L14: Normalization 2

CS3200 Database design (fa18 s2)

https://northeastern-datalab.github.io/cs3200/

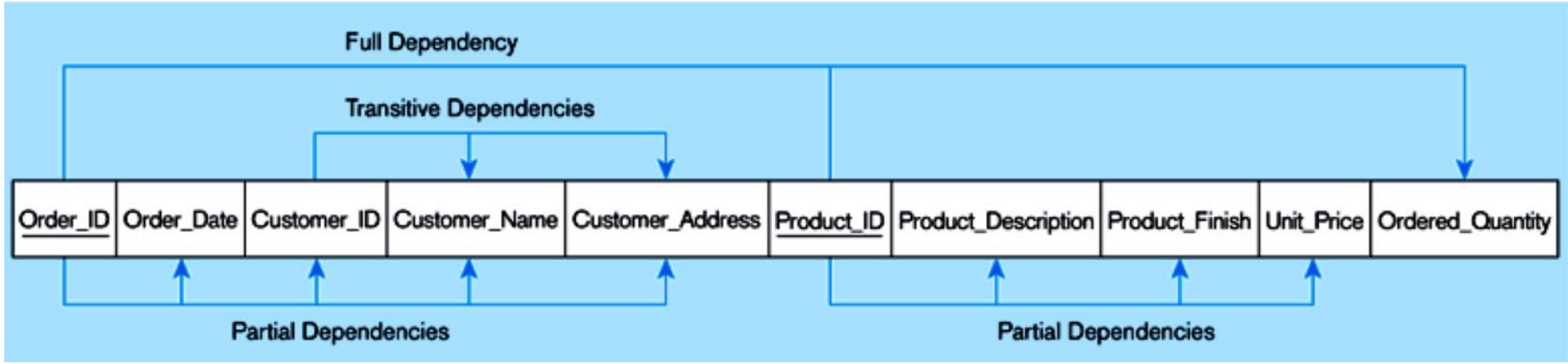
Version 10/25/2018

Announcements!

- Class contributions:
  - Great by some! Thanks! Everyone: Please keep bringing your name plates
  - Recall that I ask people in class rows if nobody answers
- Changed Gradiance policies (feedback that you spend too much time)
  - you can work together on Gradiance
  - bumped Gradiance scores
- HW6 will be posted today
- Jupyter exercises

Quick recap FDs

- **Functional Dependency (FD)**: The value of one set of attributes (the **determinant**) uniquely determines the value of another set of attributes (the **dependents**)
- A **superkey (SK)** is as a set of attributes of a relation schema upon which all attributes of the schema are functionally dependent.
- A **(candidate) key (CK)** is a non-redundant (minimal) SK (sometimes called just "a key")
- **Prime attribute**: belonging to some candidate key
- **Partial FD**: FD in which more non-prime attributes are functionally dependent on part (but not all) of any CK
- **Transitive FD**: An FD between two (or more) nonkey attributes (important for distinction 3NF vs BCNF!)
- **3NF**: no partial nor transitive FD



“Good” vs. “Bad” FDs

We can start to develop a notion of **good** vs. **bad** FDs:

EmpID	Name	Phone	Position
E0045	Smith	1234	Clerk
E3542	Mike	9876	Salesrep
E1111	Smith	9876	Salesrep
E9999	Mary	1234	Lawyer

Intuitively:

EmpID -> Name, Phone, Position is *“good FD”*

- ***Minimal redundancy, less possibility of anomalies***

“Good” vs. “Bad” FDs

We can start to develop a notion of **good** vs. **bad** FDs:

EmpID	Name	Phone	Position
E0045	Smith	1234	Clerk
E3542	Mike	9876	Salesrep
E1111	Smith	9876	Salesrep
E9999	Mary	1234	Lawyer

Intuitively:

EmpID -> Name, Phone, Position is *“good FD”*

But Position -> Phone *is a “bad FD”*

- ***Redundancy! Possibility of data anomalies***

“Good” vs. “Bad” FDs

Student	Course	Room
Mary	CS3200	WVF20
Joe	CS3200	WVF20
Sam	CS3200	WVF20
..	..	..

Returning to our original example… can you see how the “bad FD” {Course} -> {Room} could lead to an:

- Update Anomaly
- Insert Anomaly
- Delete Anomaly
- …

Given a set of FDs (from user) our goal is to:

1. Find all FDs, and
2. Eliminate the “Bad Ones".

FDs for Relational Schema Design

- High-level idea: why do we care about FDs?

1. Start with some relational schema
2. Find out its functional dependencies (FDs)
3. Use these to design a better schema
   - One which minimizes possibility of anomalies

This part can be tricky!

Finding Functional Dependencies

- There can be a very large number of FDs…
  - How to find them all efficiently?

- We can't necessarily show that any FD will hold on all instances…
  - How to do this?

We will start with this problem:
Given a set of FDs, F, what other FDs ***must*** hold?

Finding Functional Dependencies

- Equivalent to asking: Given a set of FDs, F = $\{f_1,\ldots f_n\}$, does an FD g hold?
  - Inference problem: How do we decide?

Example:

Products

Name	Color	Category	Dep	Price
Gizmo	Green	Gadget	Toys	49
Widget	Black	Gadget	Toys	59
Gizmo	Green	Whatsit	Garden	99

Provided FDs:

1. {Name} → {Color}
2. {Category} → {Department}
3. {Color, Category} → {Price}

{N, CA} → P

Given the provided FDs, we can see that {Name, Category} → {Price} must also hold on **any instance**…

Which / how many other FDs do?!?

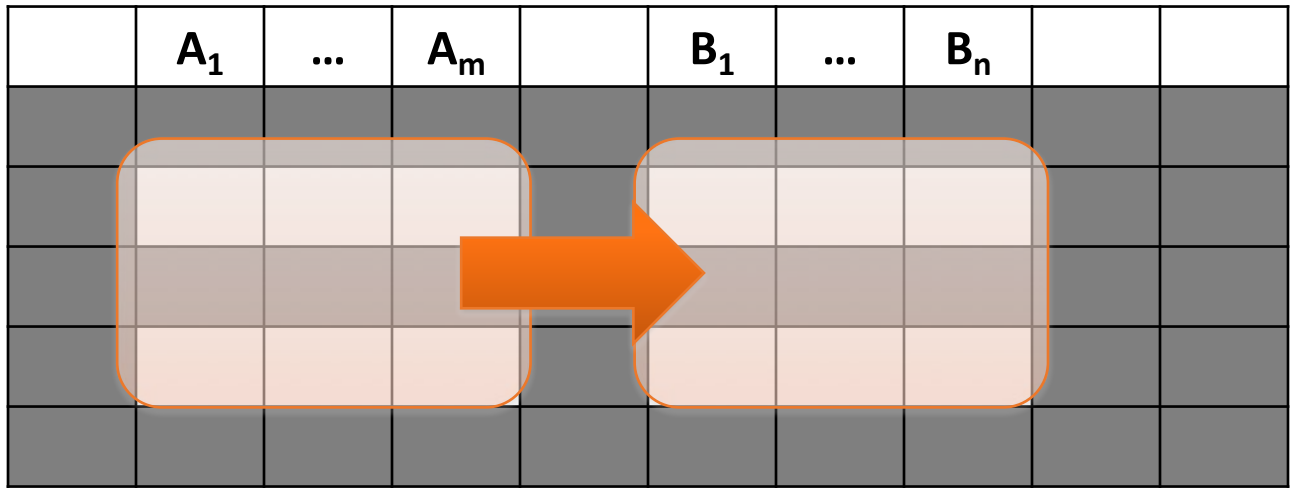
Finding Functional Dependencies

- Equivalent to asking: Given a set of FDs, F = $\{f_1,\ldots f_n\}$, does an FD g hold?
  - Inference problem: How do we decide?

Answer: Three simple rules called **Armstrong's Rules.**

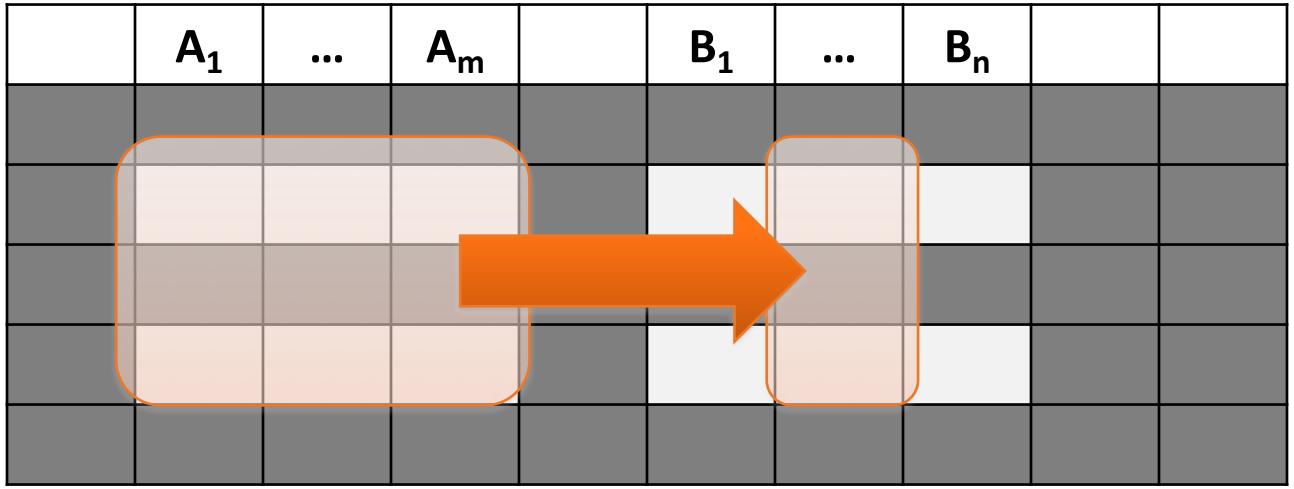
1. **Split/Combine,**
2. **Reduction (Trivial), and**
3. **Transitivity…** *ideas by picture*

1. Split/Combine



$A_1, \ldots, A_m \rightarrow B_1,\ldots,B_n$

1. Split/Combine

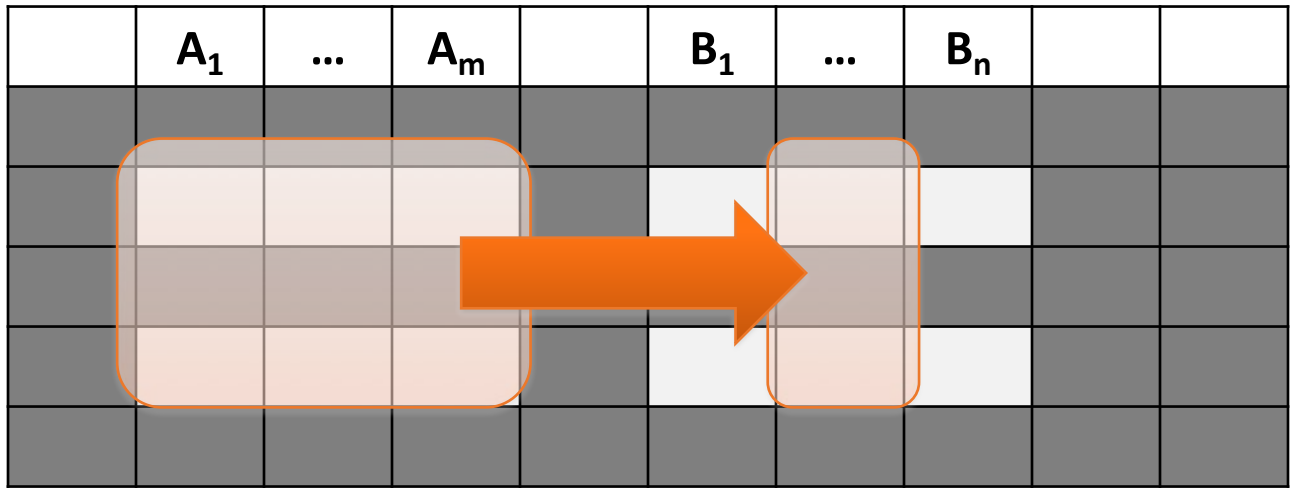


$A_1, \ldots, A_m \rightarrow B_1,\ldots,B_n$

… is equivalent to the following *n* FDs…

$A_1,\ldots,A_m \rightarrow B_i$ for i=1,…,n

1. Split/Combine

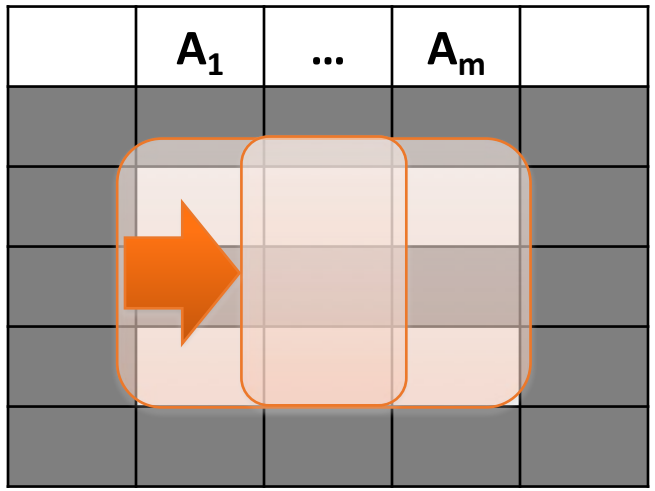


And vice-versa, $A_1,\ldots,A_m \rightarrow B_i$ for i=1,…,n

… is equivalent to …

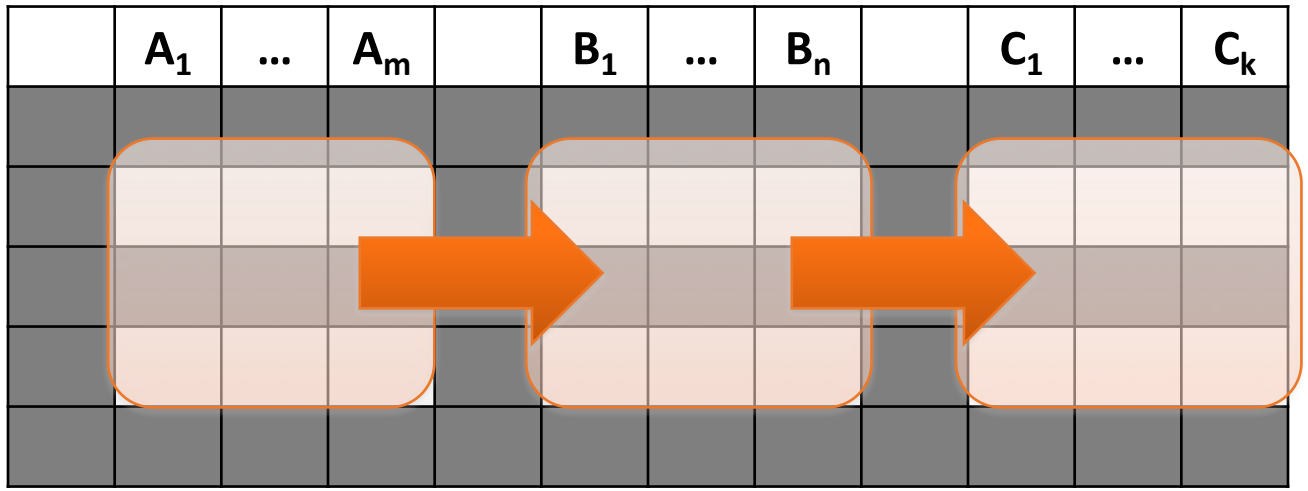
$A_1, \ldots, A_m \rightarrow B_1,\ldots,B_n$

2. Reduction (Trivial)



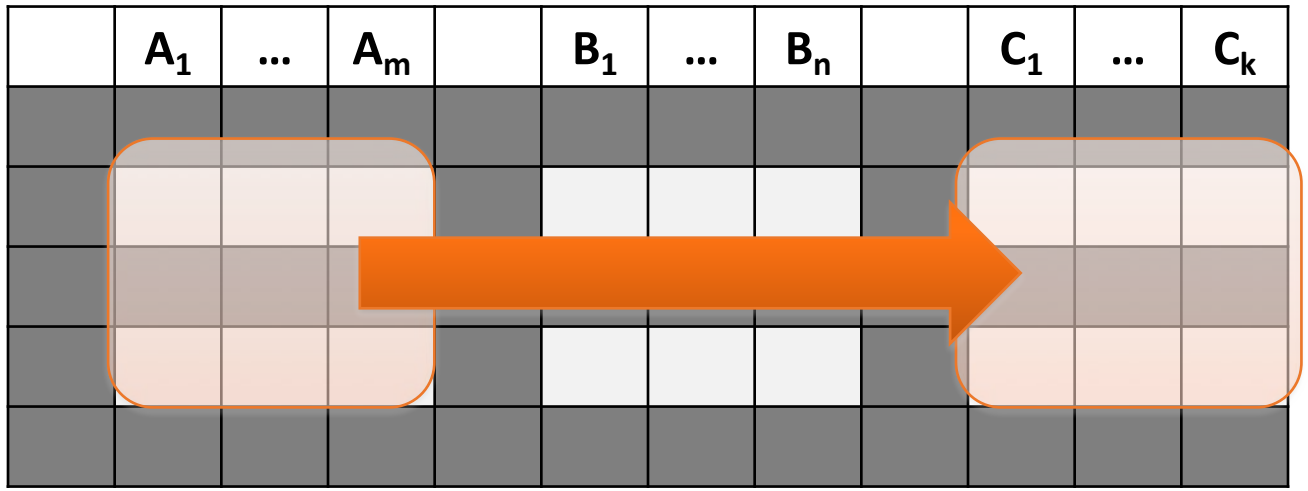
$A_1,\ldots,A_m \rightarrow A_j$ for any $j=1,\ldots,m$

3. Transitive Closure



$A_1, \ldots, A_m \rightarrow B_1,\ldots,B_n$ and
$B_1,\ldots,B_n \rightarrow C_1,\ldots,C_k$

3. Transitive Closure



$A_1, \ldots, A_m \rightarrow B_1,\ldots,B_n$ and
$B_1,\ldots,B_n \rightarrow C_1,\ldots,C_k$

implies

$A_1,\ldots,A_m \rightarrow C_1,\ldots,C_k$

Finding Functional Dependencies

Example:

Products

Name	Color	Category	Dep	Price
Gizmo	Green	Gadget	Toys	49
Widget	Black	Gadget	Toys	59
Gizmo	Green	Whatsit	Garden	99

Provided FDs:

1. {Name} → {Color}
2. {Category} → {Department}
3. {Color, Category} → {Price}

Which / how many other FDs hold?

Finding Functional Dependencies

Armstrong's Rules:
1. Split/Combine,
2. Reduction (Trivial)
3. Transitivity

Example:

Inferred FDs:

Inferred FD	Rule used
4. {Name, Category} -> {Name}	?
5. {Name, Category} -> {Color}	?
6. {Name, Category} -> {Category}	?
7. {Name, Category -> {Color, Category}	?
8. {Name, Category} -> {Price}	?

Provided FDs:

1. {Name} → {Color}
2. {Category} → {Dept.}
3. {Color, Category} → {Price}

Which / how many other FDs hold?

Finding Functional Dependencies

Armstrong's Rules:
1. Split/Combine,
2. Reduction (Trivial)
3. Transitivity

Example:

Inferred FDs:

Inferred FD	Rule used
4. {Name, Category} -> {Name}	Trivial
5. {Name, Category} -> {Color}	?
6. {Name, Category} -> {Category}	?
7. {Name, Category -> {Color, Category}	?
8. {Name, Category} -> {Price}	?

Provided FDs:

1. {Name} → {Color}
2. {Category} → {Dept.}
3. {Color, Category} → {Price}

Which / how many other FDs hold?

Finding Functional Dependencies

Armstrong's Rules:
1. Split/Combine,
2. Reduction (Trivial)
3. Transitivity

Example:

Inferred FDs:

Inferred FD	Rule used
4. {Name, Category} -> {Name}	Trivial
5. {Name, Category} -> {Color}	Transitive (4 -> 1)
6. {Name, Category} -> {Category}	?
7. {Name, Category -> {Color, Category}	?
8. {Name, Category} -> {Price}	?

Provided FDs:

1. {Name} → {Color}
2. {Category} → {Dept.}
3. {Color, Category} → {Price}

Which / how many other FDs hold?

Finding Functional Dependencies

Armstrong's Rules:
1. Split/Combine,
2. Reduction (Trivial)
3. Transitivity

Example:

Inferred FDs:

Inferred FD	Rule used
4. {Name, Category} -> {Name}	Trivial
5. {Name, Category} -> {Color}	Transitive (4 -> 1)
6. {Name, Category} -> {Category}	Trivial
7. {Name, Category -> {Color, Category}	?
8. {Name, Category} -> {Price}	?

Provided FDs:

1. {Name} → {Color}
2. {Category} → {Dept.}
3. {Color, Category} → {Price}

Which / how many other FDs hold?

Finding Functional Dependencies

Armstrong's Rules:
1. Split/Combine,
2. Reduction (Trivial)
3. Transitivity

Example:

Inferred FDs:

Inferred FD	Rule used
4. {Name, Category} -> {Name}	Trivial
5. {Name, Category} -> {Color}	Transitive (4 -> 1)
6. {Name, Category} -> {Category}	Trivial
7. {Name, Category -> {Color, Category}	Split/combine (5 + 6)
8. {Name, Category} -> {Price}	?

Provided FDs:

1. {Name} → {Color}
2. {Category} → {Dept.}
3. {Color, Category} → {Price}

Which / how many other FDs hold?

Finding Functional Dependencies

Armstrong's Rules:
1. Split/Combine,
2. Reduction (Trivial)
3. Transitivity

Can we find an algorithmic way to do this?

Example:

Inferred FDs:

Inferred FD	Rule used
4. {Name, Category} -> {Name}	Trivial
5. {Name, Category} -> {Color}	Transitive (4 -> 1)
6. {Name, Category} -> {Category}	Trivial
7. {Name, Category -> {Color, Category}	Split/combine (5 + 6)
8. {Name, Category} -> {Price}	Transitive (7 -> 3)

Provided FDs:

1. {Name} → {Color}
2. {Category} → {Dept.}
3. {Color, Category} → {Price}

Which / how many other FDs hold?

Closure of a set of Attributes

Given a set of attributes $A_1, \ldots, A_n$ and a set of FDs **F**:
Then the **closure**, $\{A_1, \ldots, A_n\}^+$ is the set of attributes **B** s.t. $\{A_1, \ldots, A_n\} \rightarrow B$

Example: F =

```
{name} → {color}
{category} → {dept}
{color, category} → {price}
```

Closure of a set of Attributes

Given a set of attributes $A_1, \ldots, A_n$ and a set of FDs **F**:
Then the **closure**, $\{A_1, \ldots, A_n\}^+$ is the set of attributes **B** s.t. $\{A_1, \ldots, A_n\} \rightarrow B$

Example: F =

```
{name} → {color}
{category} → {dept}
{color, category} → {price}
```

Example Closures:

```
{name}+ = ?
{name, category}+ = ?

{color}+ = ?
```

Closure of a set of Attributes

Given a set of attributes $A_1, \ldots, A_n$ and a set of FDs **F**:
Then the **closure**, $\{A_1, \ldots, A_n\}^+$ is the set of attributes **B** s.t. $\{A_1, \ldots, A_n\} \rightarrow B$

Example: F =

```
{name} → {color}
{category} → {dept}
{color, category} → {price}
```

Example Closures:

```
{name}+ = {name, color}
{name, category}+ = ?

{color}+ = ?
```

Closure of a set of Attributes

Given a set of attributes $A_1, \ldots, A_n$ and a set of FDs **F**:
Then the **closure**, $\{A_1, \ldots, A_n\}^+$ is the set of attributes **B** s.t. $\{A_1, \ldots, A_n\} \rightarrow B$

Example: F =

```
{name} → {color}
{category} → {dept}
{color, category} → {price}
```

Example Closures:

```
{name}+ = {name, color}
{name, category}+ =
  {name, category, color, ...}
{color}+ = ?
```

Closure of a set of Attributes

Given a set of attributes $A_1, \ldots, A_n$ and a set of FDs **F**:
Then the **closure**, $\{A_1, \ldots, A_n\}^+$ is the set of attributes **B** s.t. $\{A_1, \ldots, A_n\} \rightarrow B$

Example: F =

```
{name} → {color}
{category} → {dept}
{color, category} → {price}
```

Example Closures:

```
{name}+ = {name, color}
{name, category}+ =
  {name, category, color, dept, price}
{color}+ = ?
```

Closure of a set of Attributes

Given a set of attributes $A_1, \ldots, A_n$ and a set of FDs **F**:
Then the **closure**, $\{A_1, \ldots, A_n\}^+$ is the set of attributes **B** s.t. $\{A_1, \ldots, A_n\} \rightarrow B$

Example: F =

```
{name} → {color}
{category} → {dept}
{color, category} → {price}
```

Example Closures:

```
{name}+ = {name, color}
{name, category}+ =
  {name, category, color, dept, price}
{color}+ = {color}
```

Closure Algorithm

Start with X = $\{A_1, \ldots, A_n\}$ and set of FDs F.

Repeat until X doesn't change; **do**:

if $\{B_1, \ldots, B_m\} \rightarrow C$ is entailed by F

and $\{B_1, \ldots, B_m\} \subseteq X$

then add C to X.

Return X as X^+

Closure Algorithm

Start with X = $\{A_1, \ldots, A_n\}$, FDs F.
Repeat until X doesn't change; **do**:
  if $\{B_1, \ldots, B_m\} \rightarrow$ C is in F **and** $\{B_1, \ldots, B_m\} \subseteq$ X:
    then add C to X.
Return X as X^+

{name, category}$^+$ =
{name, category}

F =

{name} → {color}

{category} → {dept}

{color, category} → {price}

Closure Algorithm

Start with X = $\{A_1, \ldots, A_n\}$, FDs F.
Repeat until X doesn't change; **do**:
  if $\{B_1, \ldots, B_m\} \rightarrow$ C is in F **and** $\{B_1, \ldots, B_m\} \subseteq$ X:
    then add C to X.
Return X as X^+

{name, category}$^+$ =
{name, category}

{name, category}$^+$ =
{name, category, color}

F =

{name} → {color}

{category} → {dept}

{color, category} → {price}

Closure Algorithm

Start with X = $\{A_1, \ldots, A_n\}$, FDs F.
Repeat until X doesn't change; **do**:
  if $\{B_1, \ldots, B_m\} \rightarrow$ C is in F **and** $\{B_1, \ldots, B_m\} \subseteq$ X:
    then add C to X.
Return X as X^+

F =

{name} → {color}

{category} → {dept}

{color, category} → {price}

{name, category}$^+$ =
{name, category}

{name, category}$^+$ =
{name, category, color}

{name, category}$^+$ =
{name, category, color, dept}

Closure Algorithm

Start with X = $\{A_1, \ldots, A_n\}$, FDs F.
Repeat until X doesn't change; **do**:
  if $\{B_1, \ldots, B_m\} \rightarrow C$ is in F **and** $\{B_1, \ldots, B_m\} \subseteq X$:
    then add C to X.
Return X as X^+

F =

{name} → {color}

{category} → {dept}

{color, category} → {price}

{name, category}$^+$ =
{name, category}

{name, category}$^+$ =
{name, category, color}

{name, category}$^+$ =
{name, category, color, dept}

{name, category}$^+$ =
{name, category, color, dept, price}

Example

R(A,B,C,D,E,F)

{A,B} → {C}
{A,D} → {E}
{B} → {D}
{A,F} → {B}

Compute $\{A,B\}^+ = \{A, B,$ $\}$

Compute $\{A, F\}^+ = \{A, F,$ $\}$

Example

R(A,B,C,D,E,F)

{A,B} → {C}
{A,D} → {E}
{B} → {D}
{A,F} → {B}

Compute $\{A,B\}^+ = \{A, B, C, D \qquad \}$

Compute $\{A, F\}^+ = \{A, F, \qquad \}$

Example

R(A,B,C,D,E,F)

{A,B} → {C}
{A,D} → {E}
{B} → {D}
{A,F} → {B}

Compute $\{A,B\}^+ = \{A, B, C, D, E \quad\}$

Compute $\{A, F\}^+ = \{A, F, \quad\}$

Example

R(A,B,C,D,E,F)

$\{A,B\} \rightarrow \{C\}$
$\{A,D\} \rightarrow \{E\}$
$\{B\} \rightarrow \{D\}$
$\{A,F\} \rightarrow \{B\}$

Compute $\{A,B\}^+ = \{A, B, C, D, E\}$

Compute $\{A, F\}^+ = \{A, B, F, \quad\quad\quad\quad \}$

Example

R(A,B,C,D,E,F)

{A,B} → {C}
{A,D} → {E}
{B} → {D}
{A,F} → {B}

Compute $\{A,B\}^+ = \{A, B, C, D, E\}$

Compute $\{A, F\}^+ = \{A, B, C, F, \quad\}$

Example

R(A,B,C,D,E,F)

{A,B} → {C}
{A,D} → {E}
{B} → {D}
{A,F} → {B}

Compute $\{A,B\}^+ = \{A, B, C, D, E\}$

Compute $\{A, F\}^+ = \{A, B, C, D, E, F\}$

Closures, Superkeys, and (Candidate) Keys

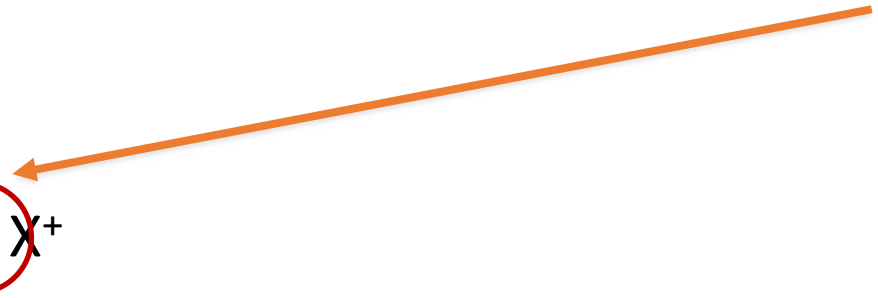
What we will see next

- Closures Part 2
- Superkeys & Keys
- Practice: The key or a key?

Why Do We Need the Closure?

- With closure we can find all FD's easily

- To check if $X \rightarrow A$
  - Compute X^+
  - Check if $A \in X^+$



Note here that **X** is a *set* of attributes, but **A** is a *single* attribute. Why does considering FDs of this form suffice?

Recall the **Split/combine** rule:
$X \rightarrow A_1, \ldots, X \rightarrow A_n$
implies
$X \rightarrow \{A_1, \ldots, A_n\}$

Using Closure to Infer ALL FDs

Step 1: Compute X^+, for every set of attributes X:

Example:

Given F =

$\{A,B\} \rightarrow C$
$\{A,D\} \rightarrow B$
$\{B\} \rightarrow D$

$\{A\}^+ = \{A\}$
$\{B\}^+ = \{B,D\}$
$\{C\}^+ = \{C\}$
$\{D\}^+ = \{D\}$
$\{A,B\}^+ = \{A,B,C,D\}$
$\{A,C\}^+ = \{A,C\}$
$\{A,D\}^+ = \{A,B,C,D\}$
$\{A,B,C\}^+ = \{A,B,D\}^+ = \{A,C,D\}^+ = \{A,B,C,D\}$
$\{B,C,D\}^+ = \{B,C,D\}$
$\{A,B,C,D\}^+ = \{A,B,C,D\}$

No need to compute all of these- why?

Using Closure to Infer ALL FDs

Example:

Given F =

$\{A,B\} \rightarrow C$
$\{A,D\} \rightarrow B$
$\{B\} \rightarrow D$

Step 1: Compute X^+, for every set of attributes X:

$\{A\}^+ = \{A\}$, $\{B\}^+ = \{B,D\}$, $\{C\}^+ = \{C\}$, $\{D\}^+ = \{D\}$, $\{A,B\}^+ = \{A,B,C,D\}$, $\{A,C\}^+ = \{A,C\}$, $\{A,D\}^+ = \{A,B,C,D\}$, $\{A,B,C\}^+ = \{A,B,D\}^+ = \{A,C,D\}^+ = \{A,B,C,D\}$, $\{B,C,D\}^+ = \{B,C,D\}$, $\{A,B,C,D\}^+ = \{A,B,C,D\}$

Step 2: Enumerate all FDs X → Y, s.t. $Y \subseteq X^+$ and $X \cap Y = \emptyset$:

$\{A,B\} \rightarrow \{C,D\}$, $\{A,D\} \rightarrow \{B,C\}$,
$\{A,B,C\} \rightarrow \{D\}$, $\{A,B,D\} \rightarrow \{C\}$,
$\{A,C,D\} \rightarrow \{B\}$

Using Closure to Infer ALL FDs

Example:

Given F =

$\{A,B\} \rightarrow C$
$\{A,D\} \rightarrow B$
$\{B\} \rightarrow D$

Step 1: Compute X^+, for every set of attributes X:

$\{A\}^+ = \{A\}$, $\{B\}^+ = \{B,D\}$, $\{C\}^+ = \{C\}$, $\{D\}^+ = \{D\}$, $\{A,B\}^+ = \{A,B,C,D\}$, $\{A,C\}^+ = \{A,C\}$, $\{A,D\}^+ = \{A,B,C,D\}$, $\{A,B,C\}^+ = \{A,B,D\}^+ = \{A,C,D\}^+ = \{A,B,C,D\}$, $\{B,C,D\}^+ = \{B,C,D\}$, $\{A,B,C,D\}^+ = \{A,B,C,D\}$

Step 2: Enumerate all FDs X → Y, s.t. $Y \subseteq X^+$ and $X \cap Y = \emptyset$:

"Y is in the closure of X"

$\{A,B\} \rightarrow \{C,D\}$, $\{A,D\} \rightarrow \{B,C\}$, $\{A,B,C\} \rightarrow \{D\}$, $\{A,B,D\} \rightarrow \{C\}$, $\{A,C,D\} \rightarrow \{B\}$

Using Closure to Infer ALL FDs

Example:

Given F =

$\{A,B\} \rightarrow C$
$\{A,D\} \rightarrow B$
$\{B\} \rightarrow D$

Step 1: Compute X^+, for every set of attributes X:

$\{A\}^+ = \{A\}$, $\{B\}^+ = \{B,D\}$, $\{C\}^+ = \{C\}$, $\{D\}^+ = \{D\}$, $\{A,B\}^+ = \{A,B,C,D\}$, $\{A,C\}^+ = \{A,C\}$, $\{A,D\}^+ = \{A,B,C,D\}$, $\{A,B,C\}^+ = \{A,B,D\}^+ = \{A,C,D\}^+ = \{A,B,C,D\}$, $\{B,C,D\}^+ = \{B,C,D\}$, $\{A,B,C,D\}^+ = \{A,B,C,D\}$

Step 2: Enumerate all FDs $X \rightarrow Y$, s.t. $Y \subseteq X^+$ and $X \cap Y = \emptyset$:

The FD $X \rightarrow Y$ is non-trivial

$\{A,B\} \rightarrow \{C,D\}$, $\{A,D\} \rightarrow \{B,C\}$,
~~$\{A,B,C\} \rightarrow \{D\}$~~, $\{A,B,D\} \rightarrow \{C\}$,
~~$\{A,C,D\} \rightarrow \{B\}$~~

Keys and Superkeys

A **superkey** is a set of attributes $A_1, \ldots, A_n$ s.t. for *any other* attribute **B** in R, we have $\{A_1, \ldots, A_n\} \rightarrow B$

I.e. all attributes are *functionally determined* by a superkey

A **key** is a *minimal* superkey (also called "candidate key")

This means that no subset of a key is also a superkey (i.e., dropping any attribute from the key makes it no longer a superkey)

Finding Keys and Superkeys

- For each set of attributes X
  - Compute X^+
  - If X^+ = set of all attributes then X is a superkey
  - If X is minimal, then it is a key

Example of Finding Keys

```
Product(name, price, category, color)
```

```
{name, category} → price
{category} → color
```

What is a key?

Example of Finding Keys

```
Product(name, price, category, color)
```

```
{name, category} → price
{category} → color
```

$\{name, category\}^+ = \{name, price, category, color\}$

= the set of all attributes

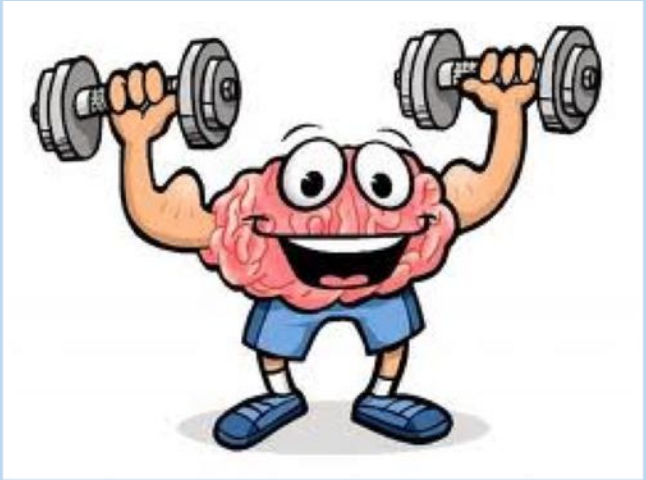
$\Longrightarrow$ this is a **superkey**

$\Longrightarrow$ this is a **key**, since neither `name` nor `category` alone is a superkey

Practice

- Activity-21.ipynb

Complete Normalization Practice!



Example: DreamHome Rental

StaffPropertyInspection

propertyNo	pAddress	iDate	iTime	comments	staffNo	sName	carReg
PG4	6 Lawrence St, Glasgow	18-Oct-03	10:00	need to replace crockery	SG37	Ann Beech	M231 JGR
		22-Apr-04	09:00	in good order	SG14	David Ford	M533 HDR
		1-Oct-04	12:00	damp rot in bathroom	SG14	David Ford	N721 HFR
PG16	5 Novar Dr, Glasgow	22-Apr-04	13:00	replace living room carpet	SG14	David Ford	M533 HDR
		24-Oct-04	14:00	good condition	SG37	Ann Beech	N721 HFR

Can a database store this information?
Is it in 1NF?

Members of DreamHome inspect properties

- When staff are required to undertake these inspections, they are allocated a company car for use on the day of the inspections. (One car per person & day)
- However, a car may be allocated to several members of staff as required throughout the working day.
- A member of staff may inspect several properties on a given date, but a property is only inspected once on a given date.

Source: Connolly, Begg: Database systems, 4th ed, p. 423, 2005.

Example: DreamHome Rental

StaffPropertyInspection

propertyNo	iDate	iTime	pAddress	comments	staffNo	sName	carReg
PG4	18-Oct-03	10:00	6 Lawrence St, Glasgow	need to replace crockery	SG37	Ann Beech	M231 JGR
PG4	22-Apr-04	09:00	6 Lawrence St, Glasgow	in good order	SG14	David Ford	M533 HDR
PG4	1-Oct-04	12:00	6 Lawrence St, Glasgow	damp rot in bathroom	SG14	David Ford	N721 HFR
PG16	22-Apr-04	13:00	5 Novar Dr, Glasgow	replace living room carpet	SG14	David Ford	M533 HDR
PG16	24-Oct-04	14:00	5 Novar Dr, Glasgow	good condition	SG37	Ann Beech	N721 HFR

No! Only now a database can store the information: 1NF
But we still need a primary key

Members of DreamHome inspect properties

- When staff are required to undertake these inspections, they are allocated a company car for use on the day of the inspections. (One car per person & day)
- However, a car may be allocated to several members of staff as required throughout the working day.
- A member of staff may inspect several properties on a given date, but a property is only inspected once on a given date.

Source: Connolly, Begg: Database systems, 4th ed, p. 423, 2005.

Example: DreamHome Rental

StaffPropertyInspection

propertyNo	iDate	iTime	pAddress	comments	staffNo	sName	carReg
PG4	18-Oct-03	10:00	6 Lawrence St, Glasgow	need to replace crockery	SG37	Ann Beech	M231 JGR
PG4	22-Apr-04	09:00	6 Lawrence St, Glasgow	in good order	SG14	David Ford	M533 HDR
PG4	1-Oct-04	12:00	6 Lawrence St, Glasgow	damp rot in bathroom	SG14	David Ford	N721 HFR
PG16	22-Apr-04	13:00	5 Novar Dr, Glasgow	replace living room carpet	SG14	David Ford	M533 HDR
PG16	24-Oct-04	14:00	5 Novar Dr, Glasgow	good condition	SG37	Ann Beech	N721 HFR

Now 1NF + PK

Members of DreamHome inspect properties

- When staff are required to undertake these inspections, they are allocated a company car for use on the day of the inspections. (One car per person & day)
- However, a car may be allocated to several members of staff as required throughout the working day.
- A member of staff may inspect several properties on a given date, but a property is only inspected once on a given date.

Source: Connolly, Begg: Database systems, 4th ed, p. 423, 2005.

Example: DreamHome Rental

StaffPropertyInspection

propertyNo	iDate	iTime	pAddress	comments	staffNo	sName	carReg

Draw all FDs

Members of DreamHome inspect properties

- When staff are required to undertake these inspections, they are allocated a company car for use on the day of the inspections. (One car per person & day)
- However, a car may be allocated to several members of staff as required throughout the working day.
- A member of staff may inspect several properties on a given date, but a property is only inspected once on a given date.

Source: Connolly, Begg: Database systems, 4th ed, p. 423, 2005.

Example: DreamHome Rental

StaffPropertyInspection

propertyNo	iDate	iTime	pAddress	comments	staffNo	sName	carReg

(full, PK)

Members of DreamHome inspect properties

- When staff are required to undertake these inspections, they are allocated a company car for use on the day of the inspections. (One car per person & day)
- However, a car may be allocated to several members of staff as required throughout the working day.
- A member of staff may inspect several properties on a given date, but a property is only inspected once on a given date.

Source: Connolly, Begg: Database systems, 4th ed, p. 423, 2005.

Example: DreamHome Rental

StaffPropertyInspection

propertyNo	iDate	iTime	pAddress	comments	staffNo	sName	carReg

(full, PK)

(partial)

(transitive)

Members of DreamHome inspect properties

- When staff are required to undertake these inspections, they are allocated a company car for use on the day of the inspections. (One car per person & day)
- However, a car may be allocated to several members of staff as required throughout the working day.
- A member of staff may inspect several properties on a given date, but a property is only inspected once on a given date.

Source: Connolly, Begg: Database systems, 4th ed, p. 423, 2005.

Example: DreamHome Rental

StaffPropertyInspection

propertyNo	iDate	iTime	pAddress	comments	staffNo	sName	carReg

(full, PK)

(partial)

(transitive)

Members of DreamHome inspect properties

- When staff are required to undertake these inspections, they are allocated a company car for use on the day of the inspections. (One car per person & day)
- However, a car may be allocated to several members of staff as required throughout the working day.
- A member of staff may inspect several properties on a given date, but a property is only inspected once on a given date.

Source: Connolly, Begg: Database systems, 4th ed, p. 423, 2005.

Example: DreamHome Rental

StaffPropertyInspection

propertyNo	iDate	iTime	pAddress	comments	staffNo	sName	carReg

(full, PK)

(partial)

(transitive)

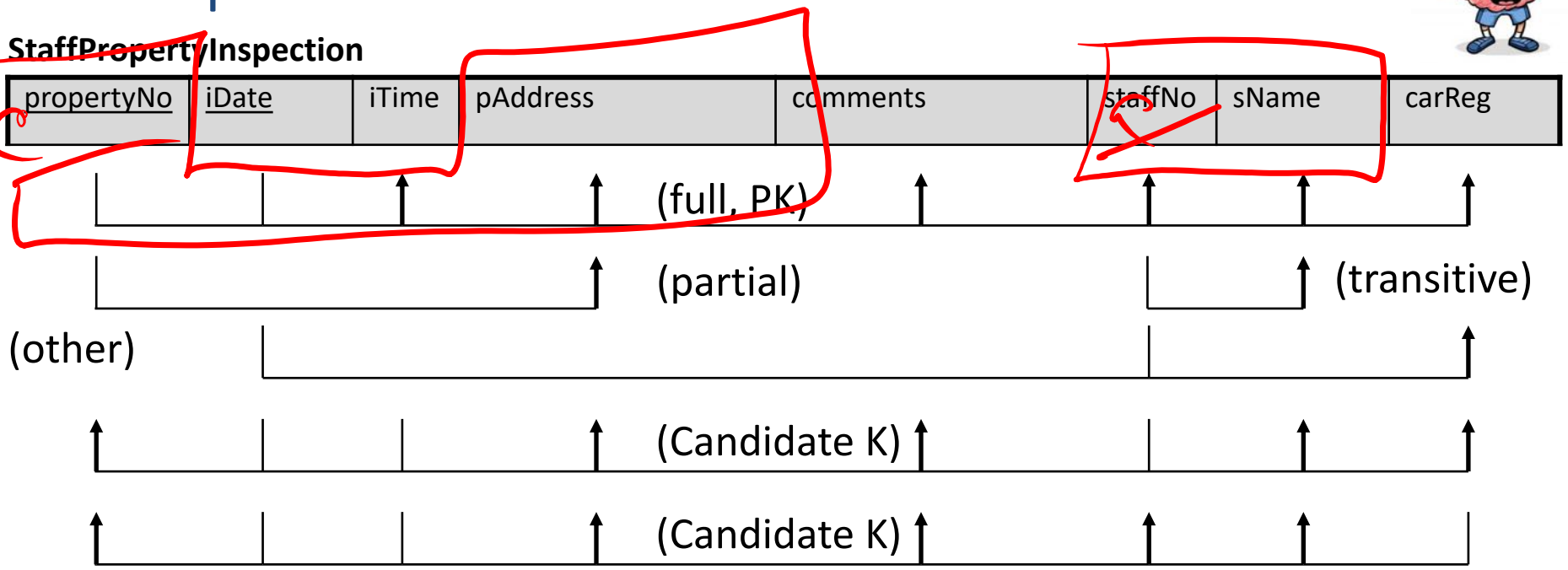
(other)

Members of DreamHome inspect properties

- When staff are required to undertake these inspections, they are allocated a company car for use on the day of the inspections. (One car per person & day)
- However, a car may be allocated to several members of staff as required throughout the working day.
- A member of staff may inspect several properties on a given date, but a property is only inspected once on a given date.

Source: Connolly, Begg: Database systems, 4th ed, p. 423, 2005.

Example: DreamHome Rental

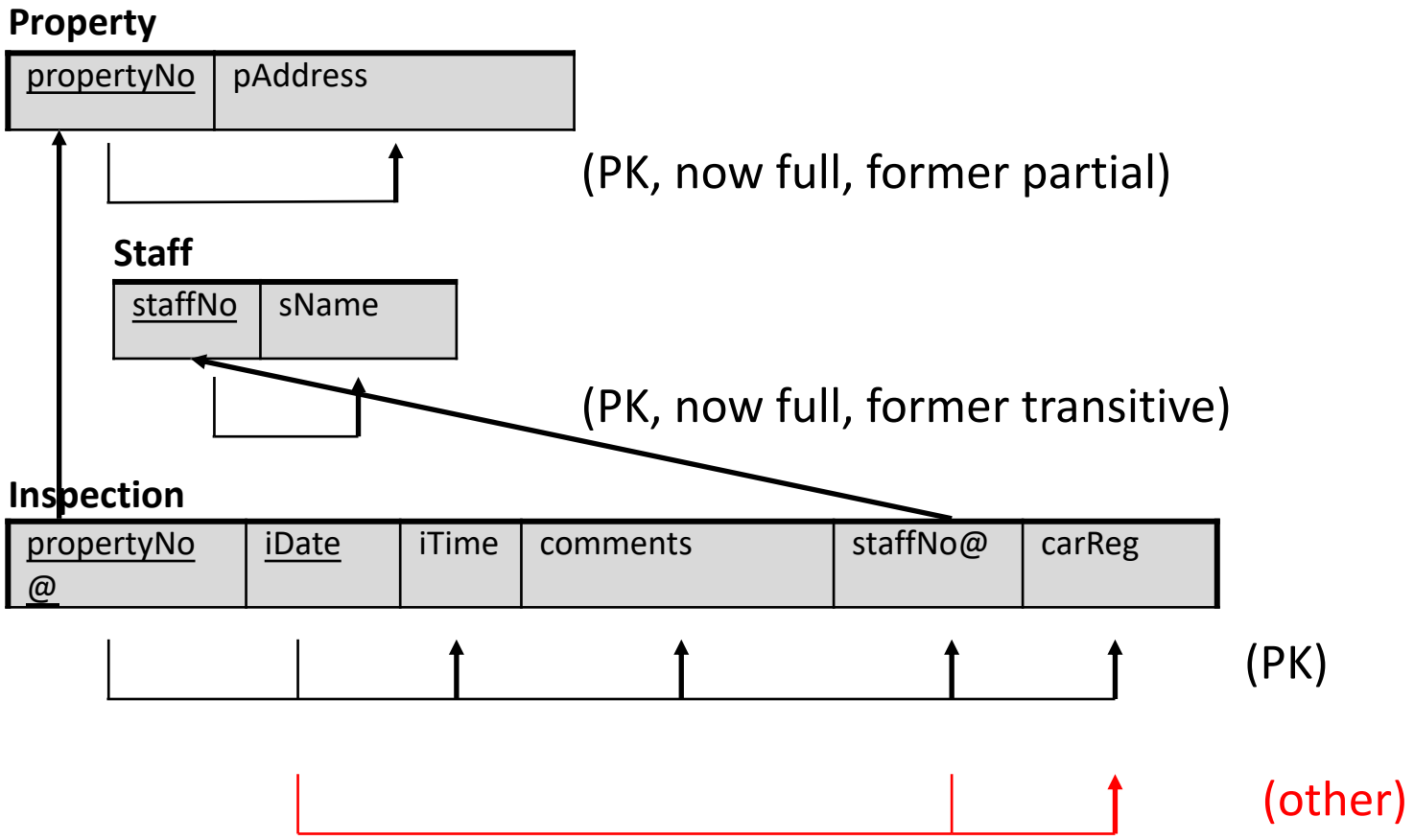


Members of DreamHome inspect properties

- When staff are required to undertake these inspections, they are allocated a company car for use on the day of the inspections. (One car per person & day)
- However, a car may be allocated to several members of staff as required throughout the working day.
- A member of staff may inspect several properties on a given date, but a property is only inspected once on a given date.

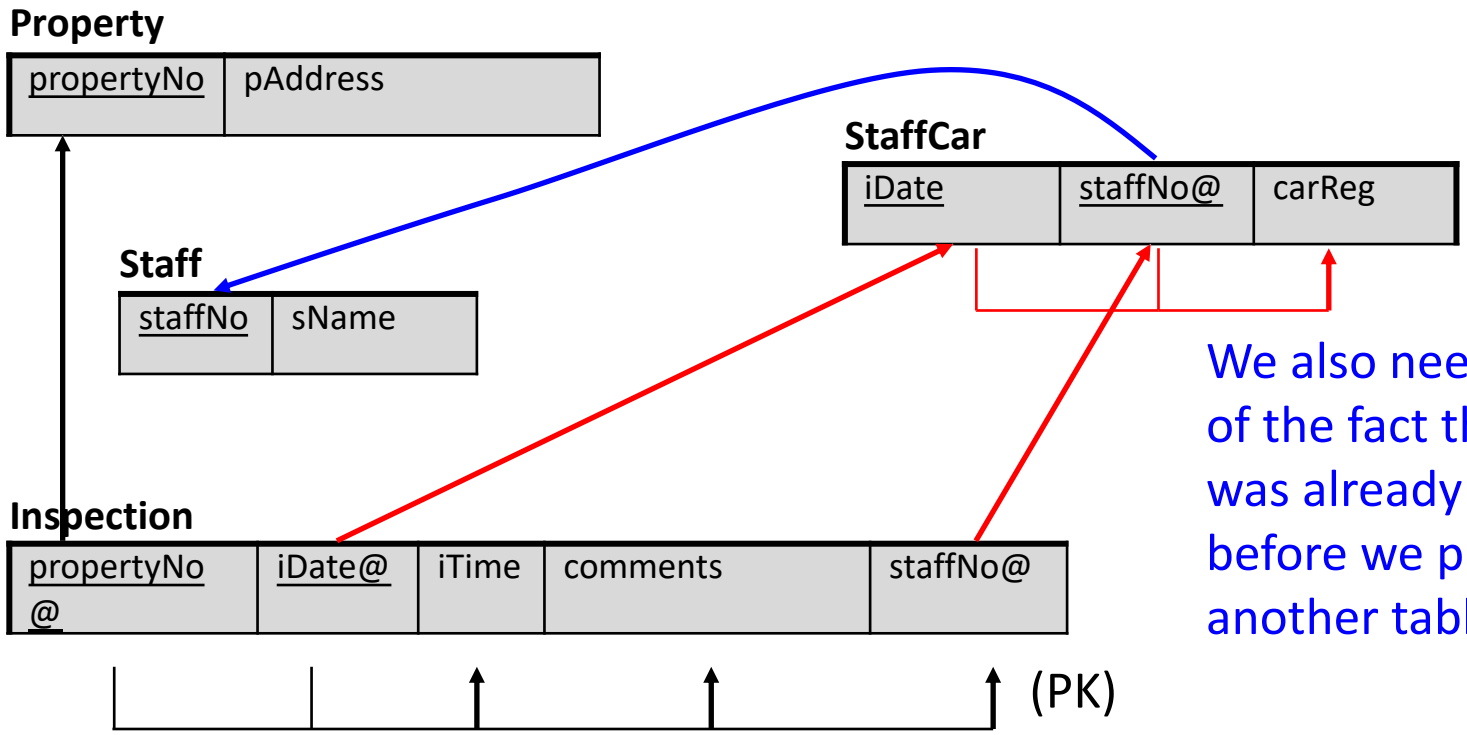
Source: Connolly, Begg: Database systems, 4th ed, p. 423, 2005.

Example: DreamHome Rental



Source: Connolly, Begg: Database systems, 4th ed, p. 423, 2005.

Example: DreamHome Rental

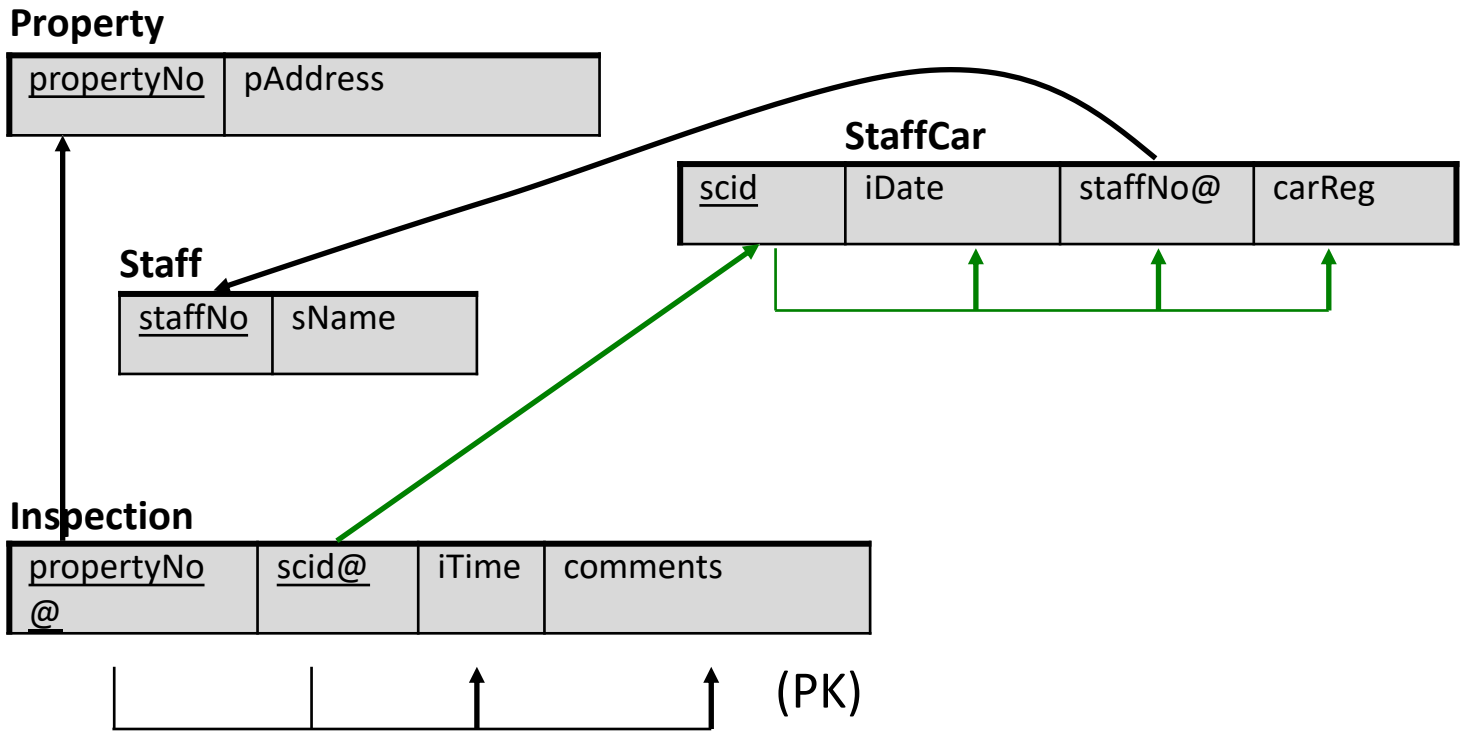


We also need to keep track of the fact that "staffno@" was already a foreign key before we put it into another table

Extra question: We now have a composite FK (idate, staffno) from INSPECTION to STAFFCAR. Thus (idate, staffno) is a composite PK in STAFFCAR.
Assume we like to replace it with a surrogate key. How would the resulting completely normalized tables look like?

Source: Connolly, Begg: Database systems, 4th ed, p. 423, 2005.

Example: DreamHome Rental



This is now fully normalized.
Downside: we need to join INSPECTION with STAFFCAR every time we like to find out about when a property (by "properyNo") was last inspected

Source: Connolly, Begg: Database systems, 4th ed, p. 423, 2005.