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

- About the nature of data
 - Think about it as the potential contents of a table
 - Instead of the actual contents
- Example:
 - Students can have a double major
 - But an actual set of students might not include a student with double major

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

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

- 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

• A **superkey** is a set of attributes in a table that determines all attributes

•
$$R(A_1, A_2, ..., A_n)$$

•
$$A_{i_1}, A_{i_2}, \ldots A_{i_m}$$
 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
 - Called transitivity

- 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:
 - $A_1, A_2, \ldots A_n \longrightarrow B_1, B_2, \ldots, B_m$ is equivalent to
 - $A_1, A_2, \dots A_n \longrightarrow B_1$
 - $A_1, A_2, \dots A_n \longrightarrow B_2$
 - •
 - $A_1, A_2, \dots A_n \longrightarrow B_m$

- The combining rule:
 - The set of FDs
 - $A_1, A_2, \dots A_n \longrightarrow B_1$
 - $A_1, A_2, \dots A_n \longrightarrow B_2$
 - $A_1, A_2, \dots A_n \longrightarrow B_m$
 - is equivalent to $A_1, A_2, \ldots A_n \longrightarrow B_1, B_2, \ldots, B_m$

• Quiz: Does $A_1, A_2, \dots, A_n \longrightarrow B$ imply $X, A_1, A_2, \dots, A_n \longrightarrow B$?

- Quiz: Does $A_1, A_2, \dots, A_n \longrightarrow B$ imply $X, A_1, A_2, \dots, A_n \longrightarrow B$?
- Yes:
 - Called augmentation

- Trivial FDs
 - $A_i \rightarrow A_i$ (Reflexivity)
 - $A, B \rightarrow A$ (Reflexivity & Augmentation)
- Trivial Dependency Rule:
 - $A_1A_2...A_n \rightarrow B_1B_2...B_m$ is equivalent to
 - $A_1 A_2 \dots A_n \to C_1 C_2 \dots C_r$
 - where the C_i are those of the B_i that are not among the A_i

- Closure:
 - Let ${\mathbb S}$ be a set of functional dependencies
 - Let $\mathbb{A} = \{A_1, A_2, \dots, A_n\}$ 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 $A_1, A_2, \dots, A_n \to B$.

- Closure calculation algorithm
 - Input: a set of attributes $\mathbb A$ and a set of functional dependencies $\mathbb S.$
 - Output: \mathbb{A}^+
- 1. Split all FDs in \mathbb{S} so that there is only a single attribute on the right 2.Set \mathbb{X} to be \mathbb{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} 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}
 - {E,F}
 - {E,F,H}
 - {E,F,H,K,L}
 - Hint: calculate the closure of all possible answers

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

- $\{\{E,F\} \rightarrow G, \{F\} \rightarrow I, \{F\} \rightarrow J, \{E,H\} \rightarrow K, \{E,H\} \rightarrow L, K \rightarrow M, L \rightarrow N\}$
 - 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: $\{E, F\}^+ = \{E, F, G, I, J\}$

- $\{\{E,F\} \rightarrow G, \{F\} \rightarrow I, \{F\} \rightarrow J, \{E,H\} \rightarrow K, \{E,H\} \rightarrow L, K \rightarrow M, L \rightarrow N\}$
 - Now try $\{E, F, H\} = X$
 - We can add G to X because of (1).
 - We can add I to X because of (2): $X = \{E, F, G, H, I\}$
 - We can add J to X because of (3): $X = \{E, F, G, H, I, J\}$
 - (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 :
 - $B \in \{A_1, A_2, \dots, A_n\}^+$ with regards to \mathbb{S}
 - Every relation fulfilling S fulfills $A_1A_2...A_n \to B$

- Why does
 - $B \in \{A_1, A_2, \dots, A_n\}^+$ with regards to \mathbb{S}
 - imply
 - Every relation fulfilling § fulfills $A_1A_2...A_n \to B$
- Look at the first time adding an attribute to X leads to an FD $A_1A_2...A_n \rightarrow B$ that is **not** true.
 - But *B* was added using a FD $X_1, X_2, \ldots X_m \to B$
 - Because this is the first time and $X_1, X_2, \ldots X_m$ follow from the $A_1A_2\ldots A_n$ in all relations, $A_1A_2\ldots A_n \to X_1, \ldots X_m$
 - Thus, a tuple equal in A₁A₂...A_n is also equal in all X₁,...X_m and hence equal in B.
 - Therefore $A_1A_2...A_n \rightarrow B$ has to be true and we have a contradiction

- Why does
 - Every relation fulfilling $\$ fulfills $A_1A_2...A_n \rightarrow B$
 - imply
 - $B \in \{A_1, A_2, \dots, A_n\}^+$ with regards to \mathbb{S}
- Assume $B \notin \{A_1, A_2, \dots, A_n\}^+$ 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 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 $\{A_1, A_2, \ldots, A_n\}^+$ and the D on the right side of the table.

{A1	A2	•••	An}+	e	eve	ry	thi	ng	else
0	0		0		0	С) .	•••	0
0	0		0		1	1	- •	•••	1

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

• 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 0 0 0 0 1 1 0 1

• Yes!

Therefore the assumption is violated and this finishes the proof

- With the closure calculation, we can prove
 - If in a relation R $A_1, A_2, \dots, A_m \to B_1, B_2, \dots, B_n$ and $B_1, B_2, \dots, B_n \to C_1, C_2, \dots C_t$ then $A_1, A_2, \dots, A_m \to C_1, C_2, \dots C_t$
 - 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> ₿:
 - All FDs in $\mathbb B$ have singleton right sides
 - Removing any FD from ${\mathbb B}$ is no longer a basis
 - 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
 - $A \rightarrow B, A \rightarrow C$ and all augmentations $A \rightarrow B, C$ including the trivial ones $A \rightarrow A, B, A \rightarrow A, C$ and $A \rightarrow A, B, C$
 - $B \rightarrow A$, $B \rightarrow C$ 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 $\{A \to C, B \to A, B \to C, C \to A, C \to B\}$

- 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 $\pi_L(R)$?

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

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

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

- We really do not need any more because those with two attributes on the left would follow trivially
 - Now we add the FDs derived from the closure, if all attributes are in L
 - $\mathbb{T} = \{A \to C, A \to D, C \to D\}$
 - 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, starName)

- 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 is not a key.
 - 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: $A \not\rightarrow B$, $B \not\rightarrow A$
 - No nontrivial FDs exists, R is in BCNF
 - Case 2: $A \rightarrow B, B \not\rightarrow A$
 - *A* is the only key and it is on the right of the only non-trivial FD. So BNCF.
 - Case 3: $A \not\rightarrow B, B \rightarrow B$
 - 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

•
$$A_1 A_2 \dots A_n \to B_1 \dots B_m$$

• 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

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

Answer

- $R(A, B, C, D); AB \rightarrow C; C \rightarrow D; D \rightarrow A$
- Keys are (A,B), (C,B), (D,B}
- $C \rightarrow D$ violates Boyce Codd
 - $\{C\}^+ = \{C, D, A\}$
 - $X \cup C(X^+) = \{C, B\}$
 - We find that $D \rightarrow A$ is still a violation is R(A,C,D).
 - $\{D\}^+ = \{D, A\}$
 - Complement is $\{D, C\}$

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

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

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

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

- 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 $B \rightarrow C$ 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

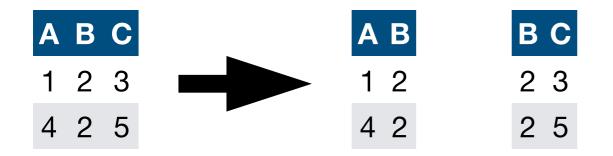
• Keys are: title, city and title, theater

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



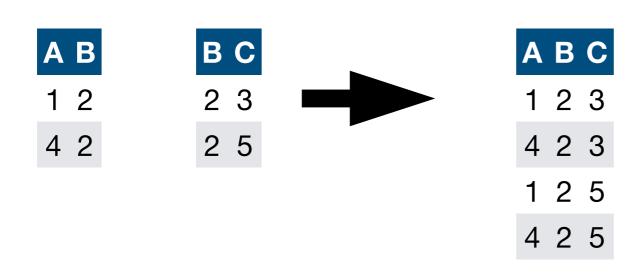
• Split into $R_1(A, B)$ and $R_2(B, C)$

• 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 $B \rightarrow C$ has a lossless join into R(A, B) and R(B, C)
- Without FD $B \rightarrow C$ or $B \rightarrow A$, the join is not loss-less

- Question: Given a set of FDs in R and a set of sets of attributes $S_1, S_2, \ldots S_n$:
 - 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 $R \subset \pi_{S_1}(R) \bowtie \pi_{S_2}(R) \bowtie \ldots \bowtie \pi_{S_n}(R)$

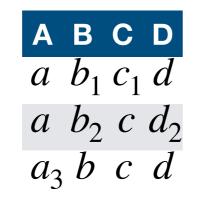
- Chase Test:
 - Task: Show that given the FDs, we can prove that

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

- 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

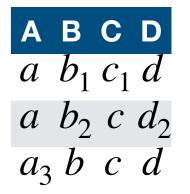


- The first row looks at the projection on A and D
 - From the projection, we know that a given tuple has certain a and d values, but the join might give some values for the b and c column

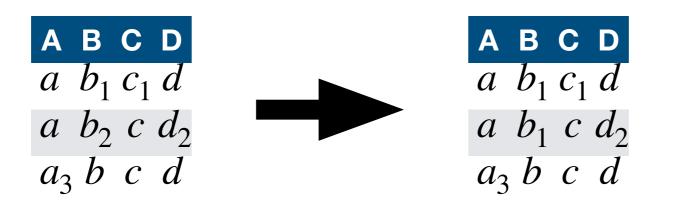
Α	В	С	D
a	b_1	<i>c</i> ₁	d
a	b_2	С	d_2
a_3	b	С	d

- 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

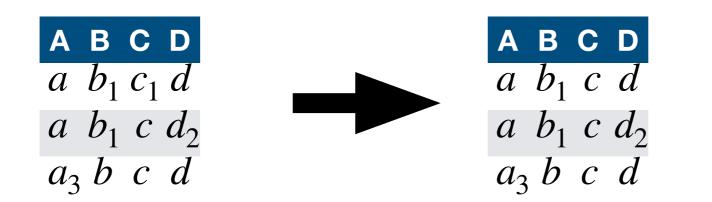
- Example:
 - Assume the following FDs for the example:
 - $A \to B, B \to C, CD \to A$
 - Whenever we have tableau entries for attributes on the right side, we can use it to equalize the entries for attributes on the right of an FD



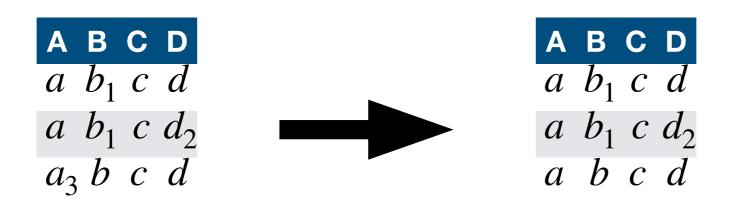
- Use $A \rightarrow B$:
 - First two rows, we have unsubscripted a.
 - Equalize the B column in these rows



• Use FD $B \to C$



• Now use $CD \to A$



- 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 $B \rightarrow AD$
 - $\{B\}^+ = \{B, A, D\}$, so Boyce-Codd would split into
 - R(B, C) and R(A, B, D)
 - But we decompose into $\{A, B\}, \{B, C\}, \{C, D\}$

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

$$\begin{array}{cccc} A & B & C & D \\ a & b & c_1 & d_1 \\ a_2 & b & c & d_2 \\ a_3 & b_3 & c & d \end{array}$$

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

 $a_3b_3 c d$

• Initial tableau is $a \ b \ c_1 \ d_1$ $a_2 \ b \ c \ d_2$

• After applying the FD, we get tableau

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

- Take this tableau and use it to construct a counter example
 - A B C D $a b c_1 d_1$ $a b c d_1$ $a_3 b_3 c d$
 - Create tuples (a, b, c_1, d_1) , (a, b, c, d_1) , (a_3, b_3, c, d) in R.
 - Fulfills the FD $B \rightarrow CD$
 - Projections are

ABBCCD
$$a$$
 b b_1 c_1 c_1 d_1 a_3 b_3 b c c d_1 b_3 c c d

• Join these together:

• Result has two additional rows

Α	В	С	D
a	b	<i>c</i> ₁	d_1
a	b	С	d_1^{-}
a	b	С	d^{-}
a_3	b_3	С	d_1
a_3	b_3		d

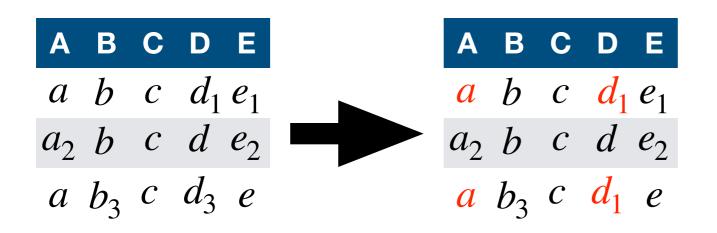
• The decomposition is not loss-less!

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

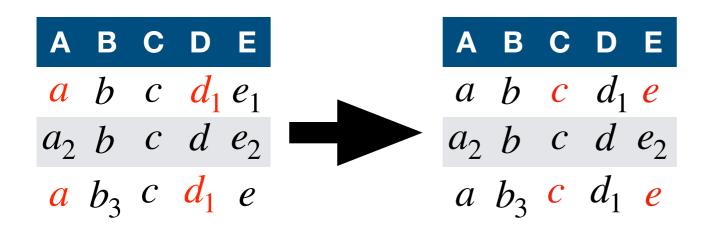
• Let R(A, B, C, D, E) be decomposed into $\{A, B, C\}$, $\{B, C, D\}, \{A, C, E\}$. Assume FDs $A \rightarrow D, CD \rightarrow E$, $E \rightarrow D$. Is the decomposition lossless?

Α	В	С	D	Е
			d_1	
a_2	b	С	d	e_2
a	b_3	С	d_3	е

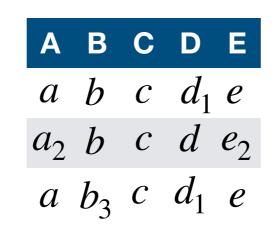
- Let R(A, B, C, D, E) be decomposed into $\{A, B, C\}$, $\{B, C, D\}, \{A, C, E\}$. Assume FDs $A \rightarrow D, CD \rightarrow E$, $E \rightarrow D$. Is the decomposition lossless?
 - Use $FDA \rightarrow D$



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



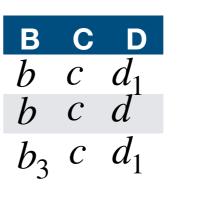
- The tableau gives us tuples that satisfy the FDs
- Make the tableau into tuples
- Look at the projections

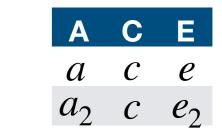
BCDE	ABC	BCD
$a b c d_1 e$	a b c	$b c d_1$
1	$a_2 b c$	b c d
$a_2 b c d e_2$	$a b_3 c$	$b_3 c d_1$
$a b_3 c d_1 e$	5	

• Join them

Α	В	С	D	Е
a	b	С	d_1	е
a_2	b	С	d	e_2
a	b_3	С	d_1	e

Α	B	С
a	b	С
a_2	b	С
a	b_{3}	С





Α	Β	С	D	Ε
a	b	С	d_1	e
a	b	С		е
a_2	b	С	d_1	e_2
a_2	b	С	đ	e_2
a	b_3	С	d_1	e^{-}
	b_3		d_1	e_2

- R(ABCDE) decomposed into $R_1(AD)$, $R_2(AB)$, $R_3(BE)$, $R_4(CDE)$, $R_5(AE)$
- Functional dependencies are $A \rightarrow C, B \rightarrow C, C \rightarrow D,$ $DE \rightarrow C$, and $CE \rightarrow A$

- R(ABCDE) decomposed into $R_1(AD)$, $R_2(AB)$, $R_3(BE)$, $R_4(CDE)$, $R_5(AE)$
- Functional dependencies are $A \to C$, $B \to C$, $C \to D$, $DE \to C$, and $CE \to A$
- Create a tableau with one row for each decomposition

- R(ABCDE) decomposed into $R_1(AD)$, $R_2(AB)$, $R_3(BE)$, $R_4(CDE)$, $R_5(AE)$
- Functional dependencies are $A \to C$, $B \to C$, $C \to D$, $DE \to C$, and $CE \to A$
- Create a tableau with one row for each decomposition
 - This represent potential values in the join

Α	В	С	D	Е
а	b1	C 1	d	e1
а	b	C 2	d ₂	e ₂
a ₃	b	C 3	d ₃	е
a 4	b 4	С	d	е
а	b_5	C 5	d_5	е

- R(ABCDE) decomposed into $R_1(AD)$, $R_2(AB)$, $R_3(BE)$, $R_4(CDE)$, $R_5(AE)$
- Functional dependencies are $A \to C, B \to C, C \to D, DE \to C$, and $CE \to A$
- Now chase using functional dependencies
 - Start with $A \rightarrow C$: Any tuple with the same A-value gets simplified

Α	В	С	D	Е
а	b1	C ₁	d	e1
а	b	C 2	d ₂	e ₂
a_3	b	C 3	d ₃	е
a 4	b 4	С	d	е
а	b 5	C 5	d_5	е

- R(ABCDE) decomposed into $R_1(AD)$, $R_2(AB)$, $R_3(BE)$, $R_4(CDE)$, $R_5(AE)$
- Functional dependencies are $A \to C$, $B \to C$, $C \to D$, $DE \to C$, and $CE \to A$
- Now chase using functional dependencies
 - Start with $A \rightarrow C$: Any tuple with the same A-value gets simplified

Α	В	С	D	Е
а	b1	C ₁	d	e1
а	b	C ₂	d ₂	e ₂
a 3	b	C 3	d ₃	е
a 4	b 4	С	d	е
а	b 5	C 5	d_5	е

- R(ABCDE) decomposed into $R_1(AD)$, $R_2(AB)$, $R_3(BE)$, $R_4(CDE)$, $R_5(AE)$
- Functional dependencies are $A \to C$, $B \to C$, $C \to D$, $DE \to C$, and $CE \to A$
- Now chase using functional dependencies
 - Start with $A \rightarrow C$: Any tuple with the same A-value gets simplified

Α	В	С	D	Е
а	b1	C ₁	d	e1
а	b	C 2	d ₂	e ₂
a 3	b	C 3	d ₃	е
a 4	b 4	С	d	е
а	b 5	C 5	d_5	е

- R(ABCDE) decomposed into $R_1(AD)$, $R_2(AB)$, $R_3(BE)$, $R_4(CDE)$, $R_5(AE)$
- Functional dependencies are $A \to C$, $B \to C$, $C \to D$, $DE \to C$, and $CE \to A$
- Now chase using functional dependencies
 - Start with $A \rightarrow C$: Any tuple with the same A-value gets simplified

Α	В	С	D	Е
а	b1	C ₁	d	e1
а	b	C1	d ₂	e ₂
a 3	b	C 3	d ₃	е
a 4	b 4	С	d	е
а	b 5	C1	d_5	е

- R(ABCDE) decomposed into $R_1(AD)$, $R_2(AB)$, $R_3(BE)$, $R_4(CDE)$, $R_5(AE)$
- Functional dependencies are $A \to C, B \to C, C \to D, DE \to C$, and $CE \to A$
- Now chase using functional dependencies
 - Start with $A \rightarrow C$: Any tuple with the same A-value gets simplified

Α	В	С	D	Е
а	b1	C ₁	d	e1
а	b	C ₁	d ₂	e ₂
a_3	b	C 3	d ₃	е
a 4	b 4	С	d	е
а	b 5	C 1	d_5	е

- R(ABCDE) decomposed into $R_1(AD)$, $R_2(AB)$, $R_3(BE)$, $R_4(CDE)$, $R_5(AE)$
- Functional dependencies are $A \to C, B \to C, C \to D, DE \to C$, and $CE \to A$
- Now chase using functional dependencies
 - Apply $B \to C$: Any tuple with the same B-value gets simplified

Α	В	С	D	Е
а	b1	C ₁	d	e1
а	b	C ₁	d ₂	e ₂
a ₃	b	C 3	d ₃	е
a 4	b 4	С	d	е
а	b 5	C 1	d_5	е

- R(ABCDE) decomposed into $R_1(AD)$, $R_2(AB)$, $R_3(BE)$, $R_4(CDE)$, $R_5(AE)$
- Functional dependencies are $A \to C, B \to C, C \to D, DE \to C$, and $CE \to A$
- Now chase using functional dependencies
 - Apply $B \rightarrow C$: Any tuple with the same B-value gets simplified

Α	В	С	D	Е
а	b1	C ₁	d	e1
а	b	C 1	d ₂	e ₂
a 3	b	C 3	d ₃	е
a_4	b 4	С	d	е
а	b ₅	C 1	d_5	е

- R(ABCDE) decomposed into $R_1(AD)$, $R_2(AB)$, $R_3(BE)$, $R_4(CDE)$, $R_5(AE)$
- Functional dependencies are $A \to C, B \to C, C \to D, DE \to C$, and $CE \to A$
- Now chase using functional dependencies
 - Apply $B \rightarrow C$: Any tuple with the same B-value gets simplified

Α	В	С	D	Е
а	b1	C 1	d	e1
а	b	C ₁	d ₂	e ₂
a ₃	b	C ₁	d ₃	е
a_4	b 4	С	d	е
а	b ₅	C 1	d_5	е

- R(ABCDE) decomposed into $R_1(AD)$, $R_2(AB)$, $R_3(BE)$, $R_4(CDE)$, $R_5(AE)$
- Functional dependencies are $A \to C, B \to C, C \to D, DE \to C$, and $CE \to A$
- Now chase using functional dependencies
 - Apply $C \rightarrow D$:

Α	В	С	D	Е
а	b1	C 1	d	e1
а	b	C 1	d ₂	e ₂
a 3	b	C 1	d ₃	е
a 4	b 4	С	d	е
а	b_5	C 1	d_5	е

- R(ABCDE) decomposed into $R_1(AD)$, $R_2(AB)$, $R_3(BE)$, $R_4(CDE)$, $R_5(AE)$
- Functional dependencies are $A \to C, B \to C, C \to D, DE \to C$, and $CE \to A$
- Now chase using functional dependencies
 - Apply $C \rightarrow D$:

Α	В	С	D	Е
а	b1	C1	d	e1
а	b	C 1	d ₂	e ₂
a_3	b	C1	d ₃	е
a 4	b 4	С	d	е
а	b_5	C 1	d_5	е

- R(ABCDE) decomposed into $R_1(AD)$, $R_2(AB)$, $R_3(BE)$, $R_4(CDE)$, $R_5(AE)$
- Functional dependencies are $A \to C, B \to C, C \to D, DE \to C$, and $CE \to A$
- Now chase using functional dependencies
 - Apply $C \rightarrow D$:

Α	В	С	D	Е
а	b1	C 1	d	e1
а	b	C 1	d	e ₂
a ₃	b	C 1	d	е
a 4	b 4	С	d	е
а	b_5	C 1	d	е

- R(ABCDE) decomposed into $R_1(AD)$, $R_2(AB)$, $R_3(BE)$, $R_4(CDE)$, $R_5(AE)$
- Functional dependencies are $A \to C$, $B \to C$, $C \to D$, $DE \to C$, and $CE \to A$
- Now chase using functional dependencies
 - Apply $DE \rightarrow C$:

Α	В	С	D	Е
а	b1	C ₁	d	e1
а	b	C ₁	d	e ₂
a 3	b	C ₁	d	е
a 4	b 4	С	d	е
а	b 5	C 1	d	е

- R(ABCDE) decomposed into $R_1(AD)$, $R_2(AB)$, $R_3(BE)$, $R_4(CDE)$, $R_5(AE)$
- Functional dependencies are $A \to C$, $B \to C$, $C \to D$, $DE \to C$, and $CE \to A$
- Now chase using functional dependencies
 - Apply $DE \rightarrow C$: Unify the C-value if the D and E value are same

Α	В	С	D	Е
а	b1	C 1	d	e1
а	b	C 1	d	e ₂
a_3	b	C 1	d	е
a 4	b 4	С	d	е
а	b_5	C 1	d	е

- R(ABCDE) decomposed into $R_1(AD)$, $R_2(AB)$, $R_3(BE)$, $R_4(CDE)$, $R_5(AE)$
- Functional dependencies are $A \to C$, $B \to C$, $C \to D$, $DE \to C$, and $CE \to A$
- Now chase using functional dependencies
 - Apply $DE \rightarrow C$: Unify the C-value if the D and E value are same

Α	В	С	D	E
а	b1	C ₁	d	e1
а	b	C 1	d	e ₂
a 3	b	С	d	е
a 4	b 4	С	d	е
а	b 5	С	d	е

- R(ABCDE) decomposed into $R_1(AD)$, $R_2(AB)$, $R_3(BE)$, $R_4(CDE)$, $R_5(AE)$
- Functional dependencies are $A \to C$, $B \to C$, $C \to D$, $DE \to C$, and $CE \to A$
- Now chase using functional dependencies
 - Apply $DE \rightarrow C$: Unify the C-value if the D and E value are same

Α	В	С	D	Е
а	b1	C ₁	d	e1
а	b	C 1	d	e ₂
a ₃	b	С	d	е
a_4	b 4	С	d	е
а	b 5	С	d	е

- R(ABCDE) decomposed into $R_1(AD)$, $R_2(AB)$, $R_3(BE)$, $R_4(CDE)$, $R_5(AE)$
- Functional dependencies are $A \to C, B \to C, C \to D, DE \to C$, and $CE \to A$
- Now chase using functional dependencies
 - Apply $CE \to A$

Α	В	С	D	Е
а	b1	C 1	d	e1
а	b	C 1	d	e ₂
a ₃	b	С	d	е
a 4	b 4	С	d	е
а	b 5	С	d	е

- R(ABCDE) decomposed into $R_1(AD)$, $R_2(AB)$, $R_3(BE)$, $R_4(CDE)$, $R_5(AE)$
- Functional dependencies are $A \to C, B \to C, C \to D, DE \to C$, and $CE \to A$
- Now chase using functional dependencies
 - Apply $CE \to A$

Α	В	С	D	Е
а	b1	C ₁	d	e ₁
а	b	C 1	d	e ₂
a 3	b	С	d	е
a 4	b 4	С	d	е
а	b 5	С	d	е

- R(ABCDE) decomposed into $R_1(AD)$, $R_2(AB)$, $R_3(BE)$, $R_4(CDE)$, $R_5(AE)$
- Functional dependencies are $A \to C, B \to C, C \to D, DE \to C$, and $CE \to A$
- Now chase using functional dependencies
 - Apply $CE \to A$

Α	В	С	D	Е
а	b1	C 1	d	e1
а	b	C 1	d	e ₂
a ₃	b	С	d	е
a 4	b4	С	d	е
а	b ₅	С	d	е

- R(ABCDE) decomposed into $R_1(AD)$, $R_2(AB)$, $R_3(BE)$, $R_4(CDE)$, $R_5(AE)$
- Functional dependencies are $A \to C, B \to C, C \to D, DE \to C$, and $CE \to A$
- Now chase using functional dependencies
 - Apply $CE \to A$

Α	В	С	D	Е
а	b1	C 1	d	e1
а	b	C 1	d	e ₂
а	b	С	d	е
а	b4	С	d	е
а	b ₅	С	d	е

- R(ABCDE) decomposed into $R_1(AD)$, $R_2(AB)$, $R_3(BE)$, $R_4(CDE)$, $R_5(AE)$
- Functional dependencies are $A \to C, B \to C, C \to D, DE \to C$, and $CE \to A$
- Now chase using functional dependencies
 - Apply $CE \to A$

Α	В	С	D	Е
а	b1	C ₁	d	e1
а	b	C ₁	d	e ₂
а	b	С	d	е
а	b 4	С	d	е
а	b 5	С	d	е

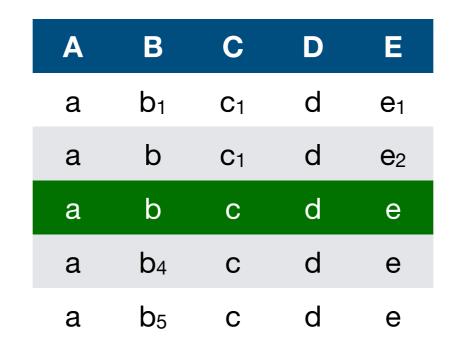
- R(ABCDE) decomposed into $R_1(AD)$, $R_2(AB)$, $R_3(BE)$, $R_4(CDE)$, $R_5(AE)$
- Functional dependencies are $A \to C$, $B \to C$, $C \to D$, $DE \to C$, and $CE \to A$
- Now chase using functional dependencies
 - While we can continue, we do not have to:

Α	В	С	D	Е
а	b1	C ₁	d	e1
а	b	C ₁	d	e ₂
а	b	С	d	е
а	b 4	С	d	е
а	b 5	С	d	е

- R(ABCDE) decomposed into $R_1(AD)$, $R_2(AB)$, $R_3(BE)$, $R_4(CDE)$, $R_5(AE)$
- Functional dependencies are $A \to C$, $B \to C$, $C \to D$, $DE \to C$, and $CE \to A$
- Now chase using functional dependencies
 - Middle row has no indexed values:

Α	В	С	D	Е
а	b1	C 1	d	e1
а	b	C 1	d	e ₂
а	b	С	d	е
а	b 4	С	d	е
а	b_5	С	d	е

• This shows that the decomposition has a lossless join



- Decomposition
 - Avoid update and delete anomalies
 - Guarantee lossless joins
 - Decomposition **does not** generate spurious data
 - Want to check functional dependencies in the derived tables

- Sometimes, checking FDs is impossible
 - Example: R(street, city, zip)
 - street, city -> zip
 - zip -> city
 - Bring into BCNF: {street, city} and {street, zip} are keys, zip -> city violates BCNF
 - Decompose into R1(zip, city), R2(zip, street)
 - But now we cannot check street, city -> zip

• Example:



looks fine

• But join gives:

street	city	zip
123 Wisconsin Avenue	Milwaukee	53230
123 Wisconsin Avenue	Milwaukee	53231

• which violates a FD

- So we should not do this.
- "Elegant" solution: define the problem away
 - The original table needs to be "normal"
- A relation is in third normal form iff
 - For every non-trivial FD $X \to A$
 - X is a superkey
 - A is prime (member of at least one key

- A relation *R* is in *third normal form*
 - If $A_1A_2...A_n \to B_1B_2...B_m$ is a non-trivial FD, then
 - either $\{A_1, A_2, \dots, A_n\}$ is a superkey
 - or those of $\{B_1, B_2, ..., B_m\}$ not in $\{A_1, A_2, ..., A_n\}$ 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

- Example:
 - addresses(city, street, zip)

```
zip --> city
```

city, street --> zip

- is in third normal form
 - zip is prime

- 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 $\mathbb G$ for $\mathbb F$
 - 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: R(A, B, C, D, E) with FDs $AB \rightarrow C, C \rightarrow B$, $A \rightarrow D$
- 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 K.
 - The closure of ${\mathbb K}$ 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 $\mathbb{X} \supset \mathbb{A} \rightarrow B$.
 - Tableau:

AX\ABrest of attributesrow Kr,s,t, e, f, b1**row FDr,s t1 e1 f1 b**

- 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
 A
 B
 C

 1
 2

$$\{3, 4\}$$
 1
 2
 3

 4
 5
 $\{3, 4\}$
 1
 2
 4

 4
 5
 $\{3, 4\}$
 1
 2
 4

 4
 5
 $\{3, 4\}$
 1
 2
 4

 4
 5
 $\{3, 4\}$
 1
 2
 4

 4
 5
 $\{3, 4\}$
 4
 5
 3

- 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
calc	1	Ross	Schwarz
calc	1	Lang	Schwarz

calc1 | {Ross, Lang} | {Krenz, Sanders, Schwarz} calc2 | {Ash} | {Gillen, Engbers}

- 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: $A_1, A_2, \dots, A_n \twoheadrightarrow B_1, \dots B_m$
- Whenever
 - two tuples agree on its values in A_1, A_2, \dots, A_n
 - the tuples have values $b_1 \dots b_m$ and $b_1' \dots b_m'$ in B_1, B_2, \dots, B_m
 - the tuples have values x₁...x_r and x'₁...x'_r in the other attributes
 - then the tuples $a_1 \dots a_n b'_1 \dots b'_m x_1 \dots x_r$ and $a'_1 \dots a'_n b_1 \dots b_m x'_1 \dots x'_r$ 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, \ldots, 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		Lang		Krenz
calc	1		Ross		Sanders
calc	1		Lang		Sanders
calc	2		Ash		Gillen
calc	2		Ash		Engbers
calc	1		Ross		Schwarz
	—	I	11000	I	DCIIWALZ

has FD course → book and course → lecturer

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

- But: for each movie, all addresses of the star need to appear
- But: for each address, all movies need to appear

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

- 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.
 - John Wayne, 123 Elm Street, Malibu, Hatari, 1962
 - John Wayne, 456 Overland, Culver City, Hatari, 1962
 - John Wayne, 123 Elm Street, Malibu, Shootist, 1976
 - John Wayne, 456 Overland, Culver City, Shootist, 1976
 - DOES NOT HAVE
 - John Wayne, 123 Elm Street, Culver City, Hatari, 1962

- Promotion
 - Any FD is also an MVD

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

Fourth Normal Form

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

Normal Forms

• We have $4NF \Rightarrow BCNF \Rightarrow 3NF$

	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