# FUNDAMENTALS OF DATABASES

## Functional Dependencies & Normal Forms

NGUYEN Hoang Ha

Email: nguyen-hoang.ha@usth.edu.vn

# FUNCTIONAL DEPENDENCIES

# Objectives

- Understand what are functional dependencies
- Understand the logic behind the FDs concept
- Understand the rules about FDs
- Understand what are keys, super-keys
- Understand what are closure-sets and how to determine them

# Functional Dependency Definition

- A **functional dependency** (FD) is a constraint between two sets of attributes in a Relation from a database
- Given a relation *R*, a set of attributes *X* in *R* is said to **functionally determine** another attribute *Y*, also in *R*, (written *X* → *Y*) if and only if each *X* value is associated with precisely one *Y* value
  - Say "X → Y holds in R."
  - Convention: …, X, Y, Z represent sets of attributes; A, B, C,… represent single attributes.
  - Convention: no set formers in sets of attributes, just ABC, rather than {A,B,C }.

# Functional Dependency Definition

- Customarily we call *X* the *determinant set* and *Y* the *dependent attribute*

- A functional dependency FD: is called **trivial** if Y is a subset of X
- A Functional Dependency (FD) X → Y on a relation R is a statement of the form:

  "If 2 tuples of R agree on attribute X, then they must also agree on Y"

# Splitting Right Sides of FD's

- $X \rightarrow A_1A_2\ldots A_n$ holds for $R$ exactly when each of $X \rightarrow A_1$, $X \rightarrow A_2, \ldots, X \rightarrow A_n$ hold for $R$.
- Example: $A \rightarrow BC$ is equivalent to $A \rightarrow B$ and $A \rightarrow C$.
- There is no splitting rule for left sides.
- We'll generally express FD's with singleton right sides.



6

# FD Example

Drinkers(name, addr, beersLiked, manf, favBeer)

- Reasonable FD's to assert:
  1. name → addr favBeer
     - Note this FD is the same as name → addr and name → favBeer.
  2. beersLiked → manf

# FD Example

- Easy to see that: the following FD is true
  (title, year) → (length, genre, studioName)

- **Exercise**: How about this FD (TRUE or FALSE)?
  (title, year) → **(starName)**

| *title* | *year* | *length* | *genre* | *studioName* | *starName* |
|---|---|---|---|---|---|
| Star Wars | 1977 | 124 | SciFi | Fox | Carrie Fisher |
| Star Wars | 1977 | 124 | SciFi | Fox | Mark Hamill |
| Star Wars | 1977 | 124 | SciFi | Fox | Harrison Ford |
| Gone With The Wind | 1939 | 231 | drama | MGM | Vivien Leigh |
| Wayne's World | 1992 | 95 | comedy | Paramount | Dana Carvey |
| Wayne's World | 1992 | 95 | comedy | Paramount | Mike Meyers |

# Properties of functional dependencies

Given that $X$, $Y$, and $Z$ are sets of attributes in a relation $R$, one can derive several properties of functional dependencies. Among the most important are Armstrong's axiom, which are used in database normalization:

- **Subset Property** (Axiom of Reflexivity): If $Y$ is a subset of $X$, then $X \rightarrow Y$
- **Augmentation** (Axiom of Augmentation): If $X \rightarrow Y$, then $XZ \rightarrow YZ$
- **Transitivity** (Axiom of Transitivity): If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$

**From these rules, we can derive these secondary rules:**

- **Union**: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
- **Decomposition**: If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$
- **Pseudo transitivity**: If $X \rightarrow Y$ and $YZ \rightarrow W$, then $XZ \rightarrow W$
- **Accumulation**: If $X \rightarrow YZ$ and $Z \rightarrow V$, then $X \rightarrow YZV$
- **Extension**: If $X \rightarrow Y$ and $W \rightarrow Z$, then $WX \rightarrow YZ$

# Keys

- A set of one or more attributes $\{A_1, A_2, .., A_n\}$ is called a **key** of the relation R if:

**1. Those attributes functionally determine all other attributes of R.**
This means that: It is impossible for 2 tuples of R to agree on all of $\{A_1, A_2, .., A_n\}$

**2. No proper subset of $\{A_1, A_2, .., A_n\}$ functionally determines all other attributes of R**
This means that: A Key must be minimal

# Example: Keys

- **Exercise 1**:
  (title, year) forms a key?

- **Exercise 2**:
  (title, year, starName) forms a key ?

| *title* | *year* | *length* | *genre* | *studioName* | *starName* |
|---|---|---|---|---|---|
| Star Wars | 1977 | 124 | SciFi | Fox | Carrie Fisher |
| Star Wars | 1977 | 124 | SciFi | Fox | Mark Hamill |
| Star Wars | 1977 | 124 | SciFi | Fox | Harrison Ford |
| Gone With The Wind | 1939 | 231 | drama | MGM | Vivien Leigh |
| Wayne's World | 1992 | 95 | comedy | Paramount | Dana Carvey |
| Wayne's World | 1992 | 95 | comedy | Paramount | Mike Meyers |

# Super-key

- A set of attributes that contains a key is called a **Super-key**

- A **superkey** is a set of attributes of a relation whose values can be used to uniquely identify a row

# Example: Super-key

- There are many Super-key in the above schema:

  (title, year, starName)
  (title, year, starName, length, studioName)

- **Exercise:** Can (title, year, starName, length, studioName) form a key?

| *title* | *year* | *length* | *genre* | *studioName* | *starName* |
|---|---|---|---|---|---|
| Star Wars | 1977 | 124 | SciFi | Fox | Carrie Fisher |
| Star Wars | 1977 | 124 | SciFi | Fox | Mark Hamill |
| Star Wars | 1977 | 124 | SciFi | Fox | Harrison Ford |
| Gone With The Wind | 1939 | 231 | drama | MGM | Vivien Leigh |
| Wayne's World | 1992 | 95 | comedy | Paramount | Dana Carvey |
| Wayne's World | 1992 | 95 | comedy | Paramount | Mike Meyers |

# Closure sets

- Suppose: X is a set of attributes, and S is a set of FDs.
- The closure of X under S is the set of attributes Y such that can be determined from X using S
- Closure of a set of attributes X or {A1, ..., An} is denoted X+ or {A1, A2, ..., An}+ respectively

# Example: Closure Set

- $R(A, B, C, D, E, F)$
- $S = \{A \rightarrow C, A \rightarrow D, D \rightarrow E, E \rightarrow F\}$
- Easy to see that:
  $\{A\}+ = \{A, C, D, E, F\}$
- Naïve idea to find closure set
  - Basis: $Y^+ = Y$.
  - Induction: Look for an FD's left side $X$ that is a subset of the current $Y^+$. If the FD is $X$ -> $A$, add $A$ to $Y^+$.

# Closure set Finding

Algorithm (*courtesy* [DBSC] Fig. 7.9 [p. 281])

- **INPUT**: A set of attributes A={A1,..,An} and a set of FD's F
  **OUTPUT**: The closure result = $A^+$

- Step 1: Split all FDs in F such that: each FD has a single attribute on the right;

  Step 2: result = A;

  Step 3: **while** (changes to result)
  **for each** (FD $X \rightarrow Y$ of F) **do**
  **if** (X is the sub-set of result)
  result = result U Y;

# Closure sets

**Algorithm explain:**

- Step 1: start with initial set of attributes $X$
- Step 2: identify FD's $A \rightarrow B$ where $A$ is a sub-set of $X$, but $B \notin X$
- Step 3: add $B$ to $X$
- Step 4: repeat until no more attributes can be added to the closure, or, in other words, when you reach a fixpoint

# Exercise

- $R(A, B, C, D, E, F)$
- $S = \{AB \rightarrow C, BC \rightarrow AD, D \rightarrow E, CF \rightarrow B\}$
- compute $\{AB\}+$
- Answer: {AB}+ = {ABCDE}

# Exercise

- $R(A, B, C, D)$
- $S = \{BC \rightarrow D, D \rightarrow A, A \rightarrow B\}$
- What are all the keys of R?
- What are all the super-keys for R that are not keys?

# Exercise

- $R(A, B, C, D)$
- $S = \{A \rightarrow B, A \rightarrow C, C \rightarrow D\}$
- What are all the keys of R?
- What are all the super-keys for R that are not keys?
- Answer:
  - {A}+ = {ABCD} → A is a super-key. A is minimal → A is a key
  - {C}+= {CD} → C is not a super-key
  - {B}+= {B} ➔ do not need to check sets whose elements existing on the right side of FDs.
  - All super sets of {A} is a super-key, e.g: {AB}, {AC}, {AD}….

# Exercise

- $R(A, B, C, D)$
- $S = \{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A\}$
- What are all the keys of R?
- What are all the super-keys for R that are not keys?
- Answer:
  - {A}+={ABCD}
  - {B}+={ABCD}
  - {C}+={ABCD}
  - {D}+={ABCD}
  - → Keys: {A}, {B}, {C}, {D}
  - → Super-keys: add any attribute to the keys.

# Exercise

- $R(A, B, C, D)$
- $S = \{AD \rightarrow B, AB \rightarrow C, BC \rightarrow D, CD \rightarrow A\}$
- What are all the keys of R?
- What are all the super-keys for R that are not keys?
- Answer:
  - {AD}+={ADBC}
  - {AB}+={ABCD}
  - {BC}+={ABCD}
  - {CD}+={ABCD}
  - → Keys: {AD}, {AB}, {BC}, {DD}
  - → Super-keys: add any attribute to the keys.

# NORMAL FORMS

# Anomalies introduction

- Careless selection of a relational database schema can lead to redundancy and anomalies
- So in this session we shall tackle the problems of relational database designing
- Problems such as redundancy that occur when we try to cram too much into a single relation are called *anomalies*

| *title* | *year* | *length* | *genre* | *studioName* | *starName* |
|---|---|---|---|---|---|
| Star Wars | 1977 | 124 | SciFi | Fox | Carrie Fisher |
| Star Wars | 1977 | 124 | SciFi | Fox | Mark Hamill |
| Star Wars | 1977 | 124 | SciFi | Fox | Harrison Ford |
| Gone With The Wind | 1939 | 231 | drama | MGM | Vivien Leigh |
| Wayne's World | 1992 | 95 | comedy | Paramount | Dana Carvey |
| Wayne's World | 1992 | 95 | comedy | Paramount | Mike Meyers |

- **The principal kinds of anomalies that we encounter are**:
- ***Redundancy***: information maybe repeated unnecessarily in several tuples (**exp**: the length and genre)
- ***Update Anomalies***: We may change information in one tuple but leave the same information unchanged in another (e.g.: if we found that *Star Wars* is 125 minutes long, we may change the length in the first tuple but not in the second and third tuples)
- ***Deletion Anomalies***: If a set of values becomes empty, we may lose other information as a side effect (e.g.: if we delete "Fox" from as the studioName of "Star Wars", then we may lost the "Fox" from the list of studio)

# Decomposition

- The accepted way to eliminate anomalies is the *decomposition* of relations
- Decomposition of a relation R involves splitting the attributes of R to make the schemas of 2 new relations
- **Definition**: Given a relation $R(A_1,..,A_n)$, we say R is decomposed into $S(B_1,..,B_m)$ and $T(C_1,..,C_k)$ if:
  + $\{A_1,..,A_n\} = \{B_1,..,B_m\} \cup \{C_1,..,C_k\}$
  + $S = \prod_{B1,..Bm}(R)$
  + $T = \prod_{C1,..,Ck}(R)$

# Example: Decomposition

| *title* | *year* | *length* | *genre* | *studioName* | *starName* |
|---|---|---|---|---|---|
| Star Wars | 1977 | 124 | SciFi | Fox | Carrie Fisher |
| Star Wars | 1977 | 124 | SciFi | Fox | Mark Hamill |
| Star Wars | 1977 | 124 | SciFi | Fox | Harrison Ford |
| Gone With The Wind | 1939 | 231 | drama | MGM | Vivien Leigh |
| Wayne's World | 1992 | 95 | comedy | Paramount | Dana Carvey |
| Wayne's World | 1992 | 95 | comedy | Paramount | Mike Meyers |

| *title* | *year* | *length* | *genre* | *studioName* |
|---|---|---|---|---|
| Star Wars | 1977 | 124 | SciFi | Fox |
| Gone With The Wind | 1939 | 231 | drama | MGM |
| Wayne's World | 1992 | 95 | comedy | Paramount |

| *title* | *year* | *starName* |
|---|---|---|
| Star Wars | 1977 | Carrie Fisher |
| Star Wars | 1977 | Mark Hamill |
| Star Wars | 1977 | Harrison Ford |
| Gone With | 1939 | Vivien Leigh |
| Wayne's W | 1992 | Dana Carvey |
| Wayne's W | 1992 | Mike Meyers |

# Discuss

| *title* | *year* | *length* | *genre* | *studioName* |
|---|---|---|---|---|
| Star Wars | 1977 | 124 | SciFi | Fox |
| Gone With The Wind | 1939 | 231 | drama | MGM |
| Wayne's World | 1992 | 95 | comedy | Paramount |

| *title* | *year* | *starName* |
|---|---|---|
| Star Wars | 1977 | Carrie Fisher |
| Star Wars | 1977 | Mark Hamill |
| Star Wars | 1977 | Harrison Ford |
| Gone With | 1939 | Vivien Leigh |
| Wayne's W | 1992 | Dana Carvey |
| Wayne's W | 1992 | Mike Meyers |

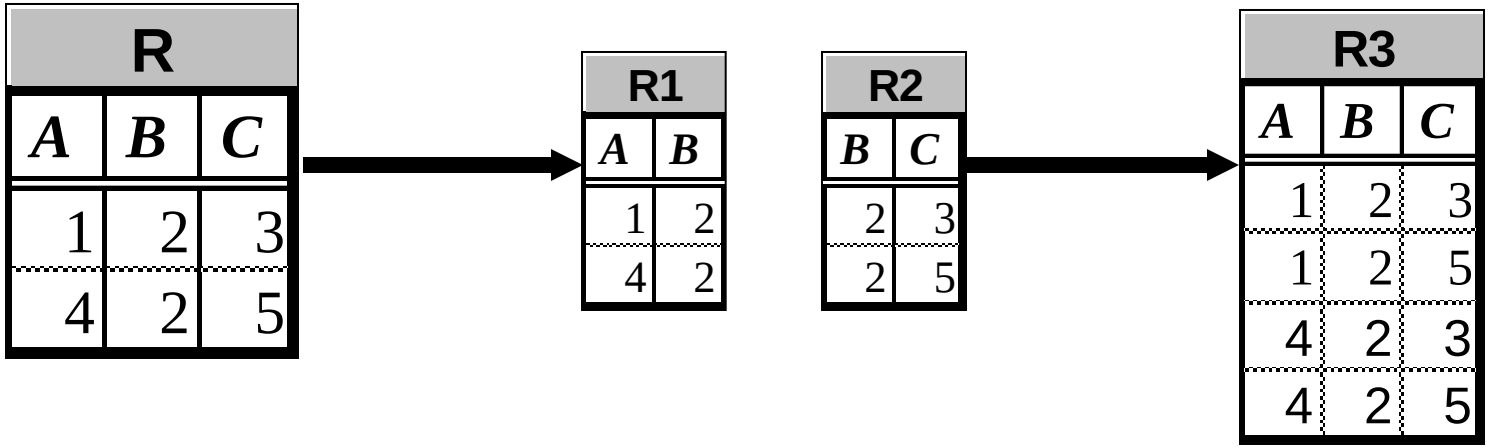
- The redundancy is eliminated (the length of each film appears only once)
- The risk of an update anomaly is gone (we only have to change the length of *Star Wars* in one tuple)
- The risk of a deletion anomaly is gone (if we delete all the stars for *Gone with the wind*, that deletion makes the movie disappear from the right but still be found in the left)

# Decomposition: The Good, Bad and Ugly

- We observed that before we decomposing a relation schema will eliminate anomalies; That's the "Good"

- But, decomposition can also have some bad:
  - Maybe we can't recovery the original information; OR
  - After reconstruction, the FDs maybe not hold

# Loss of information after decomposition




- Suppose we have R(A,B,C) but neither of the FD's B->A nor B->C holds.
- R is decomposed into R1 and R2 as above
- When we try to re-construct R by Natural Join of R1 and R2, we have: R3 = R1 X R2 (but R3 <> R1 => We lost information)

# 1NF

- First Normal Form
  - Attributes are single-valued, atomic
  - Tuples are unique
- Case study: Garment management
  - Is Products (item, colors, price, tax) in 1NF?

| item | colors | price | tax |
|---|---|---|---|
| T-shirt | red, blue | 12.00 | 0.60 |
| polo | red, yellow | 12.00 | 0.60 |
| T-shirt | red, blue | 12.00 | 0.60 |
| sweatshirt | blue, black | 25.00 | 1.25 |

→ Answer: No

# 1NF normalized

| item | colors | price | tax |
|---|---|---|---|
| T-shirt | red, blue | 12.00 | 0.60 |
| polo | red, yellow | 12.00 | 0.60 |
| T-shirt | red, blue | 12.00 | 0.60 |
| sweatshirt | blue, black | 25.00 | 1.25 |

| item | color | price | tax |
|---|---|---|---|
| T-shirt | red | 12.00 | 0.60 |
| T-shirt | blue | 12.00 | 0.60 |
| polo | red | 12.00 | 0.60 |
| polo | yellow | 12.00 | 0.60 |
| sweatshirt | blue | 25.00 | 1.25 |
| sweatshirt | black | 25.00 | 1.25 |

# Example: 1NF converting

| Name |
| --- |
| Mike Hillyer |



| ID | First_Name | Last_Name |
| --- | --- | --- |
| 1 | Mike | Hillyer |

- The Name attribute is not atomic, so it must be divided into First_Name and Last_Name

# How to recognize the Design is not in 1-NF ?

### *Data that Contains Non-Alphabetic Characters*

By non-alphabetic characters we mean commas, brackets, parentheses, pipe characters, etc. These act as warning signs that we are dealing with a multi-valued field. However, be careful not to go to far. For instance, if we were designing a solution to hold a block of text, we have probably normalized too much if we have a word entity, a sentence entity, and a paragraph entity. This clue is more applicable to entities that have delimited lists.

### *Fieldnames with Numbers at the End*

As we have noted, an obvious example would be finding entities with *child1*, *child2*, etc., attributes or, my favorite, *UserDefined1*, *UserDefined2*, etc. These are very blatantly wrong.

This is a very common holdover from the days of flat file databases. Multi-table data access was costly, so they put many fields in a single table. This is very detrimental for relational database systems.

# 2NF

- Non-primary key attributes depend on all components of PK
  - PK is a single attribute → guaranteed
- Our case:
  - {item} → {price} , {price} → {tax} so violate 2NF criteria

| item | color | price | tax |
|---|---|---|---|
| T-shirt | red | 12.00 | 0.60 |
| T-shirt | blue | 12.00 | 0.60 |
| polo | red | 12.00 | 0.60 |
| polo | yellow | 12.00 | 0.60 |
| sweatshirt | blue | 25.00 | 1.25 |
| sweatshirt | black | 25.00 | 1.25 |

# 2NF normalized

| item | color | price | tax |
|---|---|---|---|
| T-shirt | red | 12.00 | 0.60 |
| T-shirt | blue | 12.00 | 0.60 |
| polo | red | 12.00 | 0.60 |
| polo | yellow | 12.00 | 0.60 |
| sweatshirt | blue | 25.00 | 1.25 |
| sweatshirt | black | 25.00 | 1.25 |

| item | color |
|---|---|
| T-shirt | red |
| T-shirt | blue |
| polo | red |
| polo | yellow |
| sweatshirt | blue |
| sweatshirt | black |

| item | price | tax |
|---|---|---|
| T-shirt | 12.00 | 0.60 |
| polo | 12.00 | 0.60 |
| sweatshirt | 25.00 | 1.25 |

# 3NF

- No non-key attributes depends on others
- Example
  - Products (<u>item</u>, price, tax)

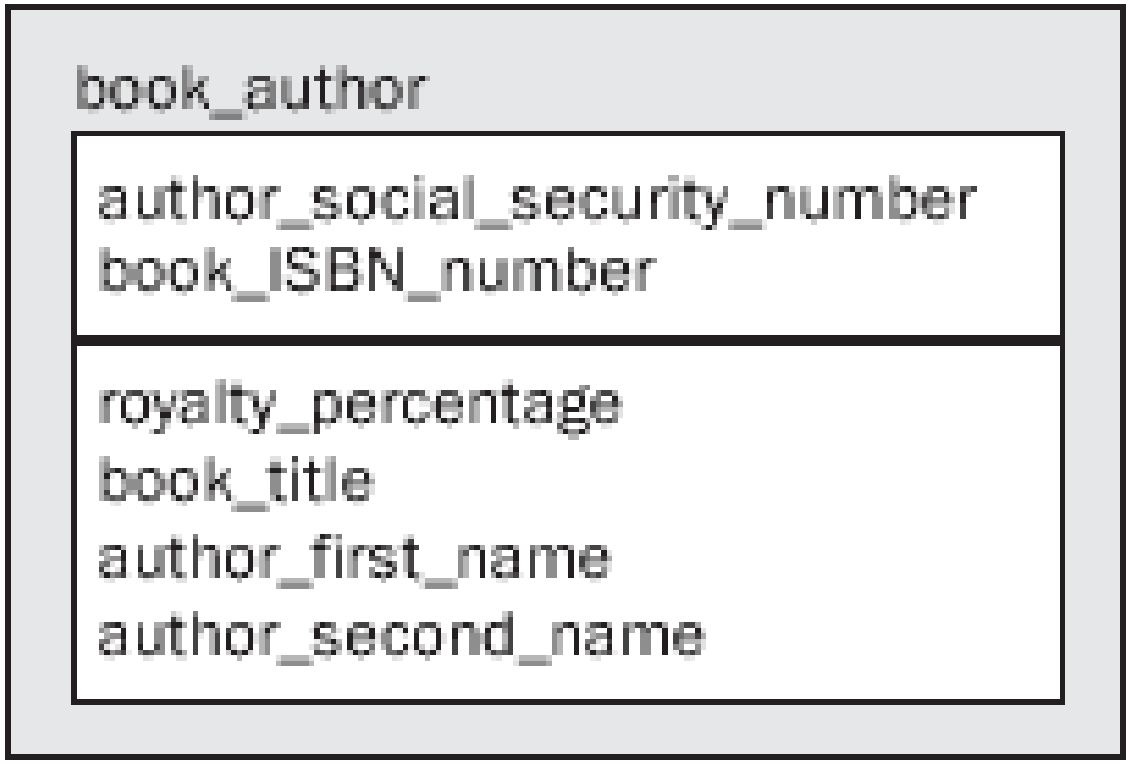
| item | price | tax |
|---|---|---|
| T-shirt | 12.00 | 0.60 |
| polo | 12.00 | 0.60 |
| sweatshirt | 25.00 | 1.25 |

  - {Item} → {price} , {price} → {tax}

| item | price |
|---|---|
| T-shirt | 12.00 |
| polo | 12.00 |
| sweatshirt | 25.00 |

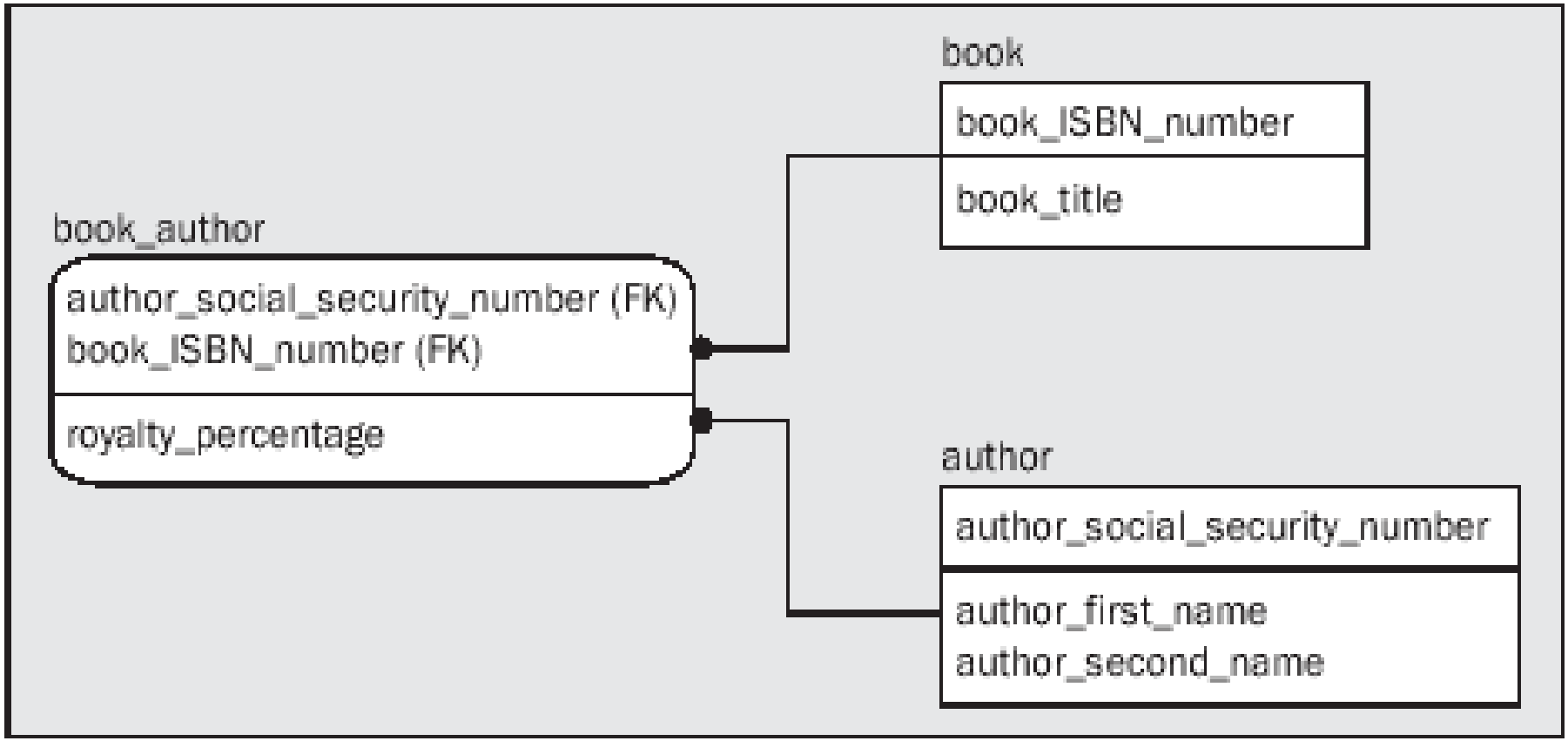
| price | tax |
|---|---|
| 12.00 | 0.60 |
| 25.00 | 1.25 |

# Exercise 4 – Make it 2NF

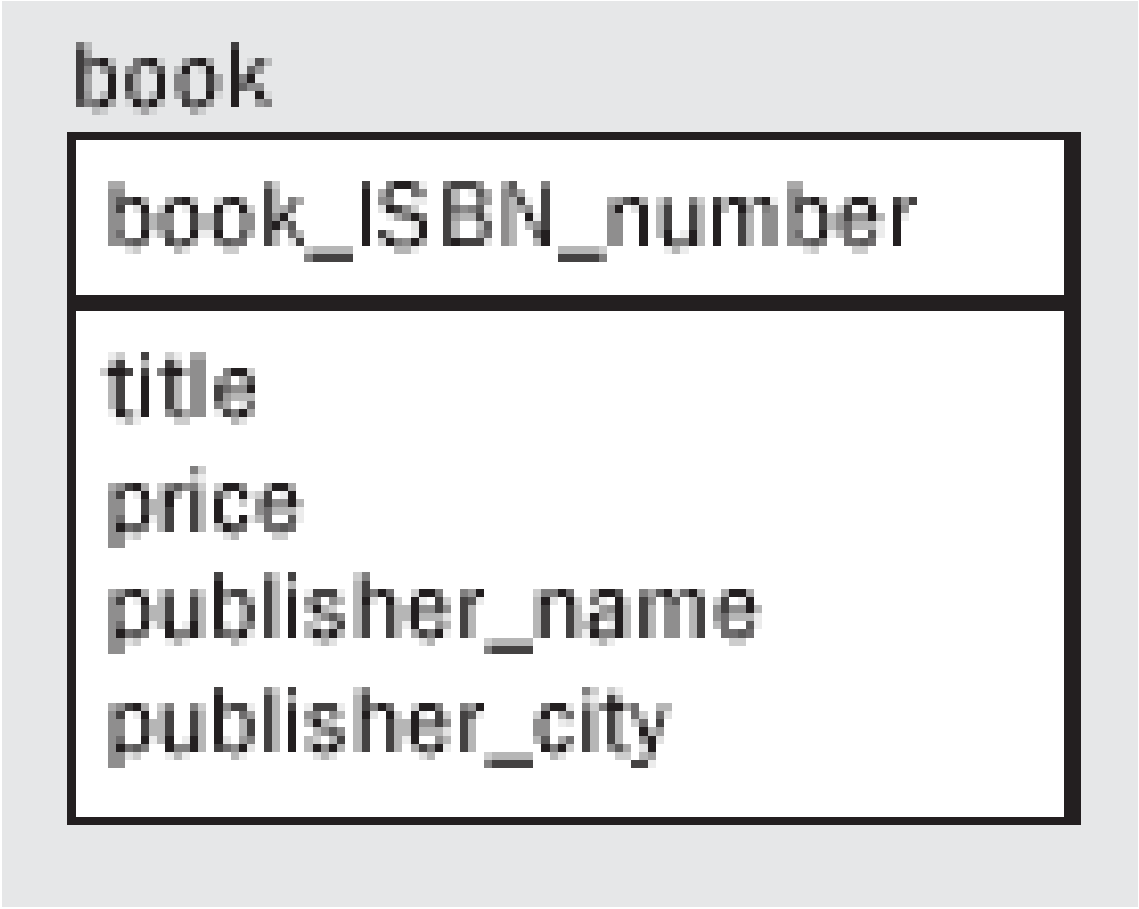



- {author_social_security_number} → {author_first_name, author_second_name}
- {Book_ISBN_number} → {book_title}
- {{author_social_security_number, Book_ISBN_number} → {royalty_percentage}

# Answer


book
book_ISBN_number
book_title
book_author
author_social_security_number (FK)
book_ISBN_number (FK)
royalty_percentage
author
author_social_security_number
author_first_name
author_second_name

# Make it 3NF


book
book_ISBN_number
title
price
publisher_name
publisher_city

# Exercise

- $R(A, B, C, D, E, F)$
- $S = \{AB \rightarrow C, BC \rightarrow AD, D \rightarrow E, CF \rightarrow B\}$

- What is the Primary Key,

  if so whether R is in 3NF?

{AB}+ = {A,B,C,D,E} but F is missing

{ABF}+ = {A,B,C,D,E, F} → ABF can be the PK

# Exercise

- $R(A, B, C, D)$
- $S = \{BC \rightarrow D, D \rightarrow A, A \rightarrow B\}$

- What is the Primary Key,

  if so whether R is in 3NF?

# Exercise

- $R(A, B, C, D)$
- $S = \{A \rightarrow B, A \rightarrow C, C \rightarrow D\}$

- What is the Primary Key,

  if so whether R is in 3NF?

# Exercise

- $R\ (A, B, C, D)$
- $S = \{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A\}$

- What is the Primary Key,

  if so whether R is in 3NF?

# Exercise

- $R\ (A, B, C, D)$
- $S = \{AD \rightarrow B, AB \rightarrow C, BC \rightarrow D, CD \rightarrow A\}$

- What is the Primary Key,

  if so whether R is in 3NF?