Lecture 03:

Functional Dependencies

Authors: Tomáš Skopal, Irena Holubová
Lecturer: Martin Svoboda
7. 3. 2017
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

<table>
<thead>
<tr>
<th>EmpId</th>
<th>Name</th>
<th>Position</th>
<th>Hourly salary</th>
<th>Hours completed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John Goodman</td>
<td>accountant</td>
<td>200</td>
<td>50</td>
</tr>
<tr>
<td>2</td>
<td>Paul Newman</td>
<td>salesman</td>
<td>500</td>
<td>30</td>
</tr>
<tr>
<td>3</td>
<td>David Houseman</td>
<td>salesman</td>
<td>500</td>
<td>45</td>
</tr>
<tr>
<td>4</td>
<td>Brad Pittman</td>
<td>accountant</td>
<td>200</td>
<td>70</td>
</tr>
<tr>
<td>5</td>
<td>Peter Hitman</td>
<td>accountant</td>
<td>200</td>
<td>66</td>
</tr>
<tr>
<td>6</td>
<td>Adam Batman</td>
<td>lecturer</td>
<td>300</td>
<td>10</td>
</tr>
</tbody>
</table>

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(id, address, education, ...)
Mobil(serial nr., manufacturer, model, ..., id)
How could this even happen?

<table>
<thead>
<tr>
<th>Serial nr.</th>
<th>Manufacturer</th>
<th>Model</th>
<th>Made in</th>
<th>Certificate</th>
</tr>
</thead>
<tbody>
<tr>
<td>13458</td>
<td>Nokia</td>
<td>Lumia</td>
<td>Finland</td>
<td>EU, USA</td>
</tr>
<tr>
<td>34654</td>
<td>Nokia</td>
<td>Lumia</td>
<td>Finland</td>
<td>EU, USA</td>
</tr>
<tr>
<td>65454</td>
<td>Nokia</td>
<td>Lumia</td>
<td>Finland</td>
<td>EU, USA</td>
</tr>
<tr>
<td>45464</td>
<td>Apple</td>
<td>iPhone 4S</td>
<td>USA</td>
<td>EU, USA</td>
</tr>
<tr>
<td>64654</td>
<td>Samsung</td>
<td>Galaxy S2</td>
<td>Taiwan</td>
<td>Asia, USA</td>
</tr>
<tr>
<td>65787</td>
<td>Samsung</td>
<td>Galaxy S2</td>
<td>Taiwan</td>
<td>Asia, USA</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>EmpId</th>
<th>Name</th>
<th>Position</th>
<th>Hourly salary</th>
<th>Hours completed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John Goodman</td>
<td>accountant</td>
<td>200</td>
<td>50</td>
</tr>
<tr>
<td>2</td>
<td>Paul Newman</td>
<td>salesman</td>
<td>500</td>
<td>30</td>
</tr>
<tr>
<td>3</td>
<td>David Houseman</td>
<td>salesman</td>
<td>500</td>
<td>45</td>
</tr>
<tr>
<td>4</td>
<td>Brad Pittman</td>
<td>accountant</td>
<td>200</td>
<td>70</td>
</tr>
<tr>
<td>5</td>
<td>Peter Hitman</td>
<td>accountant</td>
<td>200</td>
<td>66</td>
</tr>
<tr>
<td>6</td>
<td>Adam Batman</td>
<td>lecturer</td>
<td>300</td>
<td>10</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>EmpId</th>
<th>Name</th>
<th>Position</th>
<th>Hourly salary</th>
<th>Hours completed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John Goodman</td>
<td>accountant</td>
<td>200</td>
<td>50</td>
</tr>
<tr>
<td>2</td>
<td>Paul Newman</td>
<td>salesman</td>
<td>500</td>
<td>30</td>
</tr>
<tr>
<td>3</td>
<td>David Houseman</td>
<td>salesman</td>
<td>500</td>
<td>45</td>
</tr>
<tr>
<td>4</td>
<td>Brad Pittman</td>
<td>accountant</td>
<td>200</td>
<td>70</td>
</tr>
<tr>
<td>5</td>
<td>Peter Hitman</td>
<td>accountant</td>
<td>200</td>
<td>66</td>
</tr>
<tr>
<td>6</td>
<td>Adam Batman</td>
<td>lecturer</td>
<td>300</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>Fred Whitman</td>
<td>advisor</td>
<td>300</td>
<td>70</td>
</tr>
<tr>
<td>8</td>
<td>Peter Hitman</td>
<td>salesman</td>
<td>500</td>
<td>55</td>
</tr>
</tbody>
</table>

**Newly inserted records**

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

**Correct functional dependencies**

- **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

<table>
<thead>
<tr>
<th>EmpId</th>
<th>Name</th>
<th>Position</th>
<th>Hourly salary</th>
<th>Hours completed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John Goodman</td>
<td>accountant</td>
<td>200</td>
<td>50</td>
</tr>
<tr>
<td>2</td>
<td>Paul Newman</td>
<td>salesman</td>
<td>500</td>
<td>30</td>
</tr>
<tr>
<td>3</td>
<td>David Houseman</td>
<td>salesman</td>
<td>500</td>
<td>45</td>
</tr>
<tr>
<td>4</td>
<td>Brad Pittman</td>
<td>accountant</td>
<td>200</td>
<td>70</td>
</tr>
<tr>
<td>5</td>
<td>Peter Hitman</td>
<td>accountant</td>
<td>200</td>
<td>66</td>
</tr>
<tr>
<td>5</td>
<td>Adam Batman</td>
<td>salesman</td>
<td>300</td>
<td>23</td>
</tr>
</tbody>
</table>
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 → c, ac → d, cd → ed, e → f}

We could derive, e.g.,:

- ab → a  (trivial)
- ab → ac  (composition with ab → c)
- ab → d   (transitivity with ac → d)
- ab → cd  (composition with ab → c)
- ab → ed  (transitivity with cd → ed)
- ab → e   (decomposition)
- ab → f   (transitivity)
Example – deriving the decomposition rule

\( R(A,F) \)

\[
\begin{align*}
A &= \{a,b,c\} \\
F &= \{a \rightarrow bc\}
\end{align*}
\]

Deriving:

\[
\begin{align*}
a \rightarrow bc & \quad (\text{assumption}) \\
bc \rightarrow b & \quad (\text{trivial FD}) \\
bc \rightarrow c & \quad (\text{trivial FD}) \\
a \rightarrow b & \quad (\text{transitivity}) \\
a \rightarrow c & \quad (\text{transitivity})
\end{align*}
\]

i.e., \( a \rightarrow bc \Rightarrow a \rightarrow b \land a \rightarrow c \)
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\}

\[ F^+ = \{a \rightarrow a, b \rightarrow b, c \rightarrow c, d \rightarrow d, \]

\[ ab \rightarrow a, ab \rightarrow b, ab \rightarrow c, \]

\[ cd \rightarrow b, cd \rightarrow c, cd \rightarrow d, \]

\[ ad \rightarrow a, ad \rightarrow c, ad \rightarrow d, \]

\[ abd \rightarrow a, abd \rightarrow b, abd \rightarrow c, abd \rightarrow d, \]

\[ abd \rightarrow abcd, \ldots \} \]
Cover

- **cover** of a set $F$ is any set of FDs $G$ such that $F^+ = G^+$
  - 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 $F = \text{cover of } F \text{ after removing all redundant FDs}$
  - 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 → c, b → a, b → c, d → a, d → b, d → c}

FDs b → c, d → a, d → c are redundant

after their removal $F^+$ is not changed, i.e., they could be derived from the remaining FDs
b → c  derived using transitivity  a → c, b → a
d → a  derived using transitivity  d → b, b → a
d → c  derived using transitivity  d → b, b → a, a → 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\} \quad \text{it holds} \quad a \rightarrow c \quad (+ \text{trivial } a \rightarrow a)

\{b\}^+ = \{b\} \quad \text{(trivial } b \rightarrow b)\)

\{c\}^+ = \{c\} \quad \text{(trivial } c \rightarrow c)\)

\{d\}^+ = \{d\} \quad \text{(trivial } d \rightarrow d)\)

\{a,b\}^+ = \{a,b,c\} \quad a \rightarrow c \quad (+ \text{trivial})

\{a,d\}^+ = \{a,b,c,d\} \quad ad \rightarrow c, \ cd \rightarrow b \quad (+ \text{trivial})

\{c,d\}^+ = \{b,c,d\} \quad cd \rightarrow b \quad (+ \text{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 \( ij \rightarrow ? \)
   \[ \Rightarrow \text{so we can use } ijk \rightarrow l \text{ for construction of the attribute closure } \{i,j\}^+ \]
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 → e, e → d, a → b, ac → d

**Correct order of reduction:**
1. b,d are redundant in abcd → e, i.e., removing them
2. ac → d is redundant

**Wrong order of reduction:**
1. no redundant FD
2. redundant b,d in abcd → e
   (now not a minimal cover, because ac → d is redundant)
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)
2\textsuperscript{nd} normal form (2NF)

- there \textbf{do not exist} partial dependencies 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 \textit{subset} of some key

![Diagram showing the relationship between keys, key attributes, and non-key attributes]
Example – 2NF

<table>
<thead>
<tr>
<th>Company</th>
<th>DB server</th>
<th>HQ</th>
<th>Purchase date</th>
</tr>
</thead>
<tbody>
<tr>
<td>John’s firm</td>
<td>Oracle</td>
<td>Paris</td>
<td>1995</td>
</tr>
<tr>
<td>John’s firm</td>
<td>MS SQL</td>
<td>Paris</td>
<td>2001</td>
</tr>
<tr>
<td>Paul’s firm</td>
<td>IBM DB2</td>
<td>London</td>
<td>2004</td>
</tr>
<tr>
<td>Paul’s firm</td>
<td>MS SQL</td>
<td>London</td>
<td>2002</td>
</tr>
<tr>
<td>Paul’s firm</td>
<td>Oracle</td>
<td>London</td>
<td>2005</td>
</tr>
</tbody>
</table>

← 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 → everything
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 $\rightarrow$ City $\rightarrow$ Country

<table>
<thead>
<tr>
<th>ZIPcode</th>
<th>City</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>CZ 118 00</td>
<td>Prague</td>
<td>Czech rep.</td>
</tr>
<tr>
<td>CZ 190 00</td>
<td>Prague</td>
<td>Czech rep.</td>
</tr>
<tr>
<td>CZ 772 00</td>
<td>Olomouc</td>
<td>Czech rep.</td>
</tr>
<tr>
<td>CZ 783 71</td>
<td>Olomouc</td>
<td>Czech rep.</td>
</tr>
<tr>
<td>SK 911 01</td>
<td>Trenčín</td>
<td>Slovak rep.</td>
</tr>
</tbody>
</table>

no redundancy  medium redundancy  high redundancy
3rd normal form (3NF)

- **non-key attributes** are not transitively dependent on key

![Diagram showing key, X, and a](attachment:diagram.png)

- note: as the 3NF using the above definition cannot be tested without construction of $F^+$, we use a definition that assumes only $R(A,F)$:
  - at least one 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

<table>
<thead>
<tr>
<th>Company</th>
<th>HQ</th>
<th>ZIPcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>John’s firm</td>
<td>Prague</td>
<td>CZ 11800</td>
</tr>
<tr>
<td>Paul’s firm</td>
<td>Ostrava</td>
<td>CZ 70833</td>
</tr>
<tr>
<td>Martin’s firm</td>
<td>Brno</td>
<td>CZ 22012</td>
</tr>
<tr>
<td>David’s firm</td>
<td>Prague</td>
<td>CZ 11000</td>
</tr>
<tr>
<td>Peter’s firm</td>
<td>Brno</td>
<td>CZ 22012</td>
</tr>
</tbody>
</table>

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
Boyce-Codd normal form (BCNF)

- **every attribute** is (non-transitively) dependent on key
- more exactly, in a given schema $R(A, F)$ there holds \textit{at least one} 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

<table>
<thead>
<tr>
<th>Destination</th>
<th>Pilot</th>
<th>Plane</th>
<th>Day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paris</td>
<td>cpt. Oiseau</td>
<td>Boeing #1</td>
<td>Monday</td>
</tr>
<tr>
<td>Paris</td>
<td>cpt. Oiseau</td>
<td>Boeing #2</td>
<td>Tuesday</td>
</tr>
<tr>
<td>Berlin</td>
<td>cpt. Vogel</td>
<td>Airbus #1</td>
<td>Monday</td>
</tr>
</tbody>
</table>

Pilot, Day $\rightarrow$ *everything*

Plane, Day $\rightarrow$ *everything*

Destination $\rightarrow$ Pilot

is in 3NF, **not in BCNF**

(Pilot is determined by Destination, which is not a super-key)

consequence:

**redundancy of Pilot values**

<table>
<thead>
<tr>
<th>Destination</th>
<th>Pilot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paris</td>
<td>cpt. Oiseau</td>
</tr>
<tr>
<td>Berlin</td>
<td>cpt. Vogel</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Destination</th>
<th>Plane</th>
<th>Day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paris</td>
<td>Boeing #1</td>
<td>Monday</td>
</tr>
<tr>
<td>Paris</td>
<td>Boeing #2</td>
<td>Tuesday</td>
</tr>
<tr>
<td>Berlin</td>
<td>Airbus #1</td>
<td>Monday</td>
</tr>
</tbody>
</table>

Destination $\rightarrow$ Pilot

Plane, Day $\rightarrow$ *everything*

**both schemas are in BCNF**