Functional Dependencies

Gary KL Tam

Department of Computer Science
Swansea University

Overview

A primary goal of database design is to decide what tables to create. Usually there are two principles:

1. Capture all the information that needs to be captured by the underlying application.
2. Achieve the above with little redundancy.

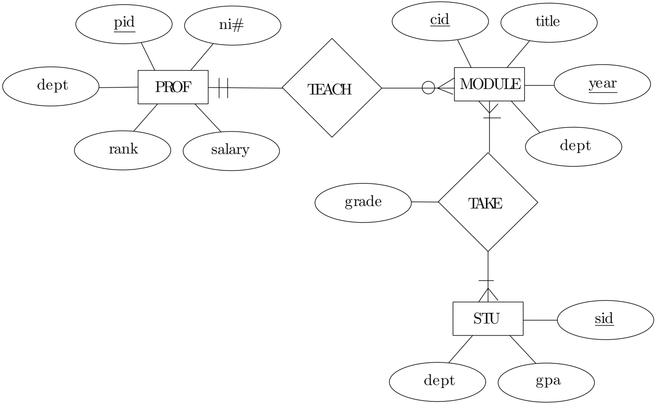
The first principle is enforced with an entity relationship (ER) diagram, while the second with normalization.

This and the next few lectures are devoted to normalization.

Roadmap

1. We need to study functional dependencies.
   - What is FD and its relation to redundancy?
   - Properties of FDs (Axioms, Rules and Closure Test)
   - Powerful Closure Test algorithm in detail.
   - Candidate keys and Closure Test
   - Inferring FDs
2. We proceed to normalization in next set of lecture slides.

Tables created from an ER diagram may still contain redundancy.



- PROF (<u>pid</u>, ni#, dept, rank, salary)
- MODULE (<u>cid, year</u>, title, dept)
- STU (<u>sid</u>, dept, gpa)
- TEACH (pid, <u>cid, year</u>), foreign key (pid), foreign key (cid, year)
- TAKE (<u>cid, year, sid</u>, grade), foreign key (sid), foreign key (cid, year)

Where is redundancy (things that are unnecessarily duplicated in tables)?

Answer:
MODULE (cid, title, year, dept)

Why? Because every time the same course is offered again, its title and department are duplicated.
For example, consider c1 = "CS250"

cid	title	year	dept
$c1$	database	2010	cs
$c1$	database	2011	cs
$c1$	database	2012	cs

The red values are redundant.
Note that the "database" and "cs" of the first tuple are not redundant.
Why?

cid	title	year	dept
$c1$	database	2010	cs
$c1$	database	2011	cs
$c1$	database	2012	cs

Observations: cid implies title, dept. This is written as:

$$\text{cid} \rightarrow \text{title}$$
$$\text{cid} \rightarrow \text{dept}$$

which are called functional dependencies.

Definition

A functional dependency (FD) has the form of $X \rightarrow Y$ (reads: X implies Y), where X and Y are sets of attributes. It means that whenever two tuples are identical on all the attributes in X, they must also be identical on all the attributes in Y.

Alternatively, you can interpret $X \rightarrow Y$ as: each possible value of X can correspond to exactly one value of Y.

Functional dependencies are constraints that are required by the underlying application.

Common examples

- NI $\rightarrow$ Name, Address, Birthdate
  Your national insurance number determines name, address and date of birth
- ISBN $\rightarrow$ Booktitle, Author, Publisher
  wikipedia: The International Standard Book Number (ISBN) is a unique numeric commercial book identifier...

Example

Assume that the following FDs hold:

$$\text{cid} \rightarrow \text{title}$$
$$\text{title} \rightarrow \text{dept}$$
$$\text{cid, year} \rightarrow \text{dept}$$
$$\text{cid, year} \rightarrow \text{cid, dept}$$

Can the following tuples co-exist?

cid	title	year	dept
c1	database	2010	cs
c1	database	2011	cs
c2	database	2012	ee

Example

Assume that the following FDs hold:

$$
\begin{aligned}
\text{cid} &\rightarrow \text{title} \\
\text{title} &\rightarrow \text{dept} \\
\text{cid, year} &\rightarrow \text{dept} \\
\text{cid, year} &\rightarrow \text{cid, dept}
\end{aligned}
$$

Can the following tuples co-exist?

cid	title	year	dept
c1	database	2010	cs
c1	database	2011	cs
c2	database	2012	ee

No, By title $\rightarrow$ dept, last two tuple cannot co-exists

Example

Assume that the following FDs hold:

$$\text{cid} \rightarrow \text{title}$$
$$\text{title} \rightarrow \text{dept}$$
$$\text{cid, year} \rightarrow \text{dept}$$
$$\text{cid, year} \rightarrow \text{cid, dept}$$

Can the following tuples co-exist?

cid	dept
c1	cs
c1	ee

Example

Assume that the following FDs hold:

$$\text{cid} \rightarrow \text{title}$$
$$\text{title} \rightarrow \text{dept}$$
$$\text{cid, year} \rightarrow \text{dept}$$
$$\text{cid, year} \rightarrow \text{cid, dept}$$

Can the following tuples co-exist?

cid	dept
c1	cs
c1	ee

No. cid $\rightarrow$ dept

Example

Assume that the following FDs hold:

$$\text{cid} \rightarrow \text{title}$$
$$\text{title} \rightarrow \text{dept}$$
$$\text{cid, year} \rightarrow \text{dept}$$
$$\text{cid, year} \rightarrow \text{cid, dept}$$

Can the following tuples co-exist?

cid	dept
c1	cs
c1	ee

No. cid $\rightarrow$ dept
Hey, there is no such FD... how do we know?

Properties of Functional Dependencies

We need some helpers.

1. Armstrong's Axioms
2. Spliting Rule
3. Closure Test (powerful)

Armstrong's Axioms

1. (Reflexivity) $X \to Y$ for any $Y \subset X$
2. (Augmentation) $X \to Y$ for any $XZ \to YZ$
3. (Transitivity) $X \to Y$ and $Y \to Z$, then $X \to Z$

Armstrong's Axioms

1. (Reflexivity) $X \to Y$ for any $Y \subset X$

Trivial FD

sid, name $\to$ name

2. (Augmentation) $X \to Y$ for any $XZ \to YZ$

Example

sid $\to$ address, then sid, *name* $\to$ address, *name*

3. (Transitivity) $X \to Y$ and $Y \to Z$, then $X \to Z$

Armstrong's Axioms

1. (Reflexivity) $X \to Y$ for any $Y \subset X$

Trivial FD

sid, name $\to$ name

2. (Augmentation) $X \to Y$ for any $XZ \to YZ$

Example

sid $\to$ address, then sid, *name* $\to$ address, *name*

3. (Transitivity) $X \to Y$ and $Y \to Z$, then $X \to Z$

By transitivity

cid $\to$ title and title $\to$ dept, then cid $\to$ dept

Splitting Right Sides of FD's

- $X \rightarrow A_1A_2...A_N$ holds for R iff 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.

There is a useful & powerful algorithm: Closure Test

There is a useful & powerful algorithm: Closure Test

Think of Closure Test as an algorithm.
Given **input** X and some FDs F, it computes **output** X^+.
Closure of X (X^+) contains all attributes implied by X.

There is a useful & powerful algorithm: Closure Test

Think of Closure Test as an algorithm.
Given **input** X and some FDs F, it computes **output** X^+.
Closure of X (X^+) contains all attributes implied by X.

A peek into closure: does cid $\rightarrow$ dept?

We let X = cid and F be given FDs, and compute X^+ (cid$^+$)

$$\text{output: } \text{cid}^+ = \{\text{cid, title, dept}\}$$

which literally means:

$$\text{cid} \rightarrow \text{cid}$$
$$\text{cid} \rightarrow \text{title}$$
$$\text{cid} \rightarrow \text{dept}$$

There is a useful & powerful algorithm: Closure Test

Think of Closure Test as an algorithm.
Given **input** X and some FDs F, it computes **output** X^+.
Closure of X (X^+) contains all attributes implied by X.

A peek into closure: does cid $\rightarrow$ dept?

We let X = cid and F be given FDs, and compute X^+ (cid$^+$)

$$\text{output: cid}^+ = \{\text{cid, title, dept}\}$$

which literally means:

$$\text{cid} \rightarrow \text{cid}$$
$$\text{cid} \rightarrow \text{title}$$
$$\text{cid} \rightarrow \text{dept}$$

Next slides:

1. Definition of Closure
2. Closure Test algorithm

Closure of an Attribute Set

Let F be the set of functional dependencies required by the underlying application.

Definition

The closure (X^+) of a set X of attributes is the set of all such attributes A that $X \rightarrow A$ can be deducted from F.

Example: Let F be the set of following FDs:

$$\begin{gathered} \text{cid} \rightarrow \text{title} \\ \text{title} \rightarrow \text{dept} \\ \text{cid, year} \rightarrow \text{dept} \\ \text{cid, year} \rightarrow \text{cid, dept} \end{gathered}$$

Let $X =$ cid, and F be the above FDs.
How can we systemically find the following out:
Qn: Is "cid" in cid^+?
Qn: Is "year" in cid^+ too?

Finding the Closure of an Attribute Set

algorithm (F, X)
/* F is a set of FDs, and X is an attribute set */

1. $C = X$
2. **while** F has a FD $A \rightarrow B$ such that $A \subseteq C$ **do**
3. $\quad C = C \cup B$
4. $\quad$ remove $A \rightarrow B$ from F
5. **return** C /* closure of X */

Example: Let F be the set of following FDs:

$$\text{cid} \rightarrow \text{title}$$
$$\text{title} \rightarrow \text{dept}$$
$$\text{cid, year} \rightarrow \text{dept}$$
$$\text{cid, year} \rightarrow \text{cid, dept}$$

What is the closure of X when $X = \{\text{cid}\}$? We apply the algorithm:

algorithm (F, X)
1. $C = X$
2. **while** F has a FD $A \rightarrow B$ such that $A \subseteq C$ **do**
3. $C = C \cup B$
4. remove $A \rightarrow B$ from F
5. **return** C

- Input: $X = \{\text{cid}\}$, F is given FDs.
- $C = \{\text{cid}\}$

Example: Let F be the set of following FDs:

$$\text{cid} \to \text{title}$$
$$\text{title} \to \text{dept}$$
$$\text{cid, year} \to \text{dept}$$
$$\text{cid, year} \to \text{cid, dept}$$

What is the closure of X when X ={cid}? We apply the algorithm:

algorithm (F, X)
1. $C = X$
2. **while** F has a FD $A \to B$ such that $A \subseteq C$ **do**
3. $\quad C = C \cup B$
4. $\quad$ remove $A \to B$ from F
5. **return** C

- Input: X ={cid}, F is given FDs.
- C ={cid}
- By "cid $\to$ title", {cid}$\subseteq C$, thus C ={cid, title}

Example: Let F be the set of following FDs:

$$\text{cid} \rightarrow \text{title}$$
$$\text{title} \rightarrow \text{dept}$$
$$\text{cid, year} \rightarrow \text{dept}$$
$$\text{cid, year} \rightarrow \text{cid, dept}$$

What is the closure of X when X ={cid}? We apply the algorithm:

algorithm (F, X)
1. $C = X$
2. **while** F has a FD $A \rightarrow B$ such that $A \subseteq C$ **do**
3. $C = C \cup B$
4. remove $A \rightarrow B$ from F
5. **return** C

- Input: X ={cid}, F is given FDs.
- C ={cid}
- By "cid $\rightarrow$ title", {cid}$\subseteq C$, thus C ={cid, title}
- By "title $\rightarrow$ dept", {title}$\subseteq C$, C ={cid, title, dept}
- X^+ = {cid, title, dept}

Example: Let F be the set of following FDs:

$$\text{cid} \rightarrow \text{title}$$
$$\text{title} \rightarrow \text{dept}$$
$$\text{cid, year} \rightarrow \text{dept}$$
$$\text{cid, year} \rightarrow \text{cid, dept}$$

What is the closure of X when X ={cid}? We apply the algorithm:

algorithm (F, X)
1. $C = X$
2. **while** F has a FD $A \rightarrow B$ such that $A \subseteq C$ **do**
3. $\quad C = C \cup B$
4. $\quad$ remove $A \rightarrow B$ from F
5. **return** C

- Input: X ={cid}, F is given FDs.
- C ={cid}
- By "cid $\rightarrow$ title", {cid}$\subseteq C$, thus C ={cid, title}
- By "title $\rightarrow$ dept", {title}$\subseteq C$, C ={cid, title, dept}
- X^+ = {cid, title, dept}

Think: Is "year" in the closure of X?
Can "cid $\rightarrow$ year" be deduced from F?

Example: Let F be the set of following FDs:

$$\text{cid} \rightarrow \text{title}$$
$$\text{title} \rightarrow \text{dept}$$
$$\text{cid, year} \rightarrow \text{dept}$$
$$\text{cid, year} \rightarrow \text{cid, dept}$$

What is the closure of X when X ={cid, year}?

algorithm (F, X)
1. $C = X$
2. **while** F has a FD $A \rightarrow B$ such that $A \subseteq C$ **do**
3. $C = C \cup B$
4. remove $A \rightarrow B$ from F
5. **return** C

- Input: X ={cid, year}, F is given FDs.
- C ={cid, year}

Example: Let F be the set of following FDs:

$$\text{cid} \rightarrow \text{title}$$
$$\text{title} \rightarrow \text{dept}$$
$$\text{cid, year} \rightarrow \text{dept}$$
$$\text{cid, year} \rightarrow \text{cid, dept}$$

What is the closure of X when X ={cid, year}?

algorithm (F, X)
1. $C = X$
2. **while** F has a FD $A \rightarrow B$ such that $A \subseteq C$ **do**
3. $C = C \cup B$
4. remove $A \rightarrow B$ from F
5. **return** C

- Input: X ={cid, year}, F is given FDs.
- C ={cid, year}
- By "cid $\rightarrow$ title", {cid}$\subseteq C$, C ={cid, year, title}

Example: Let F be the set of following FDs:

$$\text{cid} \rightarrow \text{title}$$
$$\text{title} \rightarrow \text{dept}$$
$$\text{cid, year} \rightarrow \text{dept}$$
$$\text{cid, year} \rightarrow \text{cid, dept}$$

What is the closure of X when X ={cid, year}?

algorithm (F, X)
1. $C = X$
2. **while** F has a FD $A \rightarrow B$ such that $A \subseteq C$ **do**
3. $C = C \cup B$
4. remove $A \rightarrow B$ from F
5. **return** C

- Input: X ={cid, year}, F is given FDs.
- C ={cid, year}
- By "cid $\rightarrow$ title", {cid}$\subseteq C$, C ={cid, year, title}
- By "title $\rightarrow$ dept", C ={cid, year, title, dept}
- X^+ = {cid, year, title, dept}

Candidate Key Revisited

In creating a table, it may seem that so far we have been specifying candidate keys based on our preferences. This illusion is created because we did not understand FDs. In fact, candidate keys are not up to us at all. Instead, they are uniquely determined by the set F of functional dependencies from the underlying application. See the next slide.

Candidate Key Revisited

Let F be a set of FDs, and R a relation.

Definition

A candidate key is a set X of attributes in R such that

- X^+ includes all the attributes in R.
- There is no proper subset Y of X such that Y^+ includes all the attributes in R.

Note: A proper subset Y is a subset of X such that $Y \neq X$ (i.e., X has at least one element not in Y).

Example. Consider a table $R(A, B, C, D)$, and that $F = A \rightarrow B, B \rightarrow C$.
Prove that AD is a candidate key by definition.

1. $AD^+ = \{A, B, C, D\}$. That is, AD implies all attributes in R.
2. Now we enumerate all proper subsets of AD : $\{A\}$ and $\{D\}$.
3. And compute their closures.
   - $A^+ = \{A, B, C\}$.
   - $D^+ = \{D\}$.
4. No proper subset of AD can imply all attributes in R.
5. By definition, AD is a candidate key. (QED)

Example. Consider a table $R(A, B, C, D)$, and that
$F = A \rightarrow B, B \rightarrow C$.
Prove that AD is a candidate key by definition.

1. $AD^+ = \{A, B, C, D\}$. That is, AD implies all attributes in R.
2. Now we enumerate all proper subsets of AD : $\{A\}$ and $\{D\}$.
3. And compute their closures.
   - $A^+ = \{A, B, C\}$.
   - $D^+ = \{D\}$.
4. No proper subset of AD can imply all attributes in R.
5. By definition, AD is a candidate key. (QED)

Similarly, can you show that ABD is NOT a candidate key of R?

DB Design

- In designing a database, for the purpose of minimizing redundancy, we need to collect a set of functional dependencies (FD) that reflect the constraints of the underlying application.
- This can be either given through discussion with clients, or identify them yourself and verify with clients.
    - We have an FD only if it holds for every instance of the relation.
    - You can't know this just by looking at one instance.
    - You can only determine this based on domain knowledge.

Coincidence or FD?

ID	Email	City	Country	Surname
5123	tff@gmail.com	Toronto	Canada	Fairgrieve
9999	cuz@bell.com	London	Canada	Samways
8798	sms@gmail.com	Winnipeg	Canada	Samways
5645	birds@gmail.com	Aachen	Germany	Lakemeyer

Think

In this instance, are these FDs?

- Surname $\rightarrow$ Country
- City $\rightarrow$ Country

Coincidence or FD?

ID	Email	City	Country	Surname
5123	tff@gmail.com	Toronto	Canada	Fairgrieve
9999	cuz@bell.com	London	Canada	Samways
8798	sms@gmail.com	Winnipeg	Canada	Samways
5645	birds@gmail.com	Aachen	Germany	Lakemeyer

Think

In this instance, are these FDs?

- Surname $\rightarrow$ Country
- City $\rightarrow$ Country

Paris, France	VS	Paris, Texas
Aberdeen, Scotland	VS	Aberdeen, Washington
Venice, Italy	VS	Venice, Louisiana
Perth, Australia	VS	Perth, Scotland

Swansea, New South Wales 2281, Australia

Inferring FDs

- Given a set of FDs, we can often infer further FDs.
- This will come in handy when we apply FDs to the problem of database design.
- Big task: given a set of FDs, infer every other FD that must also hold.
- Simpler task: given a set of FDs, infer whether a given FD must also hold.

Inferring FDs

Examples of Simpler Tasks:

- If $A \to B$, $B \to C$ hold, must $A \to C$ hold?

- If $A \to H$, $C \to F$, $FG \to AD$,
  must $FA \to D$ hold?
  must $CG \to FH$ hold?

- If $H \to GD$, $HD \to CE$, $BD \to A$ hold,
  must $EH \to C$ hold?

- Aside: we are not generating new FDs, but testing a specific possible one.

Inferring FDs

Two methods:

1. Using first principles - Show it by referring back to
    - The FDs that you know hold
    - The Armstrong's Axioms
    - The definition of functional dependency.
2. Using the closure test
    - Assume you know the values of the LHS attributes, and figure out everything else that is determined.
    - If it includes the RHS attributes, then you know that LHS $\rightarrow$ RHS

Inferring FDs

If A $\rightarrow$ B, B $\rightarrow$ C hold,
must A $\rightarrow$ C hold?

By applying transitivity,
Yes. it holds.

If A $\rightarrow$ H, C $\rightarrow$ F, FG $\rightarrow$ AD,
must FA $\rightarrow$ D hold?

$FA^+ = \{F, A\}$
$FA^+ = \{F, A, H\}$ by A $\rightarrow$ H
No. FA $\rightarrow$ D does not hold.

Inferring FDs

If A → H, C → F, FG → AD, must CG → FH hold?

$CG^+ = \{C, G\}$
$CG^+ = \{C, G, F\}$, by C → F
$CG^+ = \{C, G, F, A, D\}$, by FG → AD
$CG^+ = \{C, G, F, A, D, H\}$ by A → H
yes, CG → FH hold.

If H → GD, HD → CE, BD → A hold, must EH → C hold?

$EH^+ = \{E, H\}$
$EH^+ = \{E, H, G, D\}$, by H → GD
$EH^+ = \{E, H, G, D, C\}$, by HD → CE
yes, EH → C hold.

- Ideally, we do not want to miss any FD, i.e., we want to obtain a set of FDs that is as large as possible. However, in practice, FD collection is a difficult process. No one can guarantee always discovering all FDs.

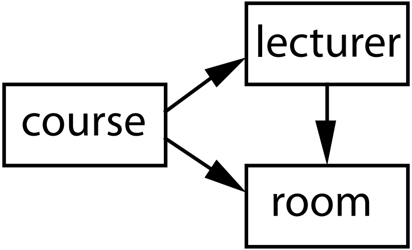
- What about if we cannot find all FDs?
  There is nothing we can do about them, unfortunately, and will have to continue the design without them. This is why even an experienced database professional may not always be able to come up with a perfect design!

Functional Dependence Diagram

Functional dependencies can be represented as diagram.

Example

course $\rightarrow$ lecturer, room
lecturer $\rightarrow$ room

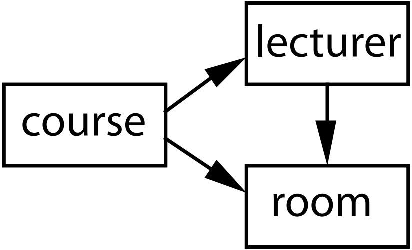


Functional Dependence Diagram

Functional dependencies can be represented as diagram.

Example

course $\rightarrow$ lecturer, room
lecturer $\rightarrow$ room

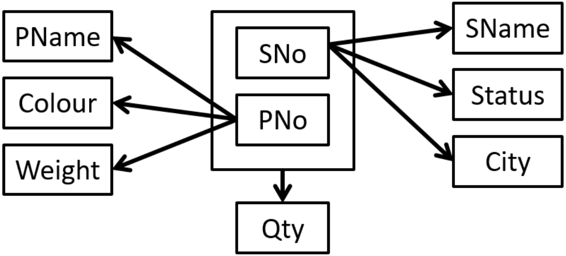


Remember: this is how you draw Functional Dependence Diagram.
FD diagram also helps you spot the candidate key too.

Functional Dependence Diagram

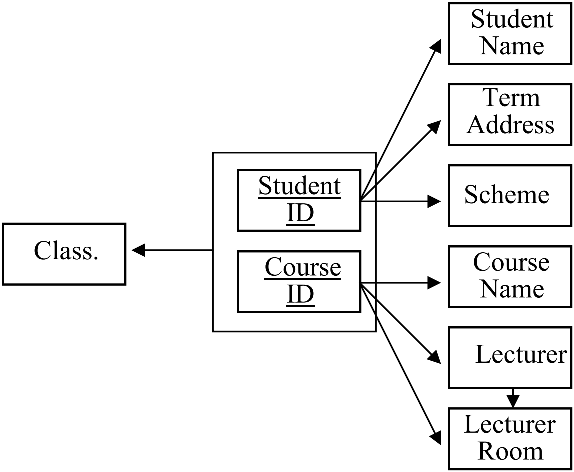
Given the following dependencies:

supplier:	sno	$\rightarrow$	sname, status, city
part:	pno	$\rightarrow$	pname, color, weight
supply:	sno, pno	$\rightarrow$	qty



- direction of arrow
- how composite key is depicted
- real world constraints: e.g. sno $\rightarrow$ city
  means each supplier can only locate at exactly one city.

Another example



Candidate Keys & Primary Key

- Primary key - one of the candidate keys chosen by DB designer.

How to identify potential candidate keys?

- Primary key - one of the candidate keys chosen by DB designer.

How to identify potential candidate keys?

- Closure Test

- Primary key - one of the candidate keys chosen by DB designer.

How to identify potential candidate keys?

- Closure Test

Example - Elements table

- Element Name, Symbol and Atomic Number
- e.g. Hydrogen, H, 1

- Each of these attributes can be a candidate key
- Which one to choose?

Candidate Keys & Primary Key

Some rules for choosing Primary Keys

1. One/few attributes is preferable to many attributes

Candidate Keys & Primary Key

Some rules for choosing Primary Keys

1. One/few attributes is preferable to many attributes
2. Multi-attributes and/or string keys
   - are redundant

     e.g. Movies: (title, year): 16bytes.

     e.g. movieID (int, 4bytes. Nbr movies ever made $\ll 2^{32}$)

Candidate Keys & Primary Key

Some rules for choosing Primary Keys

1. One/few attributes is preferable to many attributes
2. Multi-attributes and/or string keys
    - are redundant
      e.g. Movies: (title, year): 16bytes.
      e.g. movieID (int, 4bytes. Nbr movies ever made $\ll 2^{32}$)
    - are not secure
      e.g. Patient: (FirstName, LastName, Phone) vs PatientID

Candidate Keys & Primary Key

Some rules for choosing Primary Keys

1. One/few attributes is preferable to many attributes
2. Multi-attributes and/or string keys
   - are redundant
     e.g. Movies: ($\underline{\text{title, year}}$): 16bytes.
     e.g. movieID (int, 4bytes. Nbr movies ever made $\ll 2^{32}$)
   - are not secure
     e.g. Patient: ($\underline{\text{FirstName, LastName, Phone}}$) vs PatientID
   - are brittle
     e.g. name/phone change?
     e.g. two movies with the same title and year?

Candidate Keys & Primary Key

Some rules for choosing Primary Keys

1. One/few attributes is preferable to many attributes
2. Multi-attributes and/or string keys
   - are redundant
     e.g. Movies: (title, year): 16bytes.
     e.g. movieID (int, 4bytes. Nbr movies ever made $\ll 2^{32}$)
   - are not secure
     e.g. Patient: (FirstName, LastName, Phone) vs PatientID
   - are brittle
     e.g. name/phone change?
     e.g. two movies with the same title and year?
3. Also computers are really good at integers!