B0B36DBS, BD6B36DBS: **Database Systems**

http://www.ksi.mff.cuni.cz/~svoboda/courses/172-B0B36DBS/

Lecture 8

# Functional Dependencies


Authors: **Tomáš Skopal**, **Irena Holubová**

Lecturer: **Martin Svoboda**, martin.svoboda@fel.cvut.cz


17. 4. 2018


**Czech Technical University in Prague**, Faculty of Electrical Engineering

# Today's lecture outline

- motivation
  - data redundancy and update/insertion/deletion anomalies
- functional dependencies
  - Armstrong's axioms
  - attribute and dependency closures
- normal forms
  - 3NF
  - BCNF

# Functional Dependencies

# Motivation

- result of relational design = a set of relational schemas
- problems:
  - data redundancy
    - unnecessary multiple storage of the same data
    - increased space cost
  - insert/update/deletion anomalies
    - insertions and updates must preserve redundant data storage
    - deletion might cause loss of some data
  - null values
    - unnecessary empty space
    - increased space cost
- solution
  - relational schema **normalization**

# Example of “abnormal” schema

| EmpId | Name | Position | Hourly salary | Hours completed |
|---|---|---|---|---|
| 1 | John Goodman | accountant | 200 | 50 |
| 2 | Paul Newman | salesman | 500 | 30 |
| 3 | David Houseman | salesman | 500 | 45 |
| 4 | Brad Pittman | accountant | 200 | 70 |
| 5 | Peter Hitman | accountant | 200 | 66 |
| 6 | Adam Batman | lecturer | 300 | 10 |

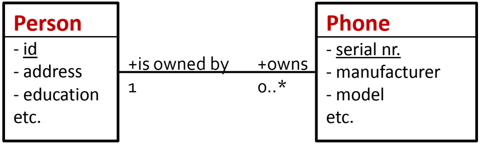
1) From functional analysis we know that position determines hourly salary:
However, hourly salary data is stored multiple times – redundancy.

2) If we delete employee 6, we lose the information on lecturer salary.

3) If we change the accountant hourly salary, we must do that in three places.

# How could this even happen?

- simply
  - during “manual” design of relation schemas
  - badly designed conceptual model
    - e.g., too many attributes in a class




the UML diagram results in 2 tables:

Person(**<u>id</u>**, address, education, ...)
Mobil(**<u>serial nr.</u>**, manufacturer, model, ..., **id**)

# How could this even happen?

| Serial nr. | Manufacturer | Model | Made in | Certificate |
|---|---|---|---|---|
| 13458 | Nokia | Lumia | Finland | EU, USA |
| 34654 | Nokia | Lumia | Finland | EU, USA |
| 65454 | Nokia | Lumia | Finland | EU, USA |
| 45464 | Apple | iPhone 4S | USA | EU, USA |
| 64654 | Samsung | Galaxy S2 | Taiwan | Asia, USA |
| 65787 | Samsung | Galaxy S2 | Taiwan | Asia, USA |

Redundancy in attributes Manufacturer, Model, Made in, Certificate

What happened?
Class Phone includes also other classes – Manufacturer, Model, ...

How to fix it?
Two options 1) fix the UML model (design of more classes)
2) alter the already created schemas (see next)

# Functional dependencies

- attribute-based integrity constraints defined by the user
  - e.g., DB application designer
- a kind of alternative to conceptual modelling
  - ER and UML invented much later
- functional dependency (FD) $X \rightarrow Y$ over schema R(A)
  - mapping $f_i : X_i \rightarrow Y_i$, where $X_i,Y_i \subseteq A$ (where i = 1..number of FDs in R(A))
  - *n*-tuple from $X_i$ **determines** *m*-tuple from $Y_i$
  - *m*-tuple from $Y_i$ **is determined by (is dependent on)** *n*-tuple from $X_i$

# Functional dependencies

- simply, for $X \rightarrow Y$,
  values in X **together determine** the values in Y
- if $X \rightarrow Y$ and $Y \rightarrow X$, then X and Y are **functionally equivalent**
  - could be denoted as $X \leftrightarrow Y$
- if $X \rightarrow a$, where $a \in A$, then $X \rightarrow a$ is an **elementary FD**
  - i.e., only a single attribute on right-hand side
- FDs represent a generalization of the key concept (identifier)
  - key is a special case, see next slides

# Example – wrong interpretation

| EmpId | Name | Position | Hourly salary | Hours completed |
|---|---|---|---|---|
| 1 | John Goodman | accountant | 200 | 50 |
| 2 | Paul Newman | salesman | 500 | 30 |
| 3 | David Houseman | salesman | 500 | 45 |
| 4 | Brad Pittman | accountant | 200 | 70 |
| 5 | Peter Hitman | accountant | 200 | 66 |
| 6 | Adam Batman | lecturer | 300 | 10 |

One might **observe** from the **data**, that:

**Position** $\rightarrow$ **Hourly salary** and also **Hourly salary** $\rightarrow$ **Position**
**EmpId** $\rightarrow$ ***everything***
**Hours completed** $\rightarrow$ ***everything***
**Name** $\rightarrow$ ***everything***

(**but that is nonsense w.r.t. the natural meaning of the attributes**)

# Example – wrong interpretation

| | EmpId | Name | Position | Hourly salary | Hours completed |
|---|---|---|---|---|---|
| | 1 | John Goodman | accountant | 200 | 50 |
| | 2 | Paul Newman | salesman | 500 | 30 |
| | 3 | David Houseman | salesman | 500 | 45 |
| | 4 | Brad Pittman | accountant | 200 | 70 |
| | 5 | Peter Hitman | accountant | 200 | 66 |
| | 6 | Adam Batman | lecturer | 300 | 10 |
| newly inserted records | 7 | Fred Whitman | advisor | 300 | 70 |
| | 8 | Peter Hitman | salesman | 500 | 55 |

**Position → Hourly salary**
**EmpId → *everything***

~~**Hourly salary → Position**~~
~~**Hours completed → *everything***~~
~~**Name → *everything***~~

# Example – correct interpretation

- at first, after the data analysis the FDs are set "forever", **limiting** the content of the tables
  - e.g., **Position → Hourly salary**
    **EmpId → *everything***
  - insertion of the last row is **not allowed** as it violates both the FDs

| EmpId | Name | Position | Hourly salary | Hours completed |
|---|---|---|---|---|
| 1 | John Goodman | accountant | 200 | 50 |
| 2 | Paul Newman | salesman | 500 | 30 |
| 3 | David Houseman | salesman | 500 | 45 |
| 4 | Brad Pittman | accountant | 200 | 70 |
| 5 | Peter Hitman | accountant | 200 | 66 |
| ~~5~~ | ~~Adam Batman~~ | ~~salesman~~ | ~~300~~ | ~~23~~ |

# Armstrong's axioms

Let us have R(A,F). Let X, Y, Z $\subseteq$ A and F is the set of FDs

1) if Y $\subseteq$ X, then X $\rightarrow$ Y (trivial FD)

2) if X $\rightarrow$ Y and Y $\rightarrow$ Z, then X $\rightarrow$ Z (transitivity)

3) if X $\rightarrow$ Y and X $\rightarrow$ Z, then X $\rightarrow$ YZ (composition)

4) if X $\rightarrow$ YZ, then X $\rightarrow$ Y and X $\rightarrow$ Z (decomposition)

# Armstrong's axioms

Armstrong's axioms:

- **are correct (sound)**
  - what is derived from F is valid for any instance from R
- **are complete**
  - all FDs valid in all instances in R (w.r.t. F) can be derived using the axioms
- **1,2,3 (trivial, transitivity, composition) are independent**
  - removal of any axiom 1,2,3 violates the completeness (decomposition could be derived from trivial FD and transitivity)

# Example – deriving FDs

R(A,F)

$A = \{a,b,c,d,e\}$

$F = \{ab \rightarrow c,\ ac \rightarrow d,\ cd \rightarrow ed,\ e \rightarrow f\}$

We could derive, e.g.,:

| | |
|---|---|
| $ab \rightarrow a$ | (trivial) |
| $ab \rightarrow ac$ | (composition with $ab \rightarrow c$) |
| $ab \rightarrow d$ | (transitivity with $ac \rightarrow d$) |
| $ab \rightarrow cd$ | (composition with $ab \rightarrow c$) |
| $ab \rightarrow ed$ | (transitivity with $cd \rightarrow ed$) |
| $ab \rightarrow e$ | (decomposition) |
| $ab \rightarrow f$ | (transitivity) |

# Example – deriving the decomposition rule

R(A,F)

$A = \{a,b,c\}$

$F = \{a \rightarrow bc\}$

Deriving:

$a \rightarrow bc$ (assumption)

$bc \rightarrow b$ (trivial FD)

$bc \rightarrow c$ (trivial FD)

$a \rightarrow b$ (transitivity)

$a \rightarrow c$ (transitivity)

i.e., $\mathbf{a \rightarrow bc \;\Rightarrow\; a \rightarrow b \wedge a \rightarrow c}$

# Closure of set of FDs

- **closure $F^+$** of FDs set $F$ (FD closure) is the set of all FDs derivable from $F$ using the Armstrong's axioms
  - generally exponential size w.r.t. $|F|$

# Example – closure of set of FDs

R(A,F), A = {a,b,c,d}, F = {ab $\rightarrow$ c, cd $\rightarrow$ b, ad $\rightarrow$ c}

$F^+$ =

{a $\rightarrow$ a, b $\rightarrow$ b, c $\rightarrow$ c, d $\rightarrow$ d,
ab $\rightarrow$ a, ab $\rightarrow$ b, ab $\rightarrow$ c,
cd $\rightarrow$ b, cd $\rightarrow$ c, cd $\rightarrow$ d,
ad $\rightarrow$ a, ad $\rightarrow$ c, ad $\rightarrow$ d,
abd $\rightarrow$ a, abd $\rightarrow$ b, abd $\rightarrow$ c, abd $\rightarrow$ d,
abd $\rightarrow$ abcd, ...}

# Cover

- **cover** of a set $\boldsymbol{F}$ is any set of FDs $\boldsymbol{G}$ such that $\boldsymbol{F^+=G^+}$
  - i.e., a set of FDs which have the same closure (= generate the same set of FDs)

- **canonical cover** = cover consisting of **elementary FDs**
  - decompositions are performed to obtain singleton sets on the right-hand side

# Example – cover

R1(A,F), R2(A,G),

A = $\{a,b,c,d\}$,

F = $\{a \rightarrow c,\ b \rightarrow ac,\ d \rightarrow abc\}$,

G = $\{a \rightarrow c,\ b \rightarrow a,\ d \rightarrow b\}$

For checking that $G^+ = F^+$ we do not have to establish the whole covers, it is sufficient to derive F from G, and vice versa, i.e.,

$F' = \{a \rightarrow c,\ b \rightarrow a,\ d \rightarrow b\}$ – decomposition

$G' = \{a \rightarrow c,\ b \rightarrow ac,\ d \rightarrow abc\}$ – transitivity and composition

$\Rightarrow G^+ = F^+$

Schemas R1 and R2 are equivalent because G is cover of F, while they share the attribute set A.

Moreover, G is **minimal cover**, while F is not (for minimal cover see next slides).

# Redundant FDs

- FD $f$ is **redundant** in $F$ if $(F - \{f\})^+ = F^+$
  - i.e., $f$ can be derived from the rest of $F$

- **non-redundant cover** of $F$ = cover of $F$ after removing all redundant FDs
  - note the order of removing FDs matters – a redundant FD could become non-redundant FD after removing another redundant FD
  - i.e., there may exist multiple non-redundant covers of F

# Example – redundant FDs

R(A,F)

A = {a,b,c,d},

F = {$a \rightarrow c$, $b \rightarrow a$, $b \rightarrow c$, $d \rightarrow a$, $d \rightarrow b$, $d \rightarrow c$}

FDs $b \rightarrow c$, $d \rightarrow a$, $d \rightarrow c$ are redundant

after their removal $F^+$ is not changed, i.e., they could be derived from the remaining FDs

$b \rightarrow c$ derived using transitivity $a \rightarrow c$, $b \rightarrow a$

$d \rightarrow a$ derived using transitivity $d \rightarrow b$, $b \rightarrow a$

$d \rightarrow c$ derived using transitivity $d \rightarrow b$, $b \rightarrow a$, $a \rightarrow c$

# Attribute closure, key

- **attribute closure $X^+$** (w.r.t. *F*) is a subset of attributes from *A* determined by *X* (using *F*)
  - consequence: if $X^+ = A$, then X is a **super-key**
- if *F* contains a FD $X \rightarrow Y$ and there exist an attribute *a* in *X* such that $Y \subseteq (X - a)^+$, then *a* is an **attribute redundant in $X \rightarrow Y$**
  - i.e., we do not need *a* in *X* to determine right-hand side *Y*
- **reduced FD** does not contain any redundant attributes
- For R(A) **key** is a set $K \subseteq A$ s.t. it is a super-key (i.e., $K \rightarrow A$) and $K \rightarrow A$ is reduced
  - there could exist multiple keys (at least one)
  - if there is no FD in F, it trivially holds $A \rightarrow A$, i.e., the key is the entire set A
  - **key attribute** = attribute that is in **any** key

# Example – attribute closure

R(A,F), A = {a,b,c,d}, F = {a $\rightarrow$ c, cd $\rightarrow$ b, ad $\rightarrow$ c}

| | | | |
|---|---|---|---|
| {a}+ = {a,c} | it holds | a $\rightarrow$ c | (+ trivial a $\rightarrow$ a) |
| {b}+ = {b} | | | (trivial b $\rightarrow$ b) |
| {c}+ = {c} | | | (trivial c $\rightarrow$ c) |
| {d}+ = {d} | | | (trivial d $\rightarrow$ d) |
| {a,b}+ = {a,b,c} | | a $\rightarrow$ c | (+ trivial) |
| {a,d}+ = {a,b,c,d} | | ad $\rightarrow$ c, cd $\rightarrow$ b | (+ trivial) |
| {c,d}+ = {b,c,d} | | cd $\rightarrow$ b | (+ trivial) |

# Example – redundant attribute

R(A,F), A = {i,j,k,l,m},
F = {m $\rightarrow$ k, lm $\rightarrow$ j, **ijk $\rightarrow$ l**, j $\rightarrow$ m, l $\rightarrow$ i, l $\rightarrow$ k}

**Hypothesis:**

**k** is redundant in **ijk $\rightarrow$ l**, i.e., it holds **ij $\rightarrow$ l**

**Proof:**

1. based on the hypothesis let's construct FD **ij $\rightarrow$ ?**
2. note that **ijk $\rightarrow$ l** remains in F because we **ADD** new FD **ij $\rightarrow$ ?**
   $\Rightarrow$ so we can use **ijk $\rightarrow$ l** for construction of the attribute closure $\{i,j\}^+$
3. we obtain $\{i,j\}^+$ = {i, j, m, k, l},
   i.e., there exists **ij $\rightarrow$ l** which we add into F (it is the member of $F^+$)
4. now forget how **ij $\rightarrow$ l** got into F
5. because **ijk $\rightarrow$ l** could be trivially derived from **ij $\rightarrow$ l**,
   it is redundant FD and we can remove it from F
6. so, we removed the redundant attribute **k** in **ijk $\rightarrow$ l**

In other words, we transformed the problem of removing redundant attribute on the problem of removing redundant FD.

# FDs vs. attributes

FDs:

- can be redundant
  - "we don't need it"
- can have a closure
  - "all derivable FDs"
- can be elementary
  - "single attribute on the right-hand side"
- can be reduced
  - "no redundancies on the left-hand side"

Attributes:

- can be redundant
  - "we don't need it"
- can have a closure
  - "all derivable attributes"
- can form (super-)keys

# Minimal cover

- **non-redundant canonical cover that consists of only reduced FDs**
  - **i.e. no redundant FDs, no redundant attributes, decomposed FDs**
  - **is constructed by removing redundant attributes in FDs followed by removing of redundant FDs**
    - **i.e., the order matters!!!**

**Example:** abcd $\rightarrow$ e, e $\rightarrow$ d, a $\rightarrow$ b, ac $\rightarrow$ d

**Correct order of reduction:**

1. b,d are redundant in abcd $\rightarrow$ e, i.e., removing them
2. ac $\rightarrow$ d is redundant

**Wrong order of reduction:**

1. no redundant FD
2. redundant b,d in abcd $\rightarrow$ e

(now not a minimal cover, because ac $\rightarrow$ d is redundant)

# Keys

# Determining (first) key

- redundant attributes are iteratively removed from left-hand side of trivial FD $A \to A$

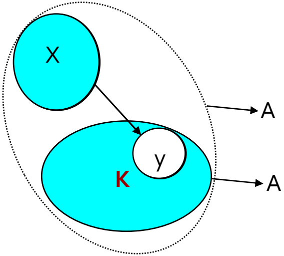
```
algorithm GetFirstKey(set of deps. F, set of attributes A)
   : returns a key K;
   return ReduceAttributes(F, A → A);
```

Note: Because multiple keys can exists, the algorithm finds only one of them.

Which one? It depends on the traversing of the attribute set within the algorithm `ReduceAttributes`.

# Determining all keys, the principle

Let us have a schema S(A, F).
Simplify *F* to minimal cover.




1. Find any key **K** (see the previous slide).
2. Take a FD **X** $\rightarrow$ **y** in *F* such that **y** $\in$ **K** or terminate if not exists (there is no other key).
3. Because **X** $\rightarrow$ **y** and **K** $\rightarrow$ **A**, it transitively holds also **X**{**K** – **y**} $\rightarrow$ **A**, i.e., **X**{**K** – **y**} is super-key.
4. Reduce FD **X**{**K** – **y**} $\rightarrow$ **A** so we obtain key **K'** on the left-hand side. This key is surely different from **K** (we removed **y**).
5. If **K'** is not among the determined keys so far, we add it, declare **K**=**K'** and continue from step 2. Otherwise we finish.

# Determining all keys, the algorithm

- Formally: **Lucchesi-Osborn algorithm**
  - having an already determined key, we search for equivalent sets of attributes, i.e., other keys
- NP-complete problem (theoretically exponential number of keys/FDs)

```
algorithm GetAllKeys(set of FDs F, set of attr. A)
   : returns set of all keys Keys;
   let all dependencies in F be non-trivial
   K := GetFirstKey(F, A);
   Keys := {K};
   for each K in Keys do
      for each X → Y in F do
         if (Y ∩ K ≠ ∅ and ¬∃K′ ∈ Keys : K′ ⊆ (K ∪ X) – Y) then
            N := ReduceAttributes(F, ((K ∪ X) – Y) → A);
            Keys := Keys  ∪ {N};
         endif
      endfor
   endfor
return Keys;
```

# Example – determining all keys

Contracts(A, F)

A = {c = ContractId, s = SupplierId, j = ProjectId, d = DeptId,
p = PartId, q = Quantity, v = Value}

F = {**c** → *all*, **sd** → **p**, **p** → **d**, **jp** → **c**, **j** → **s**}

1. Determine the first key – Keys = {**c**}
2. Iteration 1: take **jp** → **c** that has a part of the last key on the right-hand side (in this case the whole key – **c**) and **jp** is not a super-set of already determined key
3. **jp** → *all* is reduced (no redundant attribute), i.e.,

   Keys = {**c, jp**}
4. Iteration 2: take **sd** → **p** that has a part of the last key on the right-hand side (**jp**),

   {**jsd**} is not a super-set of **c** nor **jp**, i.e., it is a key candidate
5. in **jsd** → *all* we get redundant attribute **s**, i.e.,

   Keys = {**c, jp, jd**}
6. Iteration 3: take **p** → **d**, however, **jp** was already found so we do not add it
7. Finish as the iteration 3 resulted in no key addition.

# Normal Forms

# First normal form (1NF)

Every attribute in a relational schema is of **simple non-structured type**.

- 1NF is the basic condition on „flat database“
- a table is really two-dimensional array
  - not involving arrays, subtables, trees, structures, …

# Example – 1NF

Person(Id: **Integer**, Name: **String**, Birth: **Date**)

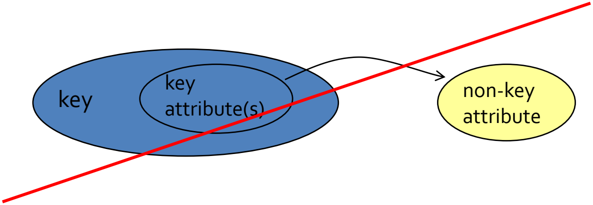
**is in 1NF**

Employee(Id: **Integer**, Subordinate : **Person[ ]**, Boss : **Person**)

**not in 1NF**
(nested table of type Person in attribute Subordinate, and the Boss attribute is structured)

# 2nd normal form (2NF)

- there **do not exist** <u>partial dependencies</u> of non-key attributes on (any) key, i.e.,
  it holds $\forall x \in NK \nexists KK : KK \rightarrow x$
  - where NK is the set of non-key attributes, and
  - KK is <u>subset</u> of some key



# Example – 2NF

| Company | DB server | HQ | Purchase date |
|---|---|---|---|
| John's firm | Oracle | Paris | 1995 |
| John's firm | MS SQL | Paris | 2001 |
| Paul's firm | IBM DB2 | London | 2004 |
| Paul's firm | MS SQL | London | 2002 |
| Paul's firm | Oracle | London | 2005 |

← **not in 2NF**, because HQ is determined by a part of key (Company)

consequence:
**redundancy of HQ values**

**Company, DB Server → *everything***
**Company → HQ**

both schemas **are in 2NF** →

| Company | DB server | Purchase date |
|---|---|---|
| John's firm | Oracle | 1995 |
| John's firm | MS SQL | 2001 |
| Paul's firm | IBM DB2 | 2004 |
| Paul's firm | MS SQL | 2002 |
| Paul's firm | Oracle | 2005 |

**Company, DB Server → *everything***

| Company | HQ |
|---|---|
| John's firm | Paris |
| Paul's firm | London |

**Company → HQ**

# Transitive dependency on key

- FD $A \rightarrow B$ such that $A \not\rightarrow$ *some key*
  (A is not a super-key), i.e., we get transitivity *key* $\rightarrow A \rightarrow B$

- i.e., unique values of ***key*** are mapped to the same or **less** unique values of **A**, and those are mapped to the same or **less** unique values of **B**

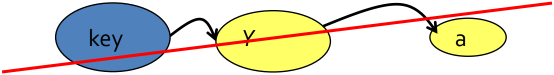
Example in 2NF:
ZIPcode → City → Country

| ZIPcode | City | Country |
|---|---|---|
| CZ 118 00 | Prague | Czech rep. |
| CZ 190 00 | Prague | Czech rep. |
| CZ 772 00 | Olomouc | Czech rep. |
| CZ 783 71 | Olomouc | Czech rep. |
| SK 911 01 | Trenčín | Slovak rep. |

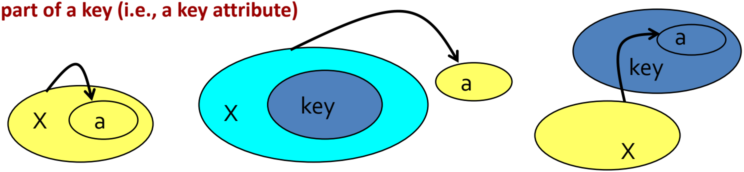
no redundancy medium redundancy high redundancy

# 3rd normal form (3NF)

- <u>non-key attributes</u> **are not transitively dependent** on key




- note: as the 3NF using the above definition cannot be tested without construction of $F^+$, we use a definition that assumes only R(A,F):
- <u>at least one</u> condition holds for each FD $X \rightarrow a$ (where $X \subseteq A$, $a \in A$):
  - **FD is trivial**
  - **X is super-key**
  - ***a* is part of a key (i.e., a key attribute)**



# Example – 3NF

| Company | HQ | ZIPcode |
|---|---|---|
| John's firm | Prague | CZ 11800 |
| Paul's firm | Ostrava | CZ 70833 |
| Martin's firm | Brno | CZ 22012 |
| David's firm | Prague | CZ 11000 |
| Peter's firm | Brno | CZ 22012 |

**Company → *everything***
**ZIPcode → HQ**

is in 2NF, **not in 3NF** (transitive dependency of HQ on key through ZIPcode)

consequence:
**redundancy of HQ values**

**Company → *everything***
**ZIPcode → *everything***

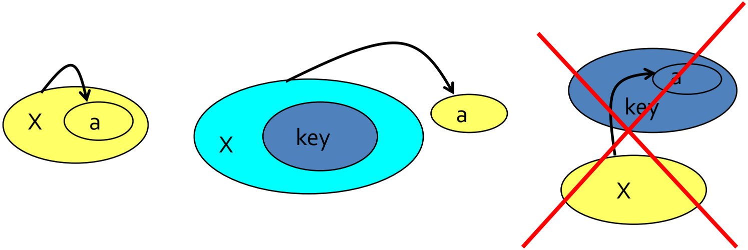
both schemas **are in 3NF**

| Company | ZIPcode |
|---|---|
| John's firm | CZ 11800 |
| Paul's firm | CZ 70833 |
| Martin's firm | CZ 22012 |
| David's firm | CZ 11000 |
| Peter's firm | CZ 22012 |

| ZIPcode | HQ |
|---|---|
| CZ 11800 | Prague |
| CZ 70833 | Ostrava |
| CZ 22012 | Brno |
| CZ 11000 | Prague |

# Boyce-Codd normal form (BCNF)

- **every attribute** is (non-transitively) dependent on key
- more exactly, in a given schema R(A, F) there holds <u>at least one</u> condition for each FD $X \rightarrow a$ (where $X \subseteq A$, $a \in A$):
  - **FD is trivial**
  - **X is super-key**
- note: the same as 3NF without the last option ($a$ is key attribute)



# Example – BCNF

| Destination | Pilot | Plane | Day |
|---|---|---|---|
| Paris | cpt. Oiseau | Boeing #1 | Monday |
| Paris | cpt. Oiseau | Boeing #2 | Tuesday |
| Berlin | cpt. Vogel | Airbus #1 | Monday |

**Pilot, Day → *everything***
**Plane, Day → *everything***
**Destination → Pilot**

is in 3NF, **not in BCNF**
(Pilot is determined by Destination, which is not a super-key)

consequence:
**redundancy of Pilot values**

**Destination → Pilot**
**Plane, Day → *everything***

| Destination | Pilot |
|---|---|
| Paris | cpt. Oiseau |
| Berlin | cpt. Vogel |

| Destination | Plane | Day |
|---|---|---|
| Paris | Boeing #1 | Monday |
| Paris | Boeing #2 | Tuesday |
| Berlin | Airbus #1 | Monday |

both schemas **are in BCNF**