite key.
  - Create relationships between these new tables and their predecessors through the use of foreign keys.
  - A relation R is in 2nf if every non-primary attribute A in R is fully Functionally dependent on the primary key.

Second Normal Form (2NF)

- 1NF PLUS ***every non-key attribute is fully functionally dependent on the ENTIRE primary key***
  - Every non-key attribute must be defined by the entire key, not by only part of the key
  - No partial functional dependencies

Second Normal Form (2NF)

Functional dependency diagram for INVOICE



Order_ID → Order_Date, Customer_ID, Customer_Name, Customer_Address

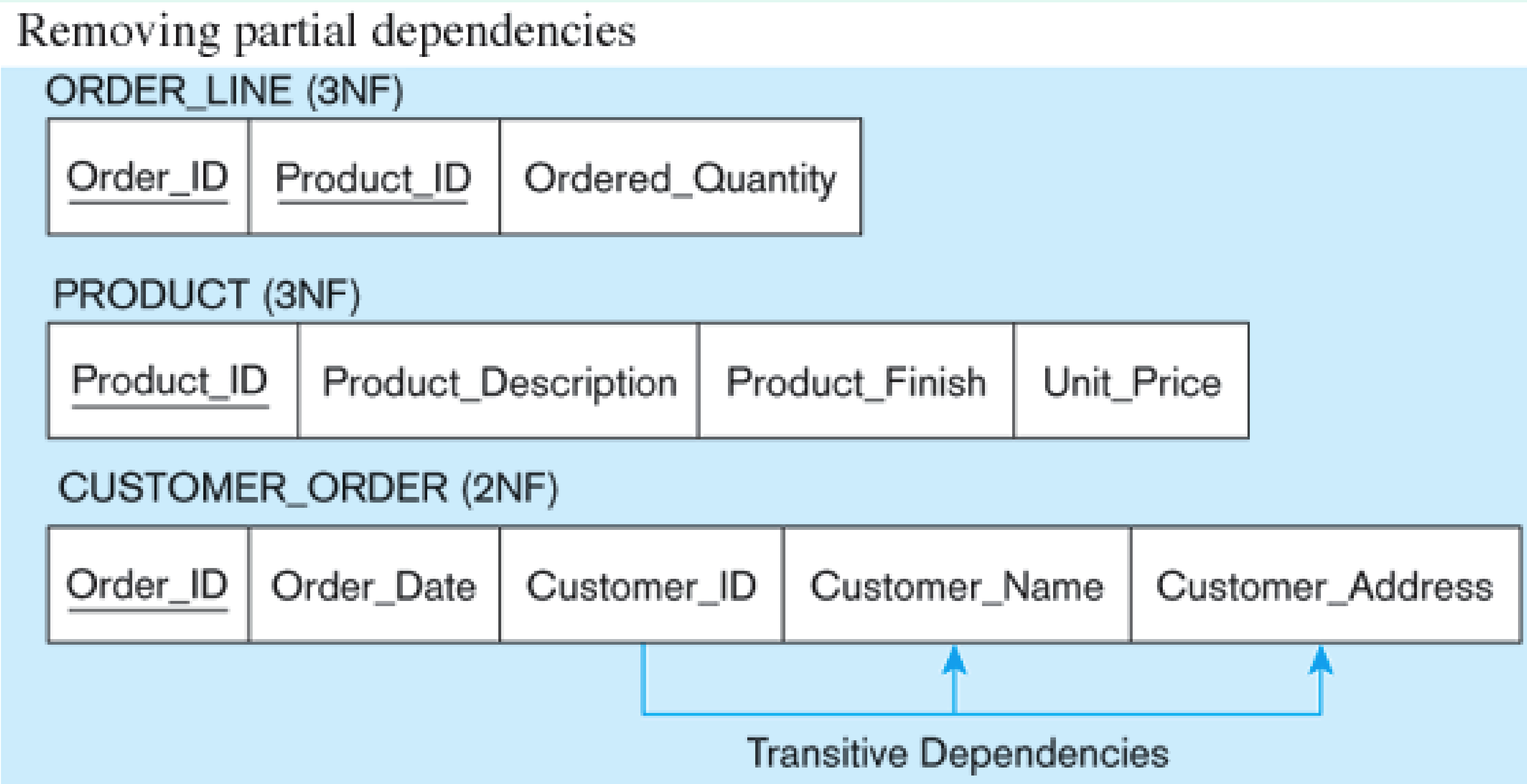
Customer_ID → Customer_Name, Customer_Address

Product_ID → Product_Description, Product_Finish, Unit_Price

Order_ID, Product_ID → Order_Quantity

Therefore, NOT in 2nd Normal Form

Second Normal Form (2NF)



Partial Dependencies are removed, but there are still transitive dependencies

Third Normal Form (3NF)

Third Normal Form (3NF)

- **2NF PLUS *no transitive dependencies* (functional dependencies on non-primary-key attributes)**
- Note: this is called transitive, because the primary key is a determinant for another attribute, which in turn is a determinant for a third
- Solution: non-key determinant with transitive dependencies go into a new table; non-key determinant becomes primary key in the new table and stays as foreign key in the old table

Third Normal Form (3NF)

- Customer_Order (2NF)

Order_ID	Order_Date	Customer_ID	Customer_Name	Customer_Address

Transitive Dependencies

Removing transitive dependencies

ORDER (3NF)

Order_ID	Order_Date	Customer_ID

CUSTOMER (3NF)

Customer_ID	Customer_Name	Customer_Address

Boyce-Codd Normal Form (BCNF)

- BCNF eliminates all redundancy that can be discovered by functional dependencies
- Most 3NF relations are also BCNF relations.
- A 3NF relation is NOT in BCNF if:
  - Candidate keys in the relation are composite keys (they are not single attributes)
  - There is more than one candidate key in the relation, and
  - The keys are not disjoint, that is, some attributes in the keys are common

Boyce-Codd Normal Form (BCNF)

- **Definition**
  - A relation schema *R* is in BCNF with respect to a set *F* if:
    - For all functional dependencies of *F* of the form $\alpha \rightarrow \beta$, where $\alpha \subseteq R$ and $\beta \subseteq R$
      - $\alpha \rightarrow \beta$ is a trivial functional dependency ($\beta \subseteq \alpha$)
      - α is a superkey for schema R
  - *A database design is in BCNF if each member of the set of relational schemas that constitute the design is in BCNF*

Boyce-Codd Normal Form (BCNF)

- **Rule for schema not in BCNF**
  - Let ***R*** be a schema **not in BCNF**, then there is at least one nontrivial functional dependency $\alpha \rightarrow \beta$ such that α is not a superkey
  - **Example of not BCNF:**
    - *bor_loan = (customer_id, loan_number, amount)*
    - *FD: loan_number → amount*
    - *But **loan_number** is not a superkey*

BCNF Decomposition

- The definition of BCNF can be used to directly test if a relationship is in BCNF
- If a relation is not in BCNF it can be decomposed to create relations that are in BCNF
- Example:
  - *borrower = (customer_id, load_number)*

    → Is BCNF because no nontrivial functional dependency hold onto it
  - *loan = (loan_number, amount)*

    → *Has **one nontrivial functional dependency** that holds,*

    loan_number→amount,

    But **loan_number** is a **superkey** so **loan is in BCNF**

3NF vs BCNF

- BCNF requires that all nontrivial dependencies be of the form $\alpha \rightarrow \beta$, where $\boldsymbol{\alpha}$ **is a superkey**
- **3NF** relaxes this constraint a little bit by **allowing nontrivial functional dependencies**

BCNF Decomposition

- If ***R*** is not in BCNF, we can decompose *R* into a collection of BCNF schemas:

  $R_1 , R_2, \ldots, R_n$

- **Example:**

***lending* = (branch_name, branch_city, assets, customer_name, loan_number, amount)**

FDs:

branch_name → assets branch_city

loan_number → amount branch_name

Candidate key : *{loan_number, customer_name}*

branch_name *is not superkey* so **lending** is not BCNF

BCNF Decomposition

- So we replace *lending* by:

  branch = *(branch_name, branch_city, assets)*

  loan_info = *(branch_name, customer_name, loan_number, amount)*

- The only nontrivial functional dependencies that hold on ***branch*** include ***branch_name*** on the left side of the arrow.

  => Since ***branch_name*** is a key for ***branch***, the relation ***branch*** **is in BCNF**

BCNF Decomposition

- **For *loan_info***
  - The functional dependency
    loan_number → *amount branch_name*
    holds on *loan_info*
  - But *loan_number* is not a key for *loan_info*
    → So we replace *loan_info* by
    loanb = *(loan_number, branch_name, amount)*
    borrower = *(customer_name, loan_number)*
    → *loanb* and *borrower* are in BCNF

Advantage / Disadvantage of 3NF

- **Advantage of 3NF**: it is always possible to obtain a 3NF design without sacrificing losslessness or dependency preservation
- **Disadvantage of 3NF**: we may have to use null values to represent some of the possible meaningful relationships among data items, and there is the problem of repetition of information

Conclusion

- Goals of database design with functional dependencies are
  1) BCNF
  2) Losslessness
  3) Dependency preservation
- Not possible to get all 3, we have to choose between BCNF or dependency preservation

Example

- R(A, B, C, D, E), F= {AB → C, B → D, CD → E}
- R is in BCNF ?
- K={AB} => R is in 1NF, B → D is partial FD

R(A, B, C, D, E)
F= {AB → C, B → D, CD → E}

CD → E

R1(C, D, E)
F1= {CD → E}
K1={CD}
R1 is in BCNF

R'(A, B, C, D)
F'= {AB → C, B → D, CD → ∅}
K'={AB}

B → D

R2(B,D)
F2= {B → D }
K2={B}
=> R2 is in BCNF

R3(A, B, C)
F3= {AB → C}
K'={AB}
=> R3 is in BCNF

=> R(A, B, C, D, E)=R1(C, D, E) * R2(B,D) * R3(A, B, C)

Bài tập

- R(A, B, C, D, E), F= {AB → C, B → D, CD → E}
- R is in BCNF ?
- K={AB} => R is in 1NF, B → D is partial FD

R(A, B, C, D, E)
F= {AB → C, B → D, CD → E}

B → D

R1(B, D)
F1= {B → D}
K1={B}
R1 in BCNF

R'(A, B, C, E)
F'= {AB → C, CB → E }
K'={AB}

CB → E

R2(CB,E)
F2= {CB → E }
K2={CB}
=> R2 in BCNF

R3(A, B, C)
F3= {AB → C}
K3={AB}
=> R3 in BCNF

=>R(A, B, C, D, E)=R1(B, D) * R2(CB,E) * R3(A, B, C)