Design Theory for Relational Databases

Thomas Schwarz, SJ

Contents

- There are many ways a database scheme can be constructed
 - A poorly designed scheme:
 - Has problems with checking constraints
 - Has problems with data coherence
 - E.g. two different spellings of the same person's first name
 - Has problems with performance

Contents

- Design theory helps to design efficient schemes
 - Functional dependencies
 - Used in the definition of a key
 - Used for flagging potentially bad records
 - Normal forms
 - Get rid of anomalies
 - Get rid of redundant storage of data
 - Expand to multivalued dependencies

- A form of constraint for a relation
 - Functional Dependency (FD) for table R(X)
 - $FDA_1, A_2, ...A_n \longrightarrow B_1, B_2, ..., B_m$
 - with $A_1, ..., A_n, B_1, ..., B_m \in X$
 - If a tuple's values agree for attributes $A_1, ..., A_n$
 - Then they agree for attributes $B_1, B_2, ..., B_m$

- Only consider FD with one attribute on the right
 - Because FD $A_1, A_2, ...A_n \longrightarrow B_1, B_2, ..., B_m$ is equivalent to all of:

•
$$A_1, A_2, ...A_n \longrightarrow B_1$$

•
$$A_1, A_2, ...A_n \longrightarrow B_2$$

•
$$A_1, A_2, ...A_n \longrightarrow B_m$$

Example:

- Movies1(title, year, length, genre, studioName, starName)
- Find all FDs

- title, year -> length
- title, year -> genre
- title, year -> studio
- However:
 - title, year → starName
 - is not an FD

Keys

- A superkey is a set of attributes in a table that determines all attributes
 - $R(A_1, A_2, ...A_n)$
 - $A_{i_1}, A_{i_2}, \dots A_{i_m}$ is a superkey if
 - $\forall j$: $A_{i_1}, A_{i_2}, \dots A_{i_m} \longrightarrow A_j$

Keys

- A key is a minimal superkey with respect to set inclusion
 - I.e. A superkey so that no attribute in it can be reomved

 If a key consists of a single attribute, then we call the attribute the key instead of the set with only element this attribute

- Quiz: Given R(A, B, C) and FDs $A \rightarrow B$ and $B \rightarrow C$,
- Does this mean $A \rightarrow C$?

- Answer: Yes.
- Show that all tuples that agree on attribute A also agree on attribute C

- A set S of FDs follows from a set T of FDs if every relation instance satisfying all FDs in T also satisfies all FDs in S
- Sets of FDs are equivalent if the set of relation instances satisfying one is equal to the set of relation instances satisfying the other one.

- The splitting rule:
 - $A_1, A_2, ...A_n \longrightarrow B_1, B_2, ..., B_m$ is equivalent to
 - $A_1, A_2, ...A_n \longrightarrow B_1$
 - $A_1, A_2, ...A_n \longrightarrow B_2$
 - •
 - $A_1, A_2, ...A_n \longrightarrow B_m$

- The combining rule:
 - The set of FDs

•
$$A_1, A_2, ...A_n \longrightarrow B_1$$

•
$$A_1, A_2, ...A_n \longrightarrow B_2$$

•
$$A_1, A_2, ...A_n \longrightarrow B_m$$

• is equivalent to $A_1, A_2, ...A_n \longrightarrow B_1, B_2, ..., B_m$

- Trivial FDs
 - $A_i \rightarrow A_i$
 - $A, B \rightarrow A$
- Trivial Dependency Rule:
 - $A_1A_2...A_n \rightarrow B_1B_2...B_m$ is equivalent to
 - $A_1A_2...A_n \rightarrow C_1C_2...C_r$
 - where the C_i are those of the B_i that are not among the ${\cal A}_i$

- Closure:
 - Let S be a set of functional dependencies
 - Let $\mathbb{A} = \{A_1, A_2, ..., A_n\}$ be a set of attributes
 - The <u>closure</u> of \mathbb{A} is the set \mathbb{A}^+ of attributes B such that every relation that satisfies all the FDs in \mathbb{S} also satisfies $A_1, A_2, \ldots, A_n \to B$.

- Closure calculation algorithm
 - Input: a set of attributes \mathbb{A} and a set of functional dependencies \mathbb{S} .
 - Output: A⁺

- 1. Split all FDs in $\mathbb S$ so that there is only a single attribute on the right
- 2.Set X to be A.
- 3.Repeatedly search for some FD $B_1, B_2, ..., B_m \to C \in \mathbb{S}$ such that $B_1, B_2, ..., B_m \in \mathbb{A}$ and $C \notin \mathbb{A}$. Then add C to \mathbb{X}
- 4. Stop when the search fails and output $X = A^+$.

- Consider the relation scheme R = {E, F, G, H, I, J, K, L, M, M} and the set of functional dependencies {{E, F} -> {G}, {F} -> {I, J}, {E, H} -> {K, L}, K -> {M}, L -> {N} on R. What is the key for R?:
 - {E,F}
 - {E,F,H}
 - {E,F,H,K,L}
 - {E}
 - Hint: calculate the closure of all possible answers

- First, normalize {{E, F} -> {G}, {F} -> {I, J}, {E, H} -> {K, L},
 K -> {M}, L -> {N}
 - $\{ \{E, F\} \to G, \{F\} \to I, \{F\} \to J, \{E, H\} \to K, \{E, H\} \to L, K \to M, L \to N \}$
 - Start with $\{E\}$.
 - There is no FD that has only E on the left side

- $\{ \{E, F\} \to G, \{F\} \to I, \{F\} \to J, \{E, H\} \to K, \{E, H\} \to L, K \to M, L \to N \}$
 - Now try $\{E, F\} = X$
 - We can add G to X.
 - We can add I to X.
 - We can add J to X.
 - Then we are stuck: $\{E, F\}^+ = \{E, F, G, I, J\}$

- $\{\{E,F\}\to G, \{F\}\to I, \{F\}\to J, \{E,H\}\to K, \{E,H\}\to L, K\to M, L\to N\}$
 - Now try $\{E, F, H\} = X$
 - We can add G to X because of (1).
 - We can add I to \mathbb{X} because of (2): $\mathbb{X} = \{E, F, G, H, I\}$
 - We can add J to \mathbb{X} because of (3): $\mathbb{X} = \{E, F, G, H, I, J\}$
 - (4) gives $X = \{E, F, G, H, I, J, K\}$
 - (5) gives $X = \{E, F, G, H, I, J, K, L\}$
 - (6) gives $X = \{E, F, G, H, I, J, K, L, M\}$
 - (7) gives $X = \{E, F, G, H, I, J, K, L, M, N\}$
- Therefore $\{E, F, H\}^+$ contains all the attributes.
 - Since $\{F,H\}^+ = \{F,H,I,J\}$, $\{E,F,H\}$ is a minimal candidate key and therefore a key.

- Why does closure work
 - Need to show equivalency of:
 - $B \in \{A_1, A_2, ..., A_n\}^+$ with regards to \mathbb{S}
 - Every relation fulfilling \mathbb{S} fulfills $A_1A_2...A_n \to B$

- Why does
 - $B \in \{A_1, A_2, ..., A_n\}^+$ with regards to \mathbb{S}
 - imply
 - Every relation fulfilling \mathbb{S} fulfills $A_1A_2...A_n \to B$
- Look at the first time adding an attribute to \mathbb{X} leads to an FD $A_1A_2...A_n \to B$ that is **not** true.
 - But B was added using a FD $X_1, X_2, ... X_m \rightarrow B$
 - Because this is the first time and $X_1,X_2,...X_m$ follow from the $A_1A_2...A_n$ in all relations, $A_1A_2...A_n \to X_1,...X_m$
 - Thus, a tuple equal in $A_1A_2...A_n$ is also equal in all $X_1,...X_m$ and hence equal in B.
 - Therefore $A_1A_2...A_n \to B$ has to be true and we have a contradiction

- Why does
 - Every relation fulfilling \mathbb{S} fulfills $A_1A_2...A_n \to B$
 - imply
 - $B \in \{A_1, A_2, ..., A_n\}^+$ with regards to \mathbb{S}
- Assume $B \notin \{A_1, A_2, ..., A_n\}^+$ with regards to \mathbb{S} , but $A_1A_2...A_n \to B$ holds in all relations that also fulfill \mathbb{S} .
 - Create a simple table:

```
{A1 A2 ... An}+ every thing else
0 0 0 0 0 ... 0
0 0 1 1 ... 1
```

- Does this instance satisfy \$?
 - Assume an FD $C_1C_2...C_r \to D$ in $\mathbb S$ is violated
 - For a violation to occur, the C_i need to be on the left side, i.e. in $\{A_1, A_2, ..., A_n\}^+$ and the D on the right side of the table.

```
{A1 A2 ... An}+ every thing else
0 0 0 0 0 ... 0
0 0 1 1 ... 1
```

• But then we did not calculate the closure correctly and D should have been in $\{A_1, A_2, \ldots, A_n\}^+$

• Does this instance not satisfy $A_1A_2...A_n \rightarrow B$

```
{A1 A2 ... An}+ every thing else 0 0 0 0 0 ... 0 0 0 0 1 1 1 ... 1
```

Yes!

Therefore the assumption is violated and this finishes the proof

- With the closure calculation, we can prove
 - If in a relation R $A_1,A_2,\ldots,A_m\to B_1,B_2,\ldots,B_n$ and $B_1,B_2,\ldots,B_n\to C_1,C_2,\ldots C_t$ then $A_1,A_2,\ldots,A_m\to C_1,C_2,\ldots C_t$
 - Transitivity

- We sometimes have a choice in the minimal set of FDs that describe a relation
 - A set of FD is called a <u>basis</u> if all FDs holding in the relation can be derived from the basis
 - A <u>minimal basis</u> B:

 - If in any FD from $\mathbb B$ we drop an attribute from the right side, then the result is no longer a basis

- Example:
 - A relation with three attributes such that each attribute determines the other attributes

What are the FDs?

Find a minimal basis

- Answer: FDs are
 - $A \to B, A \to C$ and all augmentations $A \to B, C$ including the trivial ones $A \to A, B, A \to A, C$ and $A \to A, B, C$
 - $B \to A$, $B \to C$ plus all augmentation
 - $C \rightarrow A$, $C \rightarrow B$ plus all augmentations

 Answer: To obtain a bases, we can look at <u>all</u> subsets of right side singleton

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

- For example:
 - We try to remove from left
 - $A \to B$ follows from $A \to C$ & $C \to B$
 - Left with $\{A \rightarrow C, B \rightarrow A, B \rightarrow C, C \rightarrow A, C \rightarrow B\}$

- Left with $\{A \rightarrow C, B \rightarrow A, B \rightarrow C, C \rightarrow A, C \rightarrow B\}$
 - Now can get rid of $B \to A$
- Left with $\{A \rightarrow C, B \rightarrow C, C \rightarrow A, C \rightarrow B\}$

- Another possibility:
 - $\{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$

- Projecting Functional Dependencies
 - Given a relation R with a set of FDs S and a subset L
 of attributes of R:
 - What are the FDs induced in $\pi_L(R)$?

- ullet FDs can only involve attributes from L
- But restricting S to those is not enough

- Algorithm:
 - Start out with an empty set T of FDs
 - For each set M of attributes $\subset L$ calculate the closure M^+ in R
 - If $M \to X$ is a FD calculated this way and $X \in L$, add the FD to $\mathbb T$
 - Modify T to become a minimal basis
 - Remove all FDs that follow from others in $\mathbb T$

Functional Dependencies

- Example: R(A,B,C,D) with $\mathbb{S}=\{A\to B,B\to C,C\to D\} \text{ projected on } L=\{A,C,D\}$
- Calculate first closures
 - $\{A\}^+ = \{A, B, C, D\}$
 - $\{B\}^+ = \{B, C, D\}$
 - $\{C\}^+ = \{C, D\}$
 - $\{D\}^+ = \{D\}$

Functional Dependencies

- We really do not need any more because those with two attributes on the left would follow trivially
 - ullet Now we add the FDs derived from the closure, if all attributes are in L
 - $\mathbb{T} = \{A \rightarrow C, A \rightarrow D, C \rightarrow D\}$
 - This is not a base, because $A \rightarrow D$ follows from the other ones.
- The induced FDs have base $\mathbb{T} = \{A \to C, C \to D\}$

Anomalies

Take

```
movies = (title, year, length, genre, studioName, starName)
```

- Redundancy: The studioName for Star Wars is repeated for every star
 - This implies:
- Update anomaly: If we update the length of the movie, we need to repeat this update operation for every star or we get incoherent information
- **Delete anomaly**: If we delete all stars from an animation cartoon, we have no information left on the movie!

Divide the information over two tables

```
movies = (title, year, length, genre, studioName, starName)
```

becomes

```
movies1=(title, year, length, genre, studioName)
movies2=(title, year, studioName)
```

- Relation in BCNF if and only if:
 - Whenever there is a non-trivial FD $A_1...A_n \to B$ then $A_1...A_n$ is a superkey

- Example
 - movies1(title, year, length, genre, studio, star)
 - Has FD title, year --> studio
 - but because of the star attribute,
 title, year
 - We can decompose:
 - Take the left side of the FD
 - Calculate its closure
 - {title, year}+ = {title, year, length, genre, studio}
 - Decompose into closure and right side
 - movies(<u>title</u>, <u>year</u>, length, genre, studio)
 starsIn(title, year, star)

- What is good about BCNF?
 - Update anomaly
 - Decomposition prevents having to enter the same information multiple times
 - Delete anomaly
 - Can now have movies without stars
 - Can we do better?
 - Yes, sometimes. starsIn has still a two-attribute key

- Any two attribute table R(A,B) is in BCNF
 - Proof by case distinction:
 - Case 1: $A \nrightarrow B$, $B \nrightarrow A$
 - No nontrivial FDs exists, R is in BCNF
 - Case 2: $A \rightarrow B, B \nrightarrow A$
 - A is the only key and it is on the right of the only non-trivial FD. So BNCF.
 - Case 3: $A \nrightarrow B, B \rightarrow B$
 - Same as before
 - Case 4: $A \rightarrow B$, $B \rightarrow A$
 - Both A, B are keys. So, BCNF

- Decomposition:
 - Does decomposition loose information or add spurious information?
 - Does decomposition preserve dependencies
 - How do we do decomposition

- Finding decompositions
 - Look for a non-trivial FD.
 - If the right side is not a superkey:
 - Expand the right side as much as possible
 - $A_1A_2...A_n \rightarrow B_1...B_m$
 - Right side are <u>all</u> attributes that are dependent on $A_1...A_n$

Example:

prod(title, year, studio, president, presAddr)

```
• with FD title year -->studio
studio --> president
president --> presAddr
```

Question: What are possible keys?

- Only key is title, year
 - Just look at the closures of all subsets of attributes

Which FDs violate BCNF?

- Two FDs:
 - studio --> president
 - president --> presAddr

• What happens with studio --> president

- We calculate the closure of the right side
 - studio --> president
 - {studio}+ = {president, presAddr}
 - This gives a decomposition
 - (title, year, studio) (studio, president, presAddr)
 - Using projection of FDs, we get
 - title, year -->studio
 - studio --> president, president --> presAddr
 - so second relation is not in BCNF (studio is the only key)

- Now we decompose the second relation again:
 - (<u>studio</u>, president)
 (president, presAddr)

- Decomposition algorithm
 - If there is an FD $X \to Y$ that violates BCNF
 - Calculate X⁺
 - Choose X^+ as one relation and $X \cup \mathbb{C}(X^+)$ as the other
 - All attributes in X and all attributes not in X^+
 - Calculate the projected FDs
 - Continue

- In class exercise.
 - Find all BNCF violations (including those following from the FDs given)
 - Decompose the relation, if possible

• R(A,B,C,D); $AB \rightarrow C$; $C \rightarrow D$; $D \rightarrow A$

- In class exercise.
 - Find all BNCF violations (including those following from the FDs given)
 - Decompose the relation, if possible

 $R(A,B,C,D); AB \rightarrow C; BC \rightarrow D; CD \rightarrow A; AD \rightarrow B$

- Recovering data from decomposition
 - Assume a relation R(A, B, C) with FD $B \rightarrow C$, where B is not a key
 - Decomposition is then $R_1(A,B)$ and $R_2(B,C)$
 - Assume t = (a, b, c) is a tuple. It is projected as $t_1 = (a, b)$ and $t_2 = (b, c)$
 - Thus, $t \in R_1 \bowtie R_2$.
 - Assume $t_1=(a,b)\in R_1$ and $t_2=(b,c)\in R_2$, i.e. $t\in R_1\bowtie R_2$
 - There is a tuple $(a, b, x) \in R$ because R_1 is a projection.
 - (Similarly, there is a tuple $(a, y, c) \in R$.)
 - Because of the FD $B \to C$ there is only one value for x
 - Hence, the tuple must have been (a, b, x = c)

- This argument generalizes to sets A, B
 - This means: Boyce Codd decomposition is recoverable
 - Since natural joins are associative and commutative, the BCNF decomposition algorithm cannot loose information

- Dependency preservation
 - Assume a table

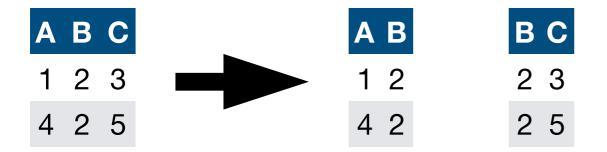
```
bookings(title, theater, city)
```

- FDs theater --> city title, city --> theater
- Keys are: title, city and title, theater

- The existence of the FDs is important
 - Assume a similar decomposition of R(A,B,C) but without the FDs $B \to A, \quad B \to C$
 - Example instance:

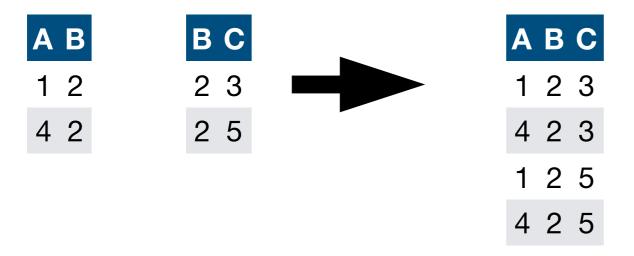
• Split into $R_1(A,B)$ and $R_2(B,C)$

Result of projection



What is the join of the two tables on the right?

Result



- which introduces spurious records.
- Of course, attribute B was not a key for the second relation!

Dependency Preservation

- Decompose into BCNF
 - (theater, city) (theater, title)
 - Must be BCNF, because it only has two attributes
 - However, FD title, city --> theater cannot be derived

• Example:

Theater	City
AMC	Wauwatosa
Marcus 1	Milwaukee
Marcus 2	Wauwatosa

```
Theater Title

Marcus 2 Doolittle

AMC Doolittle
```

- Violates the FD
 - title, city --> theater

- We just saw: R(A,B,C) with FD $B\to C$ has a lossless join into R(A,B) and R(B,C)
- Without FD $B \to C$ or $B \to A$, the join is not loss-less

- Question: Given a set of FDs in R and a set of sets of attributes $S_1, S_2, ... S_n$:
 - Is decomposition by projection onto the S_i lossless?
 - i.e.: is $\pi_{S_1}(R)\bowtie\pi_{S_2}(R)\bowtie\ldots\bowtie\pi_{S_n}(R)=R$?

- Two easy remarks:
 - Natural join is associative and commutative. The order in which we project is not important.
 - Certainly $R \subset \pi_{S_1}(R) \bowtie \pi_{S_2}(R) \bowtie \ldots \bowtie \pi_{S_n}(R)$

- Chase Test:
 - Task: Show that given the FDs, we can prove that

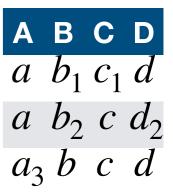
•
$$\pi_{S_1}(R) \bowtie \pi_{S_2}(R) \bowtie \ldots \bowtie \pi_{S_n}(R) \subset R$$

- Take a tuple $t \in R$
 - Use a <u>tableau</u> to determine the various versions this tuple could appear in the projections

- Tableau has one row for each decomposition
 - Put down unsubscripted letters for the attributes in the decomposed relationship
 - Put down subscripted letters for the attributes not in the decomposed relationship
 - Subscript is the number of the decomposed relationship

- Example: R(A,B,C,D) with projections on $S_1=\{A,D\}, S_2=\{A,C\}$ and $S_3=\{B,C,D\}$
 - A generic tuple in $S_1 \bowtie S_2 \bowtie S_3$ is then represented in the decomposition tableau

- The first row looks at the projection on A and D
 - From the projection, we know that a given tuple has certain a and d values, but the join might give some values for the b and c column

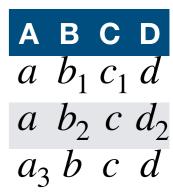


- Once given a tableau, we use the FDs in order to "chase down" identities between the elements in the tableau.
- We represent them by making subscripts equal or dropping them

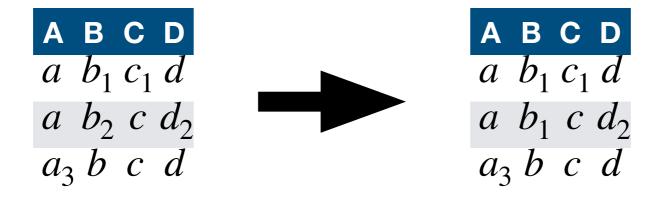
- Example:
 - Assume the following FDs for the example:

•
$$A \rightarrow B, B \rightarrow C, CD \rightarrow A$$

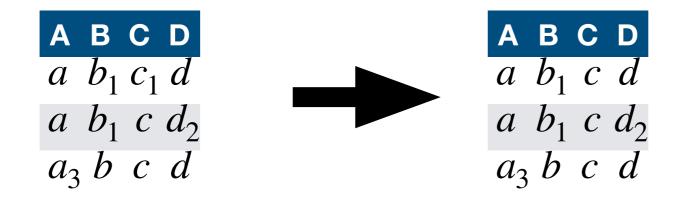
 Whenever we have tableau entries for attributes on the right side, we can use it to equalize the entries for attributes on the right of an FD



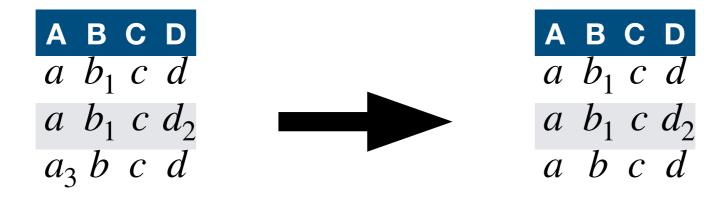
- Use $A \rightarrow B$:
 - First two rows, we have unsubscripted a.
 - Equalize the B column in these rows



• Use $FD B \rightarrow C$



• Now use $CD \rightarrow A$



- Now we have one row that is equal to t
 - This means: any tuple of the join has to be equal to the original tuple

- What happens if after applying all FDs, we still are left with unsubscripted variables?
 - Then this gives us a value in the join that is not in the original relation

- Example:
 - R(A,B,C,D) with FDs $B \to AD$ and decomposition into $\{A,B\},\{B,C\},\{C,D\}$

- Example:
 - R(A,B,C,D) with FDs $B \to AD$ and decomposition into $\{A,B\},\{B,C\},\{C,D\}$
 - Initial tableau is

```
A B C D
a \ b \ c_1 \ d_1
a_2 \ b \ c \ d_2
a_3 \ b_3 \ c \ d
```

- Example:
 - R(A,B,C,D) with FDs $B \to AD$ and decomposition into $\{A,B\},\{B,C\},\{C,D\}$
 - Initial tableau is

A B C D
$$a \ b \ c_1 \ d_1$$
 $a_2 \ b \ c \ d_2$
 $a_3 \ b_3 \ c \ d$

After applying the FD, we get tableau

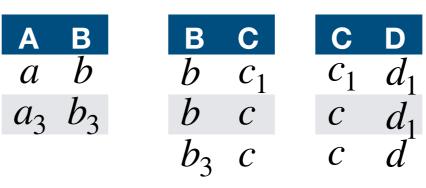
A B C D
$$a \ b \ c_1 \ d_1$$
 $a \ b \ c \ d_1$
 $a_3 \ b_3 \ c \ d$

Take this tableau and use it to construct a counter example

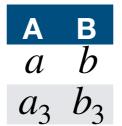
$$\begin{array}{c} \mathsf{A} \ \mathsf{B} \ \mathsf{C} \ \mathsf{D} \\ a \ b \ c_1 \ d_1 \\ a \ b \ c \ d_1 \\ \hline a_3 \ b_3 \ c \ d \end{array}$$

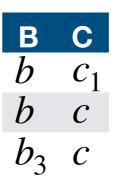
- Create tuples (a, b, c_1, d_1) , (a, b, c, d_1) , (a_3, b_3, c, d) in R.
- Fulfills the FD $B \to CD$
- Projections are

$$\begin{array}{ccc} A & B \\ a & b \\ a_3 & b_3 \end{array}$$



Join these together:





$$egin{array}{ccc} \mathbf{C} & \mathbf{D} \\ c_1 & d_1 \\ c & d_1 \\ c & d \end{array}$$

Result has two additional rows

Α	В	C	D
a	b	c_1	d_1
a	b	\mathcal{C}	d_1
\boldsymbol{a}	b	C	d
a_3	b_3	C	d_1
a_3	b_3	\mathcal{C}	d

The decomposition is not loss-less!

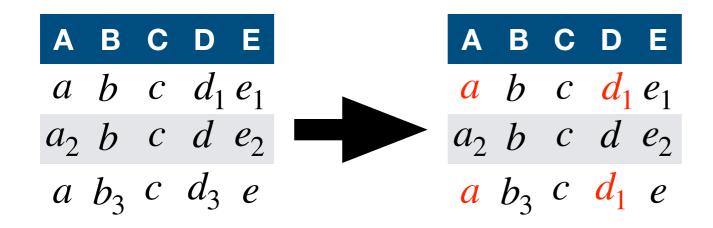
• Let R(A, B, C, D, E) be decomposed into $\{A, B, C\}$, $\{B, C, D\}$, $\{A, C, E\}$. Assume FDs $A \to D$, $CD \to E$, $E \to D$. Is the decomposition lossless?

• Let R(A,B,C,D,E) be decomposed into $\{A,B,C\}$, $\{B,C,D\}$, $\{A,C,E\}$. Assume FDs $A\to D$, $CD\to E$, $E\to D$. Is the decomposition lossless?

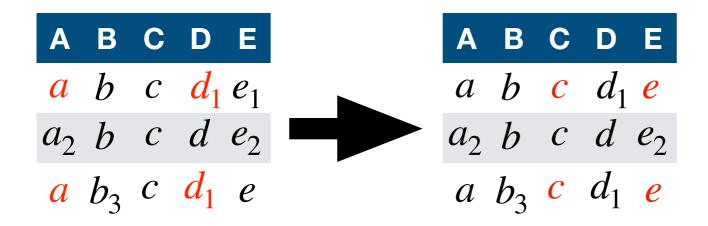
A B C D E

$$a \ b \ c \ d_1 e_1$$
 $a_2 \ b \ c \ d \ e_2$
 $a \ b_3 \ c \ d_3 \ e$

- Let R(A,B,C,D,E) be decomposed into $\{A,B,C\}$, $\{B,C,D\}$, $\{A,C,E\}$. Assume FDs $A\to D$, $CD\to E$, $E\to D$. Is the decomposition lossless?
 - Use $FDA \rightarrow D$



- Let R(A,B,C,D,E) be decomposed into $\{A,B,C\}$, $\{B,C,D\}$, $\{A,C,E\}$. Assume FDs $A\to D$, $CD\to E$, $E\to D$. Is the decomposition lossless?
 - Use FD $CD \rightarrow E$



- Let R(A,B,C,D,E) be decomposed into $\{A,B,C\}$, $\{B,C,D\}$, $\{A,C,E\}$. Assume FDs $A\to D$, $CD\to E$, $E\to D$. Is the decomposition lossless?
 - Cannot use FD $E \rightarrow D$, $A \rightarrow D$, $CD \rightarrow E$

- The tableau gives us tuples that satisfy the FDs
- Make the tableau into tuples
- Look at the projections

A	В	С	D	Е
a	b	С	d_1	e
a_2	b	C	d	e_2
a	b_3	\mathcal{C}	d_1	e

$$\begin{array}{cccc} \mathbf{B} & \mathbf{C} & \mathbf{D} \\ b & c & d_1 \\ b & c & d \\ b_3 & c & d_1 \end{array}$$

Join them

A	В	C	D	Ε
a	b	С	d_1	e
a_2	b	C	d	e_2
a	b_3	C	d_1	e

A B C
$$a b c$$
 $a_2 b c$ c $a b_3 c$

A	В	C	D	Е
a	b	$\boldsymbol{\mathcal{C}}$	d_1	e
a	b	\mathcal{C}		e
a_2	b	$\boldsymbol{\mathcal{C}}$	d_1	e_2
$\bar{a_2}$	b	\mathcal{C}	d	e_2
\tilde{a}	b_3	$\boldsymbol{\mathcal{C}}$	d_1	e^{-}
a	b_3	C		e_2

- Checking on FD is important
 - Database coherence
 - To detect faulty operation
 - E.g. booking the same movie at two theaters in a town
- Therefore: Relax conditions on BCNF
 - Third Normal Form
 - Allows checking of FDs
 - Loss-less join property

- A relation R is in third normal form
 - If $A_1A_2...A_n \rightarrow B_1B_2...B_m$ is a non-trivial FD, then
 - either $\{A_1, A_2, ..., A_n\}$ is a superkey
 - or those of $\{B_1,B_2,\ldots,B_n\}$ not in $\{A_1,A_2,\ldots,A_n\}$ are each member of *some* key (not necessarily the same)

Attributes that are part of some key are called prime

Example:

bookings(title, theater, city)

```
theater --> city
title, city --> theater
```

- is in third normal form
 - city is part of a key

- Creation of 3NF Schemas
 - Want to decompose a relation R into a set of relations such that
 - All relations in the set are in 3NF
 - The decomposition has a lossless join
 - The decomposition preserves dependencies

- Synthesis Algorithm
 - Given a relation R and a set \mathbb{F} of FDs
 - Find a minimal base G for F
 - For all FD $X \to A \in \mathbb{G}$: use XA as a schema
 - If none of the relation schemas from previous step are a superkey for R, add another relation whose schema is a key for R

- Example:
 - R(A, B, C, D, E) with FDs $AB \rightarrow C$, $C \rightarrow B$, $A \rightarrow D$

- Example:
 - The FDs are their own base:
 - Show: None of $AB \to C$, $C \to B$, $A \to D$ follows from the other two
 - Show: Cannot drop an attribute from a right side

- Example:
 - This gives relations
 - $S_1(A, B, C), S_2(B, C), S_3(A, D)$
 - Keys of R are A, B, E and A, C, E
 - Need to add one of them
 - $S_1(A, B, C)$, $S_2(B, C)$, $S_3(A, D)$, $S_4(A, C, E)$

- Why does this work
 - Lossless join:
 - We use the "Chase"
 - There is one subset of attributes in the decomposition that is a superkey \mathbb{K} .
 - The closure of K is all the attributes.
 - We start with a tableau

- Lossless join -- Chase
 - Use the FDs used in calculating the closure of \mathbb{K} .
 - We can assume that the FDs are in the base
 - Let the first FD be $\mathbb{X} \supset \mathbb{A} \to B$.
 - Tableau:

• The application of the FD sets b1 to b

- Lossless join -- Chase
 - We continue the process.
 - Next FD might use column B or not, but because of it, we loose the subscript in the column corresponding to the right side
 - Eventually, we have removed all subscripts in the first row
 - Therefore, the decomposition is loss-less

- Dependency Preservation
 - Any FD is the consequence of the FDs in the base
 - Any FD in the base is represented by a relation in the decomposition
 - Therefore, we can first check those and as a consequence get all the FDs

- Is the decomposition in third normal form
 - If we add a relation that corresponds to a key, then this relation is by definition in third normal form
 - If we add a relation that corresponds to an FD in the basis:
 - Can show: If the relation is not in 3NF, then the basis is not minimal

- First Normal Form: All values in a relation are atomic
 - This is removed by object-relational databases
- If the value of an attribute is a set, we represent it by using many relations

```
A B C
1 2 {3,4}
4 5 {3,4}
```

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

- A more practical example
 - Relation course (number, book, lecturer)
- In this department, the books recommended and the lecturers are independent.

```
calc 1 | Ross | Krenz
calc 1 | Lang | Krenz
calc 1 | Ross | Sanders
calc 1 | Lang | Sanders
calc 2 | Ash | Gillen
calc 2 | Ash | Engbers
calc 1 | Ross | Schwarz
calc 1 | Lang | Schwarz
```

The same list can be expressed using sets more simply

```
calc 1 | Ross
                 Krenz
calc 1 | Lang | Krenz
calc 1 | Ross
              | Sanders
calc 1 | Lang |
                 Sanders
calc 2 | Ash | Gillen
calc 2 | Ash | Engbers
                 Schwarz
calc 1 | Ross
calc 1
                 Schwarz
          Lang |
        {Ross, Lang} | {Krenz, Sanders, Schwarz}
calc1 |
                        {Gillen, Engbers}
calc2
        {Ash}
```

- It would be an error to add a single tuple
 - calc 1 | Burlow | Krenz
- to the relation
 - indicating that an additional book is now recommended
- Instead, need to add:
 - calc 1 | Burlow | Sanders
 - calc 1 | Burlow | Schwarz
- as well

- This gives rise to the definition of a multivalued dependency
 - Unlike before, we now demand that additional tuples exist in the relation.

- Formally: $A_1, A_2, ..., A_n \twoheadrightarrow B_1, ...B_m$
- Whenever
 - two tuples agree on its values in $A_1, A_2, ..., A_n$
 - the tuples have values $b_1...b_m$ and $b_1'...b_m'$ in B_1,B_2,\ldots,B_m
 - the tuples have values $x_1 ... x_r$ and $x_1' ... x_r'$ in the other attributes
 - then the tuples $a_1...a_nb_1'...b_m'x_1...x_r$ and $a_1'...a_n'b_1...b_mx_1'...x_r'$ also exist

- For each pair of tuples t and u of a relation R that agree on all attributes A_1, A_2, \ldots, A_n :
 - We can find another tuple v such that v agrees:
 - With both t and u on $A_1, A_2, ..., A_n$
 - With t on $B_1, B_2, ..., B_m$
 - With u on all attributes that are not among the As and Bs

- Example
 - Relation courses

```
calc 1
         Ross
                Krenz
calc 1 |
                Krenz
         Lang
calc 1 |
                Sanders
         Ross
calc 1 |
                Sanders
        Lang |
         Ash | Gillen
calc 2 |
                Engbers
calc 2 | Ash
                Schwarz
calc 1 |
         Ross
                Schwarz
calc 1
         Lang |
```

has FD course → book and course → lecturer

- Example stars(name, address, movie)
 - A star can have several address and can be in several movies

- Trivial MVD
 - If $\{B_1, ..., B_m\} \subset \{A_1, ..., A_n\}$ then
 - $A_1...A_n \Rightarrow B_1...B_m$

- Transitive MVDs
 - $A_1...A_n \twoheadrightarrow B_1...B_m$ and $B_1...B_m \twoheadrightarrow C_1...C_k$ implies $A_1...A_n \twoheadrightarrow C_1...C_k$
 - Provided that we remove any C-attributes that are also A-attributes

- Splitting is NOT true
 - stars(name, street, city, title, year)
- has MVD
 - name → street, city
- However, name → street is not true.
 - Wells Street is in Milwaukee
 - Glen Decker Ct. is in San José
 - C Rossell y Rios is in Montevideo

- Promotion
 - Any FD is also an MVD

- Complementation
 - If $A_1...A_n B_1...B_m$ and $C_1...C_k$ are the attributes not in the As and Bs, then $A_1...A_n C_1...C_k$

Fourth Normal Form

- A relation is in fourth normal form if whenever $A_1...A_n B_1...B_m$ is a non-trivial MVD
- Then $A_1 ... A_n$ is a super-key

Normal Forms

We have 4NF ⇒ BCNF ⇒ 3NF

	3NF	BCNF	4NF
eliminate redundancies due to FDs	no	yes	yes
eliminates redundancies due to MVDs	no	no	yes
preserves FDs	yes	no	no
preserves MVDs	no	no	no
lossless joins	yes	yes	yes