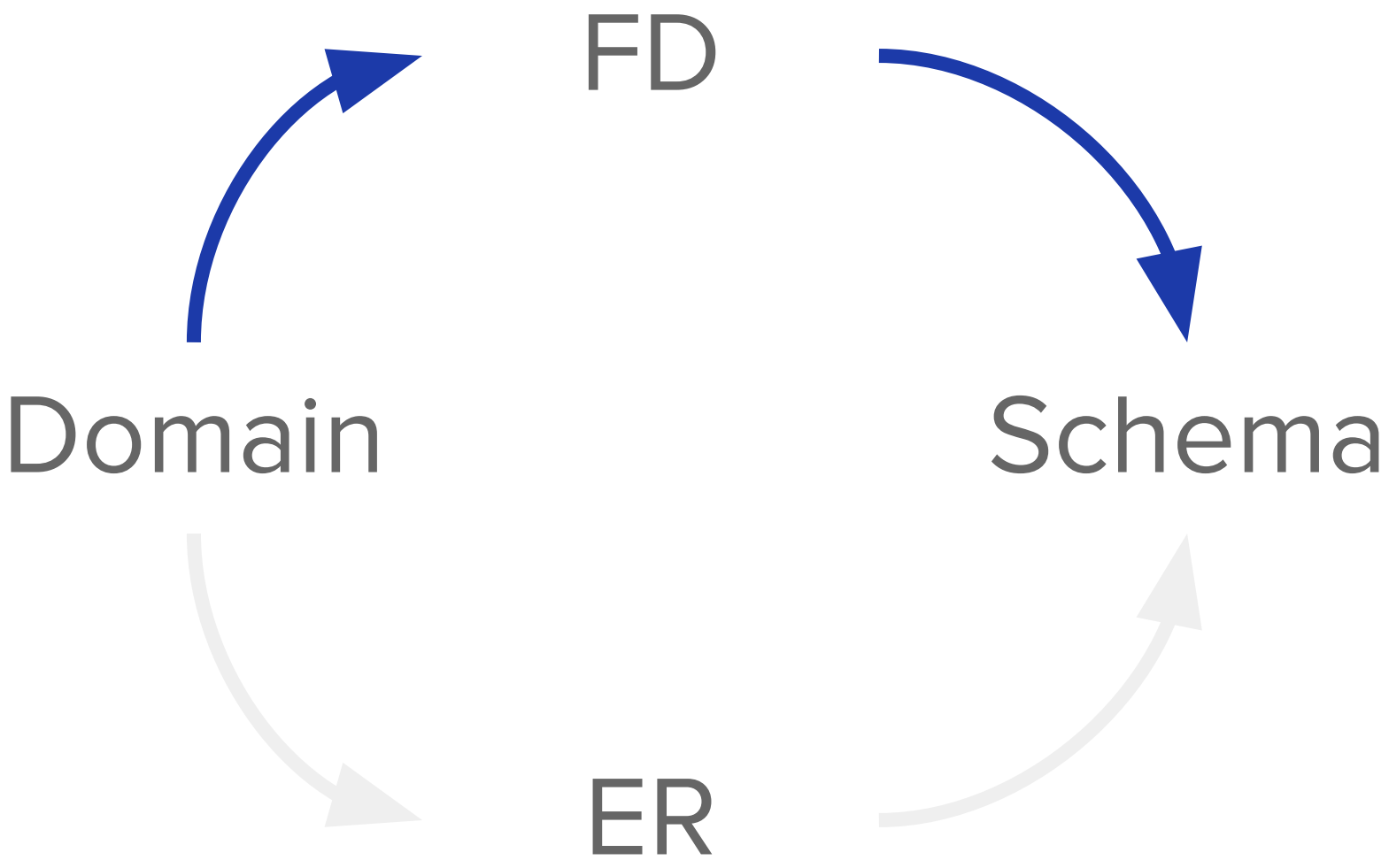
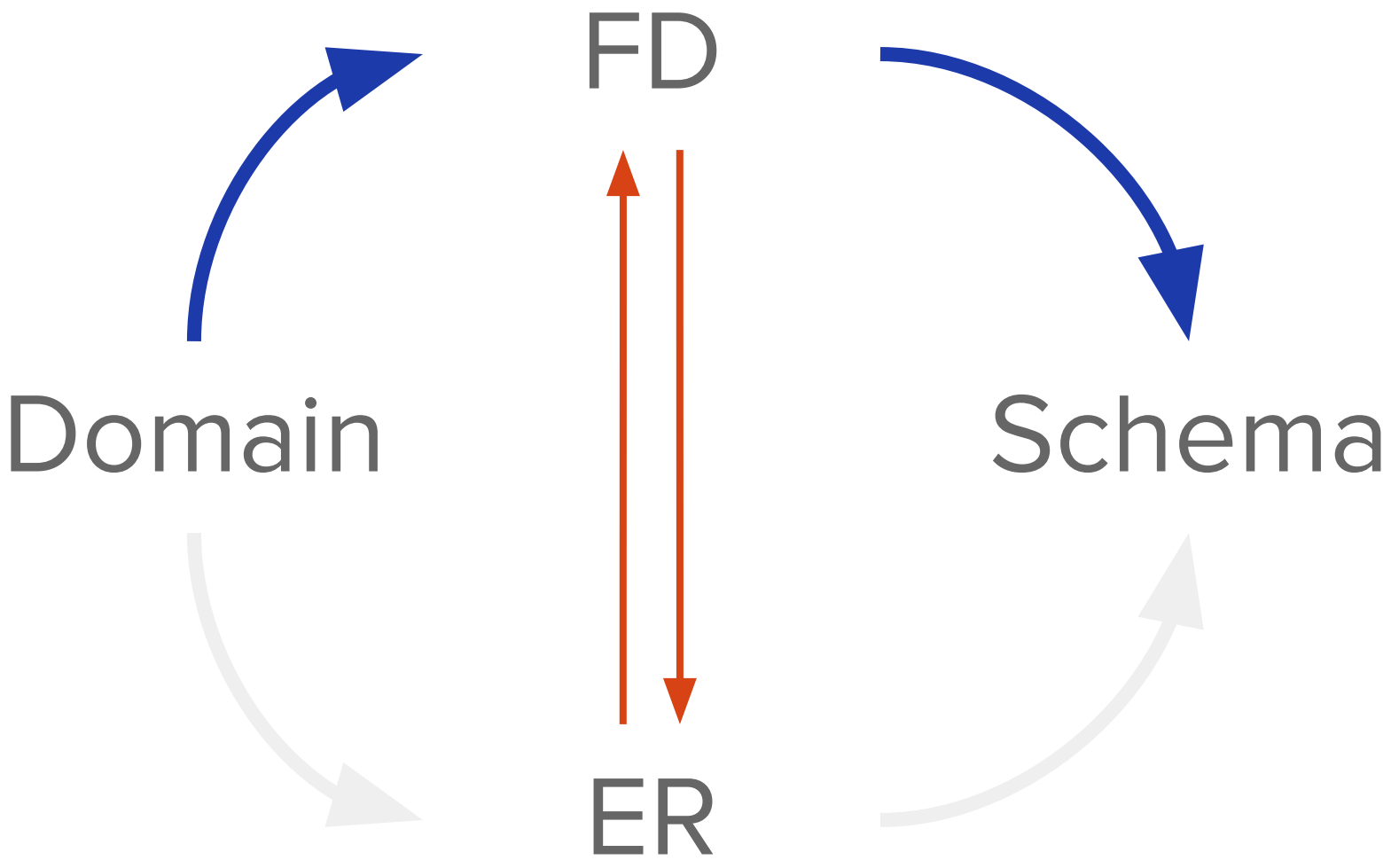
Design Theory

Functional Dependencies


Domain
ER
Schema


FD
Domain
Schema
ER


FD
Domain
Schema
ER

Definition of functional dependency

If two tuples of relation **R** agree on all the attributes $A_1 A_2 \ldots A_n$ then they must also agree on attribute B

$$A_1 A_2 \ldots A_n \rightarrow B$$

Name	Mother	Grandmother
Stefan	*XXX*	*XXX*
Stefan	*XXX*	*XXX*

Name → Name

Name	Mother	Grandmother
Stefan	XXX	XXX
Stefan	XXX	XXX

Trivially true since nothing new is asserted.

Name → Mother

Name	Mother	Grandmother
Stefan	Margareta	XXX
Stefan	Margareta	XXX

Everyone has a mother.

Mother → Grandmother

Name	Mother	Grandmother
Stefan	Margareta	Ulla
Stefan	Margareta	Ulla

Every mother has a mother.

Name ➔ Grandmother

Name	Mother	Grandmother
Stefan	Margareta	Ulla
Stefan	Margareta	Ulla

By transitivity, everyone has a grandmother.

The dependency part of FDs

A functional dependency on the form “$A_1\ A_2 \dots A_n \rightarrow B$” ...

- ... where $B \in \{ A_1\ A_2 \dots A_n \}$ is called trivial.
  - Name ➔ Name
  - Name Mother ➔ Name

- ... and another FD “$B \rightarrow C$” can be combined into “$A_1\ A_2 \dots A_n \rightarrow C$”.
  - Name ➔ Mother & Mother ➔ Grandmother ⇒ Name ➔ Grandmother

Finding FDs in a domain description

As an example, consider a course currently running at some university. Each period, i.e. spring or autumn, a course has a number of students enrolled and a teacher that is responsible for it. Lectures are held in some room at most once a day under a certain period, and rooms have a limited number of seats.

Finding FDs in a domain description

As an example, consider a course currently running at some university. Each period, i.e. spring or autumn, a course has a number of students enrolled and a teacher that is responsible for it. Lectures are held in some room at most once a day under a certain period, and rooms have a limited number of seats.

Course (name, #students, teacher, period, day, room, #seats)

name, period ➔ #students, teacher

Finding FDs in a domain description

As an example, consider a course currently running at some university. Each period, i.e. spring or autumn, a course has a number of students enrolled and a teacher that is responsible for it. Lectures are held in some room at most once a day under a certain period, and rooms have a limited number of seats.

Course (name, #students, teacher, period, day, room, #seats)

name, period → #students, teacher
name, period, day → room
room → #seats

The dependency part of FDs

$$A_1\, A_2 \cdots A_n \rightarrow B_1\, B_2 \cdots B_m$$

is equivalent to

$$A_1\, A_2 \cdots A_n \rightarrow B_1$$
$$A_1\, A_2 \cdots A_n \rightarrow B_2$$
$$\vdots$$
$$A_1\, A_2 \cdots A_n \rightarrow B_m$$

The functional part of FDs

A functional dependency of "$A_1\ A_2 \ldots A_n \rightarrow B$" can conceptually be thought of partial function from $A_1\ A_2 \ldots A_n$ to B

Note: The function is a mathematical one in the sense that it always returns the same result for the same inputs.

Note: Unlike a function, a functional dependency does not inspect the values of $\{ A_1\ A_2 \ldots A_n \}$ to produce the value B, it looks up its value in some table.

Note: Partial means that it may have no result for some inputs.

The functional part of FDs

As an example, consider a course currently running at some university. Each period, i.e. spring or autumn, a course has a number of students enrolled and a teacher that is responsible for it. Lectures are held in some room **at most** once a day under a certain period, and rooms have a limited number of seats.

Course (name, #students, teacher, period, day, room, #seats)

name, period → #students, teacher
name, period, day → room
room → #seats

Movies (Title, Year, Length, Genre, Studio, Star)

Title	*Year*	*Length*	*Genre*	*Studio*	*Star*
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

Trying to do to much leads to anomalies

1. **Redundancy**. Information about a movies length, genre, and studio is repeated for each actor/actress.

2. **Update Anomalies**. If we change, for example, the genre of Star Wars to documentary, then we must change several rows. Failing to update all rows will lead to inconsistencies in our relation.

3. **Deletion Anomalies**. If we want to remove Vivien Leigh as an actress for Gone With the Wind then we would also lose all other information about the movie.

Keys to the rescue

The set of attributes ***A*** is a *key* for a relation ***R*** if and only if:

1. **Complete**. The attributes of A functionally determine all other attributes of ***R***, that is, the closure of ***A*** contains all attributes of ***R***.

2. **Minimal**. There does not exist a proper subset of ***A*** that functionally determine all other attributes of ***R***, that is, a key must be *minimal*.

Keys that are not minimal are often referred to as super keys.

Finding the closure of attributes

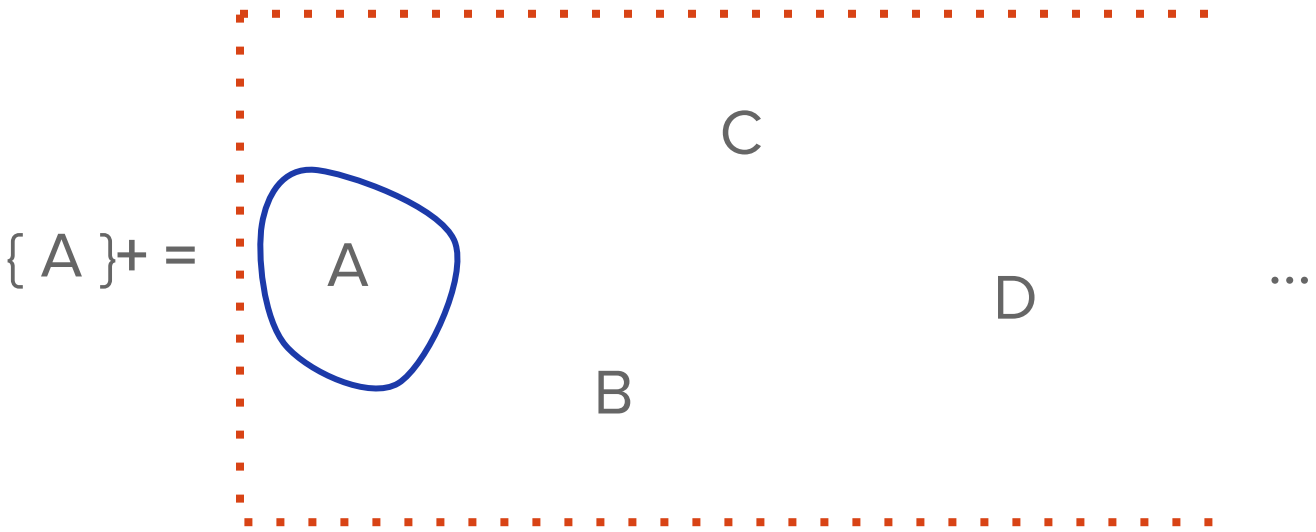
Given a set of attributes **A** and a set of FDs **F**, we compute the closure **A**+ of **A** by:

1. Search for some for some FD "$B_1 \ldots B_m \rightarrow C$" such that $B_1 \ldots B_m \subseteq$ **A** and $C \notin$ **A**.

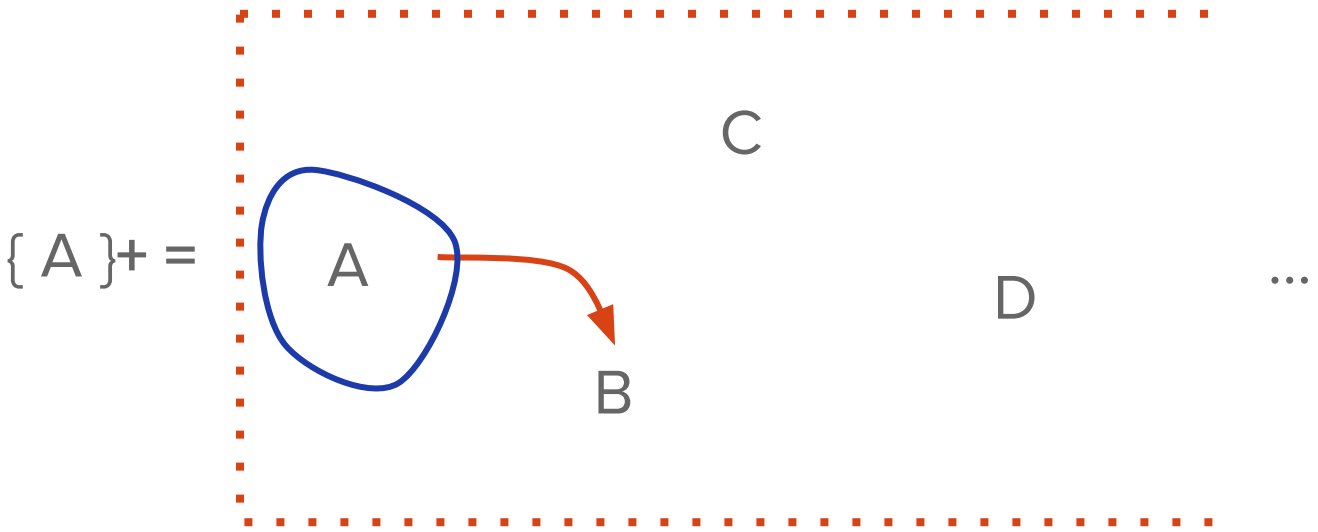
2. Add C to **A**.

3. Repeat until no additional FDs satisfying the second step can be found.

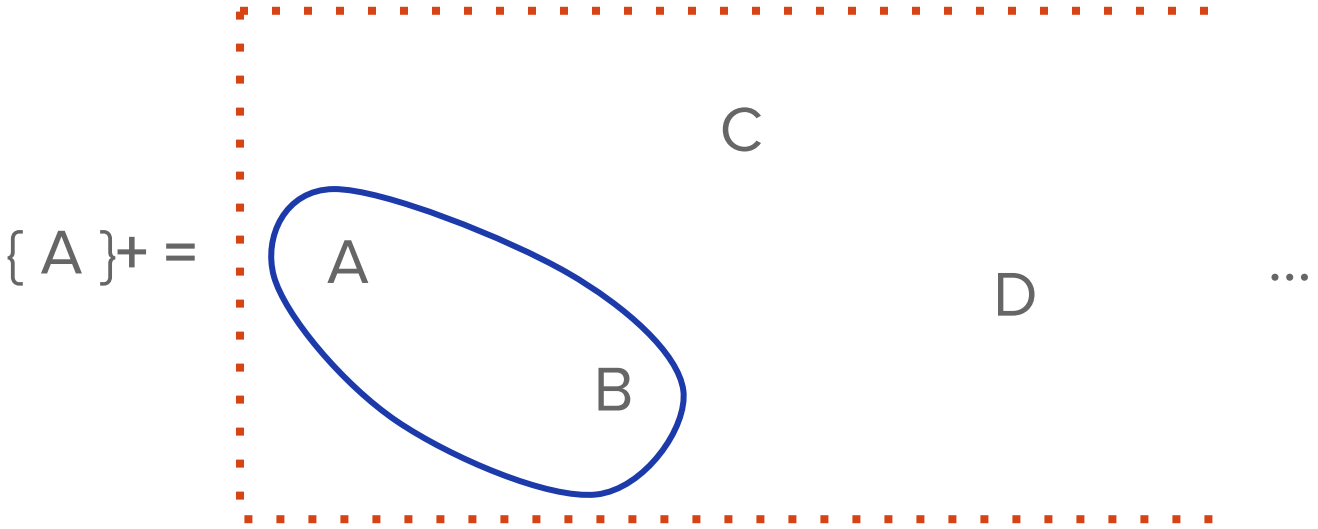
R (A,B,C,D,...) with FDs A → B & A,B → C & ...



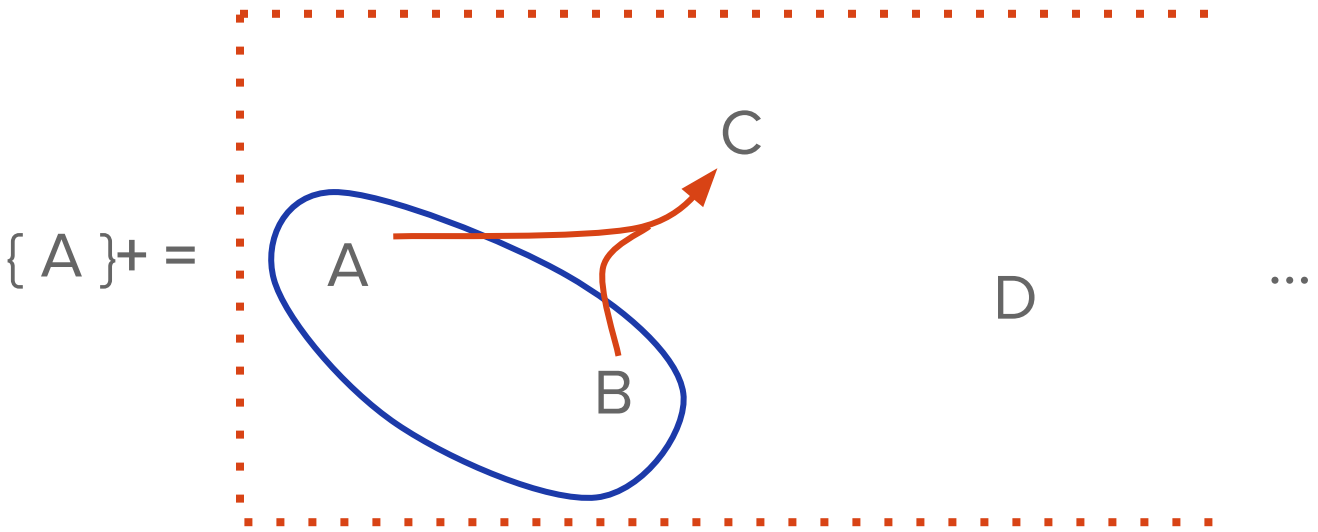
R (A,B,C,D,...) with FDs **A → B** & A,B → C & ...



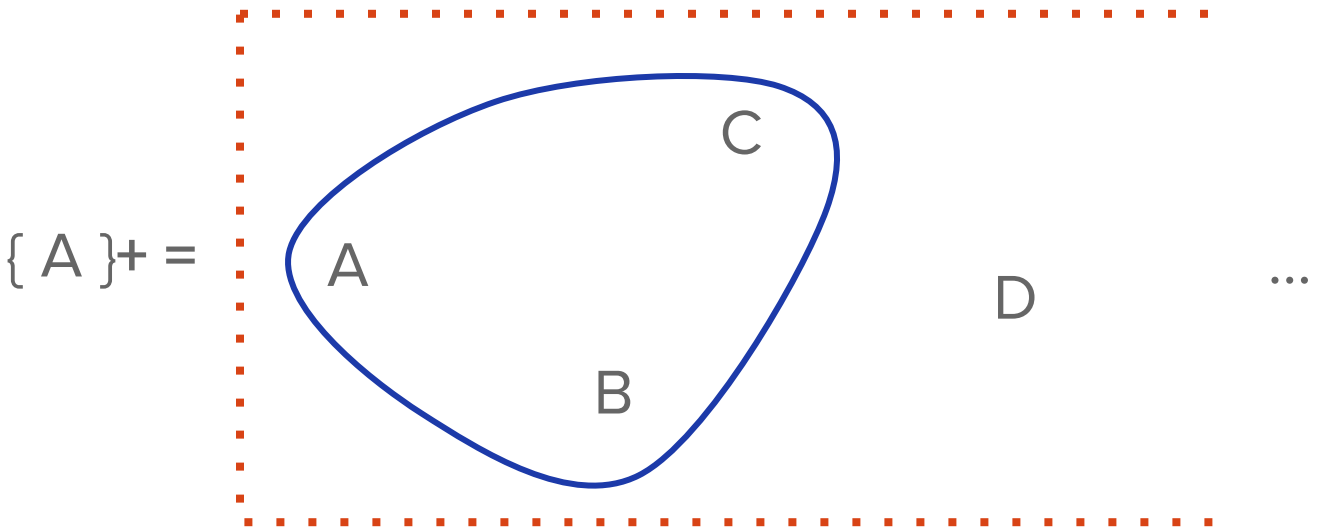
R (A,B,C,D,...) with FDs A → B & A,B → C & ...



R (A,B,C,D,...) with FDs A → B & **A,B → C** & ...



R (A,B,C,D,...) with FDs A → B & A,B → C & ...



? FDs ?

Title	*Year*	*Length*	*Genre*	*Studio*	*Star*
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, Studio

Title	*Year*	*Length*	*Genre*	*Studio*	*Star*
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 }+ = { Title, Year, Length, Genre, Studio }

Title	*Year*	*Length*	*Genre*	*Studio*	*Star*
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

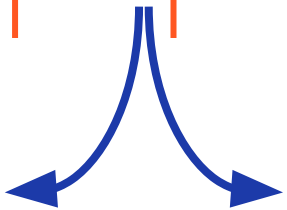
Key = { Title, Year, Star }

Title	*Year*	*Length*	*Genre*	*Studio*	*Star*
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

Normal form

The accepted way to eliminate anomalies is to *decompose* relations into NF, and typically involves splitting a relation into two or more smaller relations.

Title	*Year*	*Length*	*Genre*	*Studio*	*Star*
Star Wars	1977	124	SciFi	Fox	Carrie Fisher
⋮					⋮



Title	*Year*	*Length*	*Genre*	*Studio*
Star Wars	1977	124	SciFi	Fox
⋮				⋮

Title	*Year*	*Star*
Star Wars	1977	Carrie Fisher
⋮		⋮

Boyce-Codd Normal Form

A relation **R** is in Boyce-Codd Normal Form if and only if:

- Whenever there is a non-trivial FD "$A_1\ A_2 \ldots A_n \rightarrow B$" for **R**, it is the case that $\{ A_1\ A_2 \ldots A_n \}$ is a *superkey* for **R**.

Title, Year → Length, Genre, Studio

Title	*Year*	*Length*	*Genre*	*Studio*	*Star*
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

Decompose to Boyce-Codd Normal Form

Given a relation **R** and a set of FDs **F**, we compute the BCNF of **R** by:

1. Check if **R** is in BCNF, if so we are done.

2. Find *an* FD "**A** → **B**" in **F** that violates BCNF for **R** and
   a. Compute **A**+.
   b. Create a new relation $\mathbf{R}_1$ = **A**+
   c. Create a new relation $\mathbf{R}_2$ = **A** ∪ (**R** / **A**+)

3. Recursively apply this algorithm to $\mathbf{R}_1$ and $\mathbf{R}_2$.

Title, Year → Length, Genre, Studio

Title	*Year*	*Length*	*Genre*	*Studio*	*Star*
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

Step 1 & 2: Find violating FD

{ Title, Year }+ = { Title, Year, Length, Genre, Studio }

Title	*Year*	*Length*	*Genre*	*Studio*	*Star*
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

Step 2 a: Closure of said FD

$$R_1 = \{ \text{Title, Year} \}+$$

Title	*Year*	*Length*	*Genre*	*Studio*
Star Wars	1977	124	SciFi	Fox
Gone With the Wind	1939	231	Drama	MGM
Wayne's World	1992	95	Comedy	Paramount

Step 2 b: Create relation from closure

$$R_2 = \{ \text{Title, Year} \} \cup (\textbf{Movies} / \{ \text{Title, Year} \}+)$$

(Title, Year) → R1.(Title, Year)

Title	*Year*	*Star*
Star Wars	1977	Carrie Fisher
Star Wars	1977	Mark Hamill
Star Wars	1977	Harrison Ford
⋮		⋮

Step 2 c: Create relation from leftovers

Title	*Year*	*Length*	*Genre*	*Studio*
Star Wars	1977	124	SciFi	Fox
Gone With the Wind	1939	231	Drama	MGM
Wayne's World	1992	95	Comedy	Paramount

Title	*Year*	*Star*
Star Wars	1977	Carrie Fisher
Star Wars	1977	Mark Hamill
Star Wars	1977	Harrison Ford
Gone With the Wind	1939	Vivien Leigh
Wayne's World	1992	Dana Carvey
Wayne's World	1992	Mike Meyers

1. **No Redundancy**
2. **No Update Anomalies**.
3. **No Deletion Anomalies**.

No more anomalies, but …

There's no guarantee that dependencies are preserved in the smaller relations.

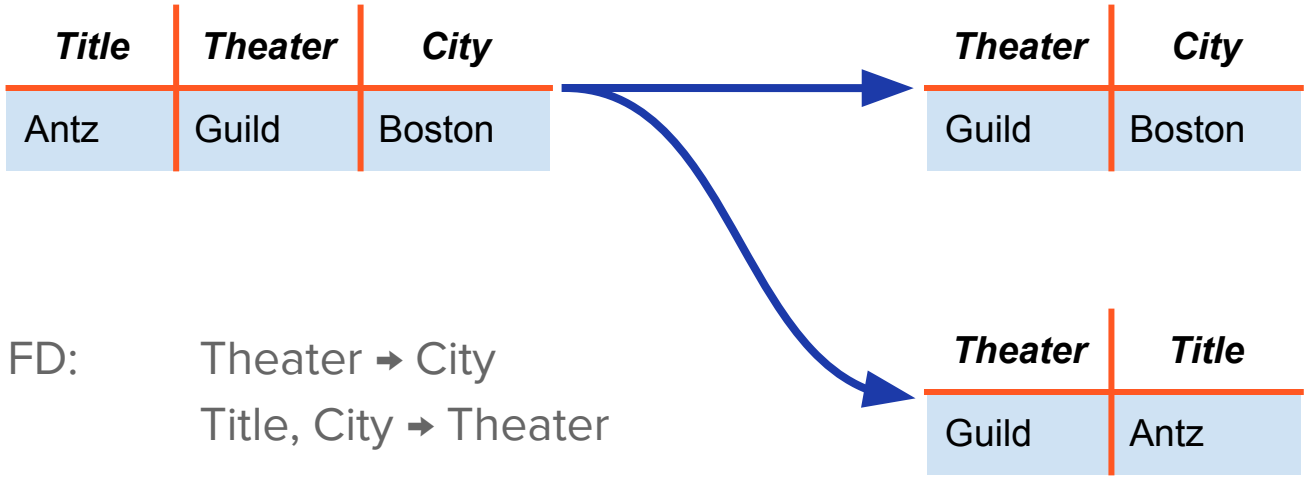
Title	*Theater*	*City*
Antz	Guild	Boston

FD: Theater → City
Title, City → Theater

Keys: { Title, City } & { Title, Theater }

No more anomalies, but …

There's no guarantee that dependencies are preserved in the smaller relations.

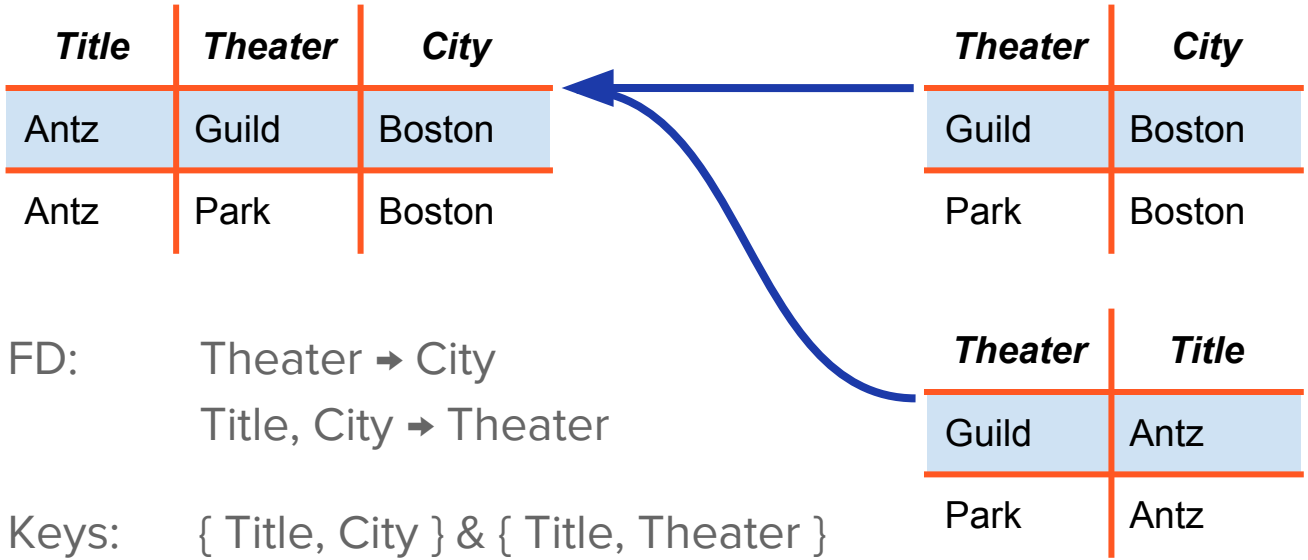


FD: Theater → City
Title, City → Theater

Keys: { Title, City } & { Title, Theater }

No more anomalies, but …

There's no guarantee that dependencies are preserved in the smaller relations.



FD: Theater → City
Title, City → Theater

Keys: { Title, City } & { Title, Theater }

First, Second, and Third Normal Form

$$1 < 2 < 3$$

First Normal Form

A relation is in 1NF if and only if the domain of each attribute contains only *atomic* values, and the value of each attribute contains only a single value from that domain.

ID	Name	Phone
1	Markus	555-861, 192-122
2	Selpi	555-403

OR

ID	Name	Phone 1	Phone 2
1	Markus	555-861	192-122
2	Selpi	555-403	

ID	Name	Phone
1	Markus	555-861
1	Markus	192-122
2	Selpi	555-403

Second Normal Form

A relation is in 2NF if and only if it is in 1NF and there's no non-prime attributes determined by a subset of some key, where non-prime means "attribute that's not part of any key for the relation".

Manufacturer	*Item*	*Headquarters*
Dent-o-Fresh	EZ-Brush	USA
Hoch	Toothmaster	Germany

AND

Manufacturer	*Headquarters*
Dent-o-Fresh	USA
Hoch	Germany

Manufacturer	*Item*
Dent-o-Fresh	EZ-Brush
Hoch	Toothmaster

FD : Manufacturer → Headquarters

Key : { Manufacturer, Item }

Third Normal Form

A relation is in 3NF if and only if it is in 2NF and all of its attributes are determined only by the keys of that relation, not by non-prime attributes.

Tournament	*Year*	*Winner*	*Birthday*
Indiana Closed	1998	Al	21 Jul. 1975
Indiana Closed	1999	Chip	14 Mar. 1977
Cleveland Open	1998	Alice	21 Sep. 1975

AND

Tournament	*Year*	*Winner*
Indiana Closed	1998	Al
Indiana Closed	1999	Chip
Cleveland Open	1998	Alice

Winner	*Birthday*
Al	21 Jul. 1975
Chip	14 Mar. 1977
Alice	21 Sep. 1975

FD : Winner ➔ Birthday & Tournament, Year ➔ Winner

Key : { Tournament, Year }

Third Normal Form

A relation **R** is in 3NF if and only if:

- Whenever there is a non-trivial FD "$A_1\ A_2 \ldots A_n \rightarrow B$" for **R**, it is the case that $\{ A_1\ A_2 \ldots A_n \}$ is a *superkey* for **R** <u>or</u> B is part of some key.

Decomposing to Third Normal Form

Given a relation **R** and a set of FDs **F**, we compute the 3NF of **R** by:

1. Check if **R** is in 3NF, if so we are done.

2. For each functional dependency "**A → B**" in **F** we create a relation from **A** ∪ **B**.

3. If none of the sets of relations from the second step is a superkey for **R**, add another relation whose schema is a key for **R**.

Decomposing to Third Normal Form

As an example, consider **R** (A, B, C, D, E) and FDs “A B → C”, “C → B”, and “A → D”.

Decomposing to Third Normal Form

As an example, consider **R** (A, B, C, D, E) and FDs “A B → C”, “C → B”, and “A → D”.

{ A, B }+ = { A, B, C, D }
{ C }+ = { C, B }
{ A }+ = { A, D }

⇒ Keys are { A, B, E } or { A, C, E }.

So “A → D” violates 3NF since { A } is not a super key and D is not part of a key.

Step 1: Already in 3NF?

Decomposing to Third Normal Form

As an example, consider **R** (A, B, C, D, E) and FDs “A B → C”, “C → B”, and “A → D”.

A B → C ⇒ $\mathbf{R_1}$ (A, B, C)
C → B ⇒ $\mathbf{R_2}$ (C, B)
A → D ⇒ $\mathbf{R_3}$ (A, D)

But no relation contains a key from **R**, so…

Step 2: Relation for each FD

Decomposing to Third Normal Form

As an example, consider **R** (A, B, C, D, E) and FDs “A B → C”, “C → B”, and “A → D”.

A B → C ⇒ $\mathbf{R_1}$ (A, B, C)
C → B ⇒ $\mathbf{R_2}$ (C, B)
A → D ⇒ $\mathbf{R_3}$ (A, D)

... we have to add it ⇒ $\mathbf{R_4}$ (A, B, E)

Step 3: Missing key

No missing dependencies, but …

There's no guarantee that redundancy is eliminated from a relation.

Title	*Theater*	*City*
Antz	Guild	Boston
Beez	Guild	Boston

FD: Theater ➔ City
Title, City ➔ Theater

Keys: { Title, City } & { Title, Theater }

The End