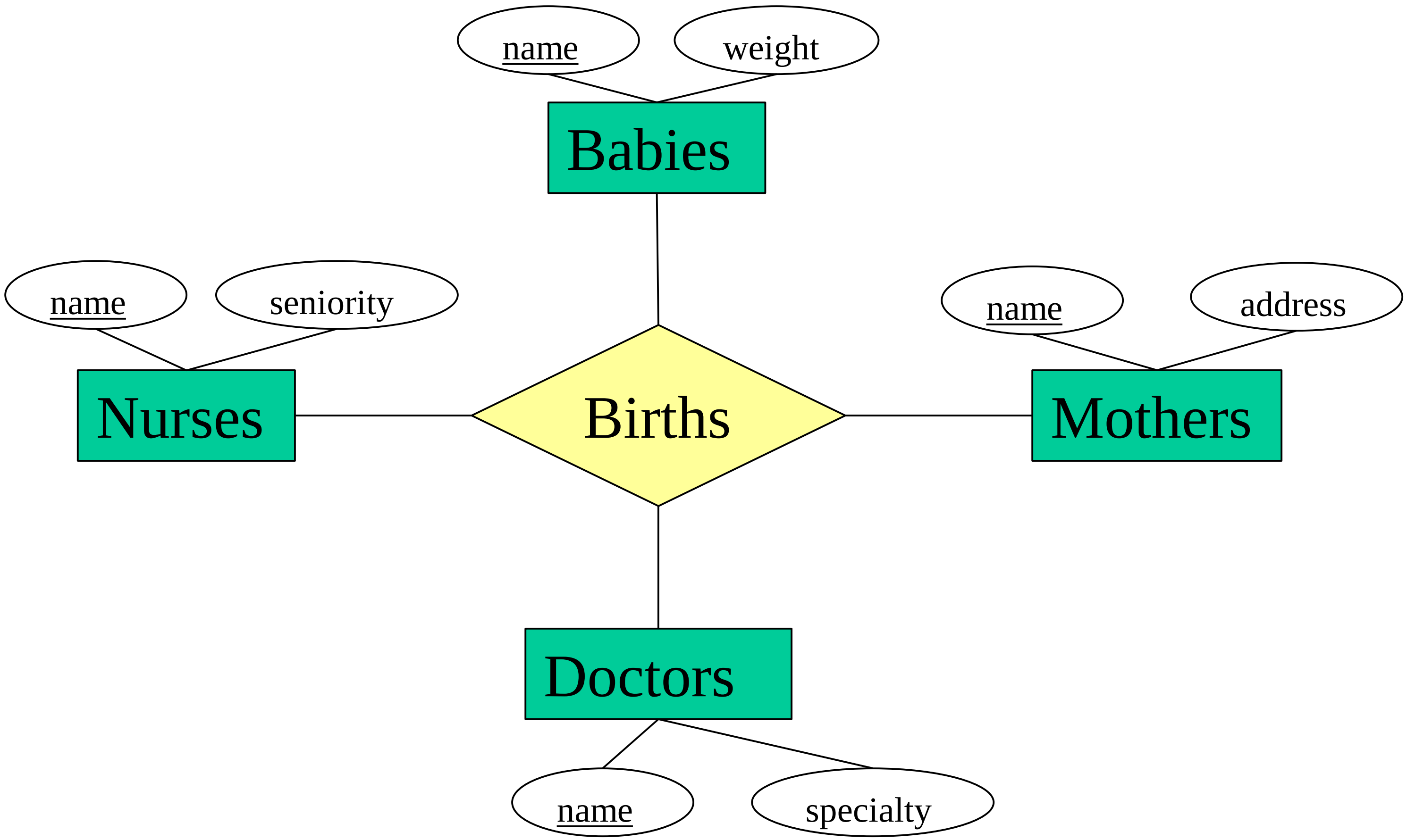
Functional Dependencies and Boyce-Codd Normal Form

Babies Schema

- At a birth, there is one baby (twins would be represented by two births), one mother, any number of nurses, and a doctor.



Babies Table

Births(baby, mother, nurse, doctor)

Some facts and assumptions

a) For every baby, there is a unique mother.

b) For every (existing) combination of a baby and a mother there is a unique doctor.

c) There are many nurses assisting in a birth.

Anomalies

Baby	Mother	Nurse	Doctor
Ben	Mary	Ann	Brown
Ben	Mary	Alice	Brown
Ben	Mary	Paula	Brown
Jason	Mary	Angela	Smith
Jason	Mary	Peggy	Smith
Jason	Mary	Rita	Smith

- **Redundancy**.
  - Information may be repeated unnecessarily in several tuples.

A Fix

Baby	Mother
Ben	Mary
Jason	Mary

Baby	Doctor
Ben	Brown
Jason	Smith

Baby	Nurse
Ben	Ann
Ben	Alice
Ben	Paula
Jason	Angela
Jason	Peggy
Jason	Rita

Redundancy in **Baby** column is a necessary redundancy.

Redundancy in **Mother** and **Doctor** columns removed.

Alternative Fix

Baby	Mother	Doctor
Ben	Mary	Brown
Jason	Mary	Smith

Baby	Nurse
Ben	Ann
Ben	Alice
Ben	Paula
Jason	Angela
Jason	Peggy
Jason	Rita

Functional Dependencies

- $X \rightarrow A$ for a relation R means that
  - whenever two tuples of R agree on all the attributes of X, then they must also agree on attribute A.
  - We say: "X functionally determines A"

Example

baby → mother

baby mother → doctor

Convention:

X, Y, Z represent sets of attributes;
A, B, C,… represent single attributes.
will write just ABC, rather than $\{A,B,C\}$.

Another Example

Drinkers(name, addr, beersLiked, manf, favBeer)

Reasonable FD’s to assert:

1. name → addr
2. name → favBeer
3. beersLiked → manf

Example Data

name	addr	beersLiked	manf	favBeer
Janeway	Voyager	Bud	A.B.	WickedAle
Janeway	Voyager	WickedAle	Pete's	WickedAle
Spock	Enterprise	Bud	A.B.	Bud

Because name -> addr

Because name -> favBeer

Because beersLiked -> manf

FD's With Multiple Attributes

- No need for FD's with more than one attribute on the **right**.
  - But convenient to combine FD's as a shorthand. E.g.

    name -> addr and

    name -> favBeer become

    name -> addr favBeer

- More than one attribute on the **left** may be **essential**. E.g.

    bar beer -> price

- An FD $\mathbf{A_1A_2 \ldots A_n \rightarrow B}$ is **trivial** if **B** is one of **A**'s. E.g.

    bar beer → beer

Keys of Relations

- K is a superkey for relation R if K functionally determines **all** of R's attributes.

- K is a key for R if
  - K is a superkey, and
  - Can't remove some attribute from K and still be superkey

- E.g. K=**{baby, nurse}** is a **key** for **Births**.

Boyce-Codd Normal Form

- Boyce-Codd Normal Form (BCNF): simple condition under which the anomalies can be guaranteed not to exist.

- Relation R is in **BCNF** if:

  Whenever there is a nontrivial dependency

  $A_1 \ldots A_n \rightarrow B_1 \ldots B_m$

  for R, it must be the case that

  {$A_1, \ldots, A_n$**}** is a **superkey** for R.

BCNF Violation - Example

Relation **Babies** isn’t in **BCNF**.

- FD: baby→mother
- Left side isn't a superkey.
  - We know: baby doesn't functionally determine nurse.

Decomposition into BCNF

- Goal of decomposition is to replace a relation by several that don't exhibit anomalies.

- **Decomposition strategy** is:
  - Find a non-trivial FD $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$ that **violates** BCNF, i.e. $A_1A_2\ldots A_n$ isn't a superkey.
  - Decompose relation schema into two schemas:
    - One is **all the attributes** involved in the **violating dependency** and
    - the other is the **left side** and **all the other attributes** not involved in the dependency.

- By **repeatedly**, choosing suitable decompositions, we can break any relation schema into a collection of smaller schemas in BCNF.

Babies Decomposition

Births(baby, mother, nurse, doctor)

baby→mother is a violating FD, so we decompose.

Baby	**Mother**
Ben	Mary
Jason	Mary

Baby	**Nurse**	**Doctor**
Ben	Ann	Brown
Ben	Alice	Brown
Ben	Paula	Brown
Jason	Angela	Smith
Jason	Peggy	Smith
Jason	Rita	Smith

This relation needs to be further decomposed using baby→doctor

We'll see a formal algorithm for deducing this FD.

Babies Decomposition

Baby	Mother
Ben	Mary
Jason	Mary

Baby	Doctor
Ben	Brown
Jason	Smith

Baby	Nurse
Ben	Ann
Ben	Alice
Ben	Paula
Jason	Angela
Jason	Peggy
Jason	Rita

Deducing FDs

- Suppose **we are told** of a set of FDs
  Based on them we can **deduce** other FDs.

Example.

baby→mother and

baby mother → doctor

imply

baby → doctor

But, what's the algorithm?

Closure of a Set of Attributes

- There is a **general principle** from which all possible **FD**'s follow.
- Suppose $\mathbf{\{A_1, A_2, \ldots, A_n\}}$ is a set of attributes and **S** is a set of **FD**'s.
- Closure of $\{A_1, A_2, \ldots, A_n\}$ under the dependencies in **S** is the **set** of attributes **B**, which are functionally determined by $A_1, A_2, \ldots, A_n$ i.e. $A_1A_2\ldots A_n \rightarrow \mathbf{B}$.
  - Closure is denoted by $\{A_1, A_2, \ldots, A_n\}^+$.
  - $A_1, A_2, \ldots, A_n$ are in $\{A_1, A_2, \ldots, A_n\}^+$

Computing the Closure - Algorithm

Brief

- Starting with the given set of attributes, repeatedly expand the set by adding the right sides of **FD**’s as soon as we have included their left sides.

- Eventually, we cannot expand the set any more, and the resulting set is the **closure**.

Computing the Closure - Algorithm

Detailed

1 Let **X** be a set of attributes that eventually will become the closure. First **initialize** **X** to be $\{A_1, A_2, \ldots, A_n\}$.

2 Now, **repeatedly** search for some **FD** in **S**:

$$B_1B_2\ldots B_m \rightarrow C$$

such that all of B's are in set **X**, but C isn't. Add C to **X**.

3 **Repeat step 2** as many times as necessary until no more attributes can be added to **X**.

Since **X** can only grow, and the number of attributes is finite, eventually nothing more can be added to **X**.

4 Set **X** after no more attributes can be added to it is: $\{A_1, A_2, \ldots, A_n\}^+$.

Computing the Closure - Example

- Consider a relation with schema R(A, B, C, D, E, F) and FD's:

  $AB \rightarrow C$,

  $BC \rightarrow AD$,

  $D \rightarrow E$,

  $CF \rightarrow B$.

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

- Iterations:

$X = \{A,B\}$	Use: $AB \rightarrow C$
$X = \{A,B,C\}$	Use: $BC \rightarrow AD$
$X = \{A,B,C,D\}$	Use: $D \rightarrow E$
$X = \{A,B,C,D,E\}$	No more changes to **X** are possible so $\mathbf{X = \{A,B\}^+}$.

- FD: $CF \rightarrow B$ wasn't used because its left side is never contained in **X**.

Why Computing the Closure?

- Having $\{A_1A_2\ldots A_n\}+$, we can test/generate any given functional dependency $A_1A_2\ldots A_n \rightarrow B$.

- If $B \in \{A_1, A_2, \ldots , A_n\}^+$ then FD: $A_1A_2\ldots A_n \rightarrow B$ **holds**.

- If $B \notin \{A_1, A_2, \ldots , A_n\}^+$ then FD: $A_1A_2\ldots A_n \rightarrow B$ **doesn't hold.**

Example

- Consider the previous example: $R(A, B, C, D, E, F)$ and FD's: $AB \to C$, $BC \to AD$, $D \to E$, $CF \to B$.

- Suppose we want to test whether FD: $AB \to D$ follows.

  Yes! Since $D \in \{A,B,C,D,E\} = \{A,B\}^+$.

- **On the other hand** consider testing FD: $D \to A$.
  - First compute $\{D\}^+$.
  - $\{D\}^+ = \{D,E\}$ and $A \notin \{D\}^+$.
  - We conclude that $D \to A$ **doesn't** follow from the given set of dependencies.

baby functionally determines **doctor**

We have

baby → mother

baby mother → doctor

Let's compute the **closure** of baby.

Iterations:

{baby}

{baby, mother}

{baby, mother, doctor}

{baby}+ = {baby, mother, doctor}

i.e. baby→doctor holds.

Closures and Keys

$\{A_1, A_2, \ldots, A_n\}$ is a **superkey**

iff

$\{A_1, A_2, \ldots, A_n\}^+$ is the set of **all** attributes.

A Few Tricks

- To deduce all the FDs, compute the closure of each subset of attributes, but
  - Never need to compute the closure of the empty set or of the set of all attributes.
  - If we find X^+ = all attributes, don't bother computing the closure of any supersets of X.

Movie Example

Movies(title, year, studioName, president, presAddr)

and FDs:

title year $\rightarrow$ studioName

studioName $\rightarrow$ president

president $\rightarrow$ presAddr

Last two violate **BCNF**. Why?

Compute {title, year}+, {studioName}+, {president}+ and see if you get all the attributes of the relation.

If not, you got a BCNF violation, and need to decompose.

Example (Continued)

Let's decompose starting with:

studioName → president

Optional rule of thumb:

Add to the right-hand side any other attributes in the closure of studioName.

{studioName}+ = {studioName, president, presAddr}

Thus, we get:

studioName→president presAddr

Example (Continued)

Using: studioName$\rightarrow$president presAddr we decompose into:

Movies1(studioName, president, presAddr)

Movies2(title, year, studioName)

Movie2 is in **BCNF**.

What about Movie1?

FD president$\rightarrow$presAddr violates **BCNF**.

Why is it bad to leave Movies1 as is?

*If many studios share the same president than we would have redundancy when repeating the **presAddr** for all those studios.*

Example (Continued)

We decompose Movies1, using FD: president→presAddr

The resulting relation schemas, both in **BCNF**, are:

Movies11(president, presAddr)

Movies12(studioName, president)

So, finally we got Movies11, Movies12, and Movies2.

In general, we must keep applying the decomposition rule as many times as needed, until all our relations are in **BCNF**.

FDs in the decomposed relations

Let **S** be one of the resulting relations in a decomposition of **R**.

Projecting FDs (*or computing FDs in a decomposed relation*)

Algorithm

Input: R, S, FDs in R

Output: FDs in S

- For each subset **X** of attributes of **S**.
  - Compute $\mathbf{X}^+$ using the FD on **R**.
  - Remove from $\mathbf{X}^+$ the attributes not in **S**.
  - For each A in $\mathbf{X}^+$ - X print $X \rightarrow A$ as FD in **S**.

Example

R(A,B,C,D,E) FDs: $C \rightarrow B$, $B \rightarrow D$, $A \rightarrow CD$

$A \rightarrow CD$ is a BCNF violating FD.

Decomposition:

S(A,C,D) and some other table.

In **S** we have $C \rightarrow D$, inferred by using B which is not in **S**.

$\{C\}^+ = \{C,B,D\}$ using the FDs in **R**;

Further decomposition of **S**:

S1(A,C), **S2**(C,D)

Recovering Info from a Decomposition

- Why a decomposition based on an FD preserves the information of the original relation?
- Because: *The **projections** of the original tuples can be **“joined”** again to produce **all** and **only** the original tuples.*

Example:

- Consider **R(A, B, C)** and FD **B→C**, which suppose is a **BCNF violation**.
- Let’s decompose based on **B→C**: **R_1(A,B)** and **R_2(B,C)**.
- Let **(a,b,c)** be a tuple of **R**, it projects as **(a,b)** for **R_1**, and as **(b,c)** for **R_2**.
- It's possible to **join** a tuple from **R_1** with a tuple from **R_2**, when they agree on the **B** component.
  - In particular, **(a,b)** joins with **(b,c)** to give us the original tuple **(a,b,c)** back again.
- Getting back those tuples we started with **isn't enough**.
- Do we also get false tuples, i.e. that weren’t in the original relation?

Example continued

- What might happen if there were two tuples of **R**, say **(a,b,c)** and **(d,b,e)**?
- We get:
  - **(a,b)** and **(d,b)** in $\mathbf{R_1}$
  - **(b,c)** and **(b,e)** in $\mathbf{R_2}$
- Now if we join $\mathbf{R_1}$ with $\mathbf{R_2}$ we get:
  - (a,b,c)
  - (d,b,e)
  - (a,b,e)   (is it bogus?)
  - (d,b,c)   (is it bogus?)
- They aren't bogus. By the FD **B→C** we know that if two tuples agree on **B**, they must agree on **C** as well. Hence **c=e** and we have:
  - (a,b,c)
  - (d,b,e)
  - (a,b,e) = (a,b,c)
  - (d,b,c) = (d,b,e)

What if **B→C** isn't a true FD?

- Suppose **R** consists of two tuples:

A	B	C
1	2	3
4	2	5

- The projections of **R** onto the relations with schemas **R_1(A,B)** and **R_2(B,C)** are:

A	B
1	2
4	2

and

B	C
2	3
2	5

- When we try to reconstruct R by joining, we get:

A	B	C
1	2	3
1	2	5
4	2	3
4	2	5

That is, we get **"too much."**

Problems

For

$R(A,B,C,D)$ with $AB \rightarrow C$, $C \rightarrow D$, and $D \rightarrow A$, and

$R(A,B,C,D)$ with $B \rightarrow C$, and $B \rightarrow D$

- Indicate all BCNF violations.
- Decompose into relations that are in BCNF.