COMP2400 Relational Databases

Assignment 2

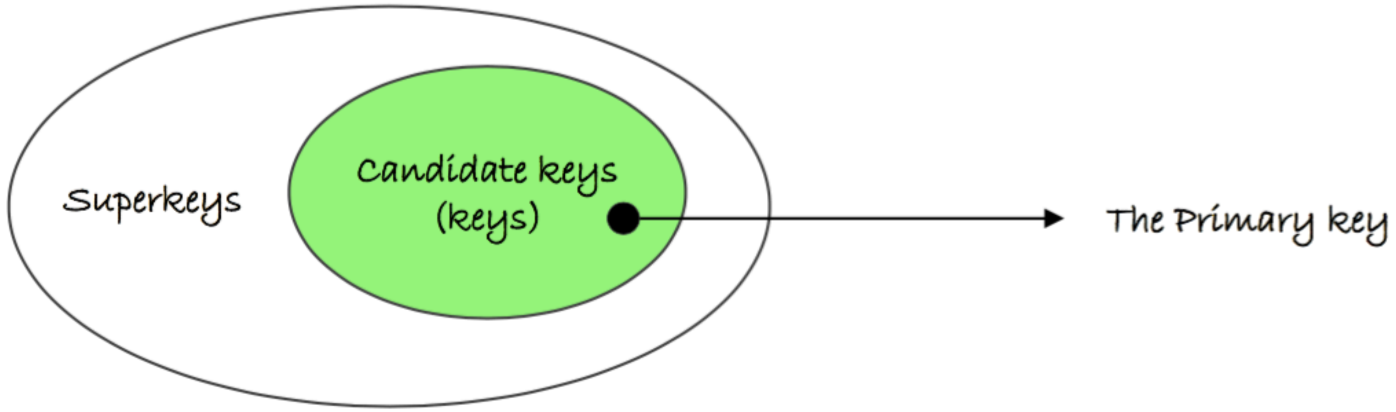
25 September 2016

Question 1

Suppose R is a relation with four attributes ABCD and the only keys are AB and BD. How many superkeys R has? Explain why.

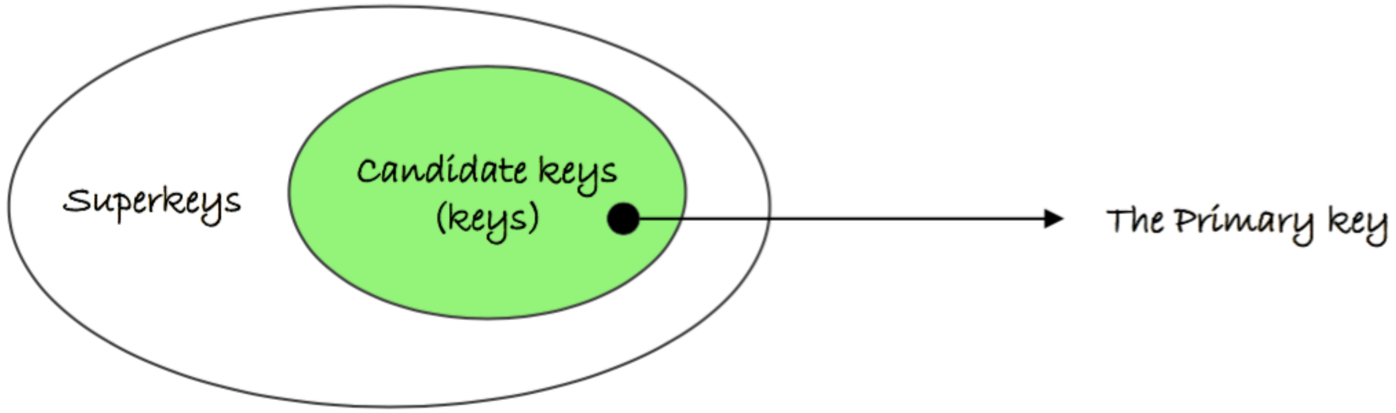
Question 1

Suppose R is a relation with four attributes ABCD and the only keys are AB and BD. How many superkeys R has? Explain why.



Question 1

Suppose R is a relation with four attributes ABCD and the only keys are AB and BD. How many superkeys R has? Explain why.



Idea: Add remaining attributes to known keys one by one, and remove duplicates.

Question 1

Suppose R is a relation with four attributes ABCD and the only keys are AB and BD. How many superkeys R has? Explain why.

Candidates:

1. ABC
2. ABD
3. ABCD
4. ABD
5. BCD
6. ABCD

Question 1

Suppose R is a relation with four attributes ABCD and the only keys are AB and BD. How many superkeys R has? Explain why.

Solution:

1. ABC
2. ABD
3. ABCD
4. BCD

Thus, the total number of superkeys R has is 4.

Question 2

Consider a relation schema R with five attributes ABCDE and the following functional dependencies on R:

$$\Sigma = \{A \rightarrow C, C \rightarrow E, EB \rightarrow A, A \rightarrow E, AD \rightarrow B\}$$

- 2.1 Find all the keys of R with respect to Σ. Justify your answer (i.e., include the steps used for finding the keys).
- 2.2 Does $\Sigma \models CD \rightarrow B$ hold? If it holds, explain why it holds. If it does not hold, give a relation that contains only two tuples, as a counterexample, to show it does not hold.
- 2.3 Does Σ contain any redundant functional dependencies? A functional dependency $X \rightarrow Y$ is redundant if we remove it from Σ, and we can still infer $X \rightarrow Y$ from the functional dependencies in $\Sigma - \{X \rightarrow Y\}$. Justify your answer.

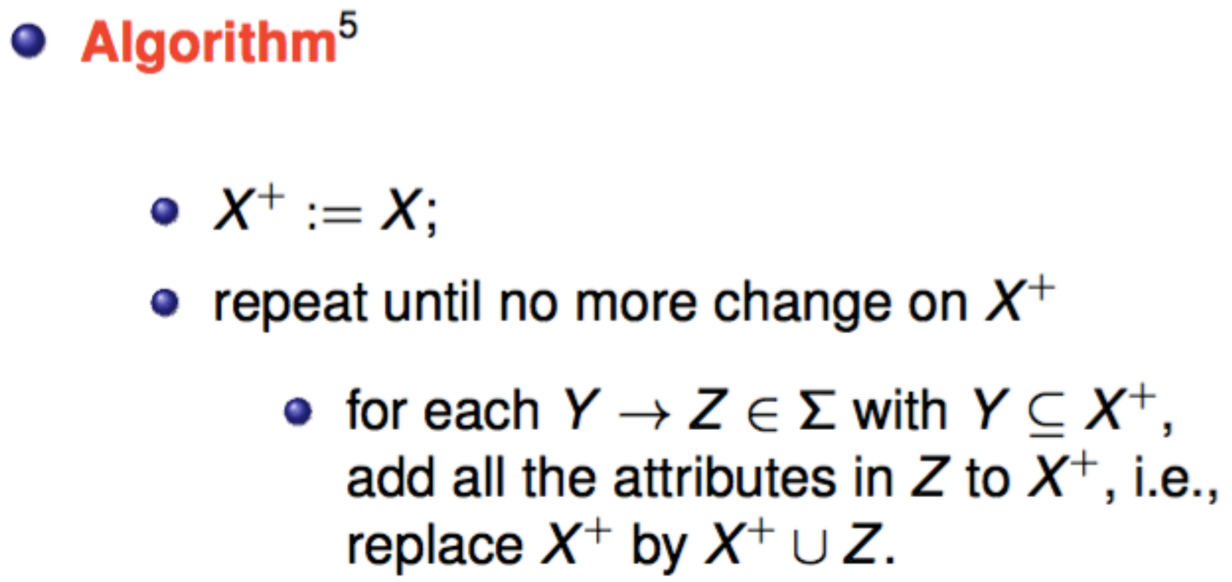
Question 2

Consider a relation schema R with five attributes ABCDE and the following functional dependencies on R:

$$\Sigma = \{A \rightarrow C, C \rightarrow E, EB \rightarrow A, A \rightarrow E, AD \rightarrow B\}$$

- 2.1 Find all the keys of R with respect to Σ. Justify your answer (i.e., include the steps used for finding the keys).

Idea: Compute the closure *X*+ for every subset *X* of the relation *R*.



- **Algorithm**[5]
  - $X^+ := X$;
  - repeat until no more change on X^+
    - for each $Y \rightarrow Z \in \Sigma$ with $Y \subseteq X^+$, add all the attributes in Z to X^+, i.e., replace X^+ by $X^+ \cup Z$.

Question 2

Consider a relation schema R with five attributes ABCDE and the following functional dependencies on R:

$$\Sigma = \{A \rightarrow C, C \rightarrow E, EB \rightarrow A, A \rightarrow E, AD \rightarrow B\}$$

- 2.1 Find all the keys of R with respect to Σ. Justify your answer (i.e., include the steps used for finding the keys).

Solution:

(AD)+ = ABCDE

(BCD)+ = ABCDE

(BDE)+ = ABCDE

So the keys are: AD, BCD, BDE.

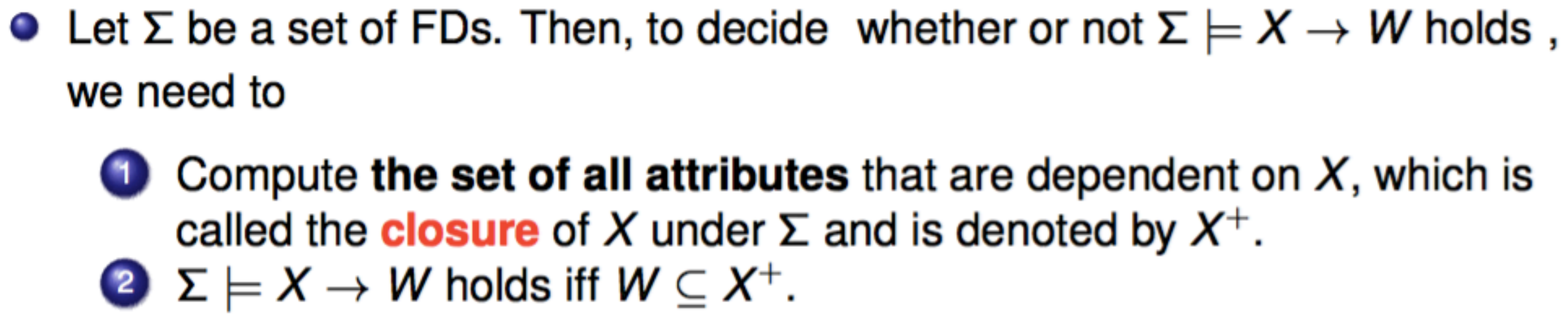
Question 2

Consider a relation schema R with five attributes ABCDE and the following functional dependencies on R:

$$\Sigma = \{A \rightarrow C, C \rightarrow E, EB \rightarrow A, A \rightarrow E, AD \rightarrow B\}$$

- 2.2 Does $\Sigma \models CD \rightarrow B$ hold? If it holds, explain why it holds. If it does not hold, give a relation that contains only two tuples, as a counterexample, to show it does not hold.

Idea:



- Let Σ be a set of FDs. Then, to decide whether or not $\Sigma \models X \rightarrow W$ holds , we need to
  1. Compute **the set of all attributes** that are dependent on X, which is called the **closure** of X under Σ and is denoted by X^+.
  2. $\Sigma \models X \rightarrow W$ holds iff $W \subseteq X^+$.

Question 2

Consider a relation schema R with five attributes ABCDE and the following functional dependencies on R:

$$\Sigma = \{A \rightarrow C, C \rightarrow E, EB \rightarrow A, A \rightarrow E, AD \rightarrow B\}$$

- 2.2 Does $\Sigma \models CD \rightarrow B$ hold? If it holds, explain why it holds. If it does not hold, give a relation that contains only two tuples, as a counterexample, to show it does not hold.

Solution:

$(CD)^+ = (CDE)$ (does not have B as a member), so the implied functional dependency does not hold.

Counterexample:

(a1, b1, c, d, e)

(a2, b2, c, d, e)

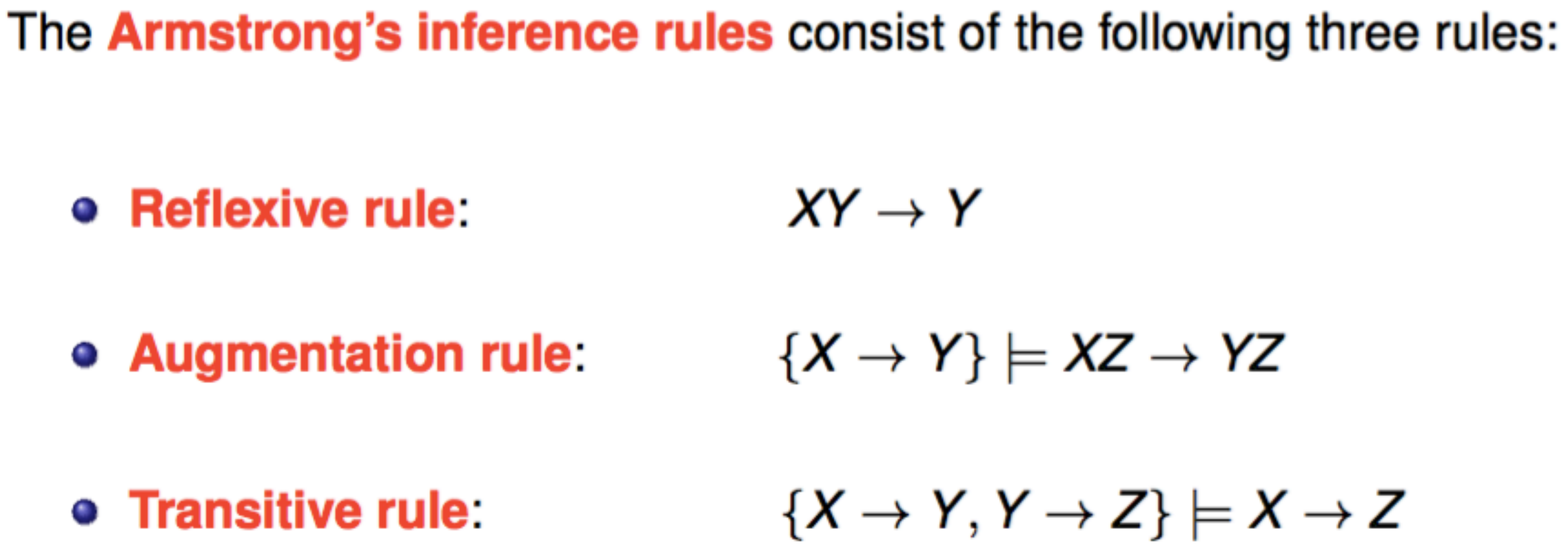
Question 2

Consider a relation schema R with five attributes ABCDE and the following functional dependencies on R:

$$\Sigma = \{A \rightarrow C, C \rightarrow E, EB \rightarrow A, A \rightarrow E, AD \rightarrow B\}$$

- 2.3 Does Σ contain any redundant functional dependencies? A functional dependency X→Y is redundant if we remove it from Σ, and we can still infer X→Y from the functional dependencies in Σ − {X → Y }. Justify your answer.

Idea: Test one by one. If the removed test functional dependency can be implied from remaining functional dependencies, then it is redundant.



The **Armstrong's inference rules** consist of the following three rules:

- **Reflexive rule**: $XY \rightarrow Y$
- **Augmentation rule**: $\{X \rightarrow Y\} \models XZ \rightarrow YZ$
- **Transitive rule**: $\{X \rightarrow Y, Y \rightarrow Z\} \models X \rightarrow Z$

Question 2

Consider a relation schema R with five attributes ABCDE and the following functional dependencies on R:

$$\Sigma = \{A \rightarrow C, C \rightarrow E, EB \rightarrow A, A \rightarrow E, AD \rightarrow B\}$$

- 2.3 Does Σ contain any redundant functional dependencies? A functional dependency $X \rightarrow Y$ is redundant if we remove it from Σ, and we can still infer $X \rightarrow Y$ from the functional dependencies in $\Sigma - \{X \rightarrow Y\}$. Justify your answer.

Solution:

If we remove A->E, we can get A->E back by using *Transitive Rule* on {A->C, C->E}.

Question 3

Consider a relation schema R with five attributes ABCDE and the following functional dependencies on R:

$$\Sigma = \{AB \rightarrow C, BD \rightarrow E, ACD \rightarrow E, AC \rightarrow B\}$$

(1) Is the decomposition {ABC,BCDE} dependency-preserving? Justify your answer.

(2) Is the decomposition {ABC,BCDE} lossless? Justify your answer. If your answer is negative, show how a relation over ABCDE is not lossless after being decomposed into two relations over ABC or BCDE.

Question 3

1. **Lossless join** – " **capture the same data** "

   To disallow the possibility of generating spurious tuples when a NATURAL JOIN operation is applied to the relations after decomposition.

2. **Dependency preservation** – " **capture the same meta-data** "

   To ensure that each functional dependency can be inferred from functional dependencies after decomposition.

Question 3 (1)

(1) Is the decomposition {ABC,BCDE} dependency-preserving? Justify your answer.

Idea: Split Σ into two subsets, one that definitely hold after decomposition, another one may or may not hold any more.

Solution:

{AB->C, BD->E, AC->B} still holds after decomposition. {ACD->E} may or may not hold. We test it.

Question 3 (1)

Test whether {ACD->E} holds or not:

(ACD)+ = (ABCD)+ = ABCDE

(ACD)+ = (ABCD)+ because of AC->B, and (ABCD)+ = ABCDE because of BD->E.

Therefore, the decomposition is dependency-preserving.

Question 3 (2)

(2) Is the decomposition {ABC,BCDE} lossless? Justify your answer. If your answer is negative, show how a relation over ABCDE is not lossless after being decomposed into two relations over ABC or BCDE.

Idea: The only possibility that doing a NATURAL JOIN over the decomposition would generate spurious data is the overlapping attributes cannot form a superkey of the original relation. Hence, we only need to check whether BC can form a superkey of R.

Question 3 (2)

(2) Is the decomposition {ABC,BCDE} lossless? Justify your answer. If your answer is negative, show how a relation over ABCDE is not lossless after being decomposed into two relations over ABC or BCDE.

Solution:

$$R: \Sigma = \{AB \rightarrow C, BD \rightarrow E, ACD \rightarrow E, AC \rightarrow B\}$$

(BC)+ = BC, is not a superkey of R. So the decomposition {ABC, BCDE} is not lossless.

Question 3 (2) - A Counterexample

Suppose R:
(a1, b1, c1, d1, e1)
(a2, b1, c1, d2, e2)

After decomposition:

- Over ABC: (a1, b1, c1), (a2, b1, c1).
- Over BCDE: (b1, c1, d1, e1), (b1, c1, d2, e2).

Do a NATURAL JOIN:
(a1, b1, c1, d1, e1)
(a1, b1, c1, d2, e2)
(a2, b1, c1, d2, e2)
(a2, b1, c1, d2, e2)

Spurious data were generated, so the given decomposition is not lossless.

Question 4

4.1 Is the Job Application relation schema a good solution? Explain your answer. If you think it is not a good design, discuss at least two potential problems.

Idea:

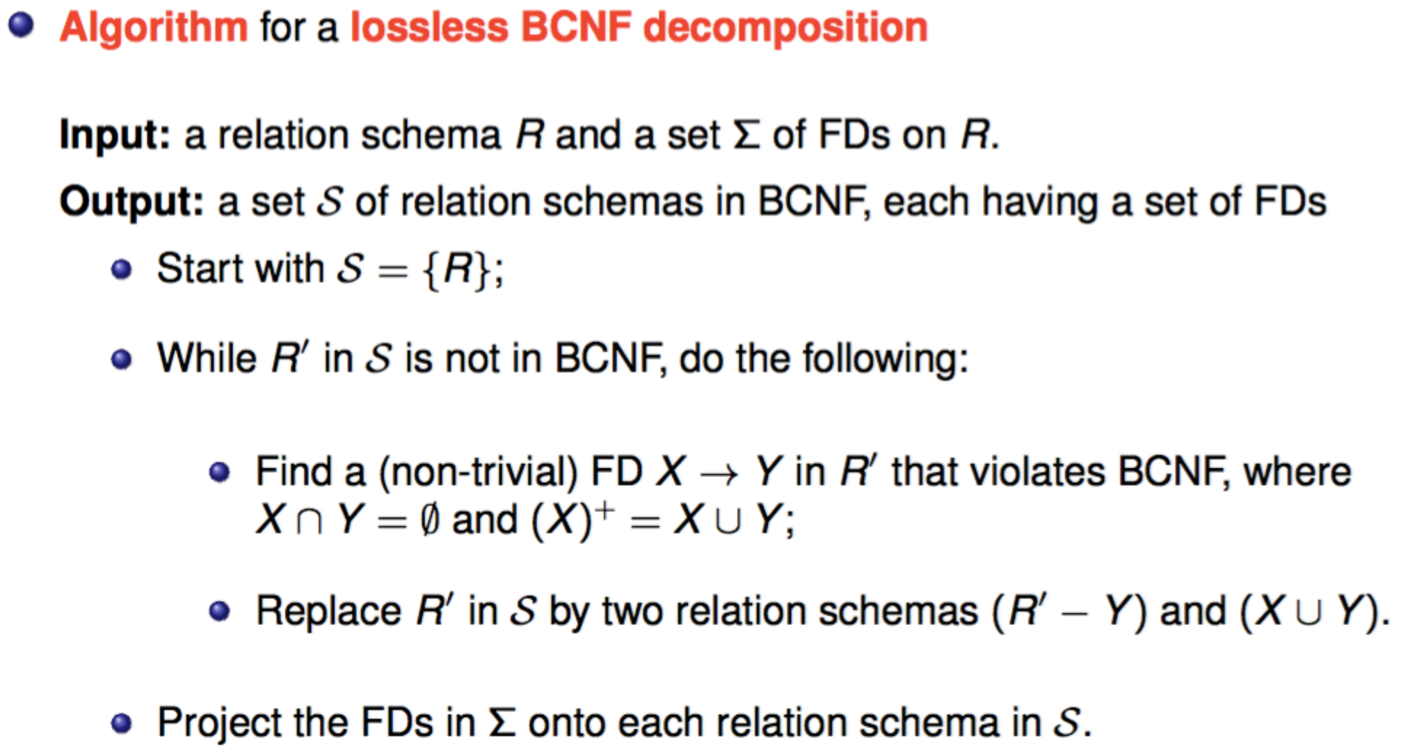
- Data inconsistency
- Data redundancy
- Update anomalies (Insertion, modification, deletion.)

Question 4

4.2 We want to redesign the Job Application relation schema. Can you identify a lossless BCNF decomposition for Job Application, which only preserves FD1, FD2, FD3, FD4, and FD6, i.e., FD5 cannot be preserved? You need to include the main steps used for identifying your BCNF decomposition and explain why your decomposition can preserve all functional dependencies except for FD5.

Question 4.2

BCNF: A relation schema R is in **BCNF** if whenever a non-trivial FD *X* -> *A* holds in R, then X is a **superkey** .



- **Algorithm** for a **lossless BCNF decomposition**

  Input: a relation schema R and a set Σ of FDs on R.

  Output: a set $\mathcal{S}$ of relation schemas in BCNF, each having a set of FDs

  - Start with $\mathcal{S} = \{R\}$;
  - While R' in $\mathcal{S}$ is not in BCNF, do the following:
    - Find a (non-trivial) FD $X \rightarrow Y$ in R' that violates BCNF, where $X \cap Y = \emptyset$ and $(X)^+ = X \cup Y$;
    - Replace R' in $\mathcal{S}$ by two relation schemas $(R' - Y)$ and $(X \cup Y)$.
  - Project the FDs in Σ onto each relation schema in $\mathcal{S}$.

Question 4.2

Decomposition Order:

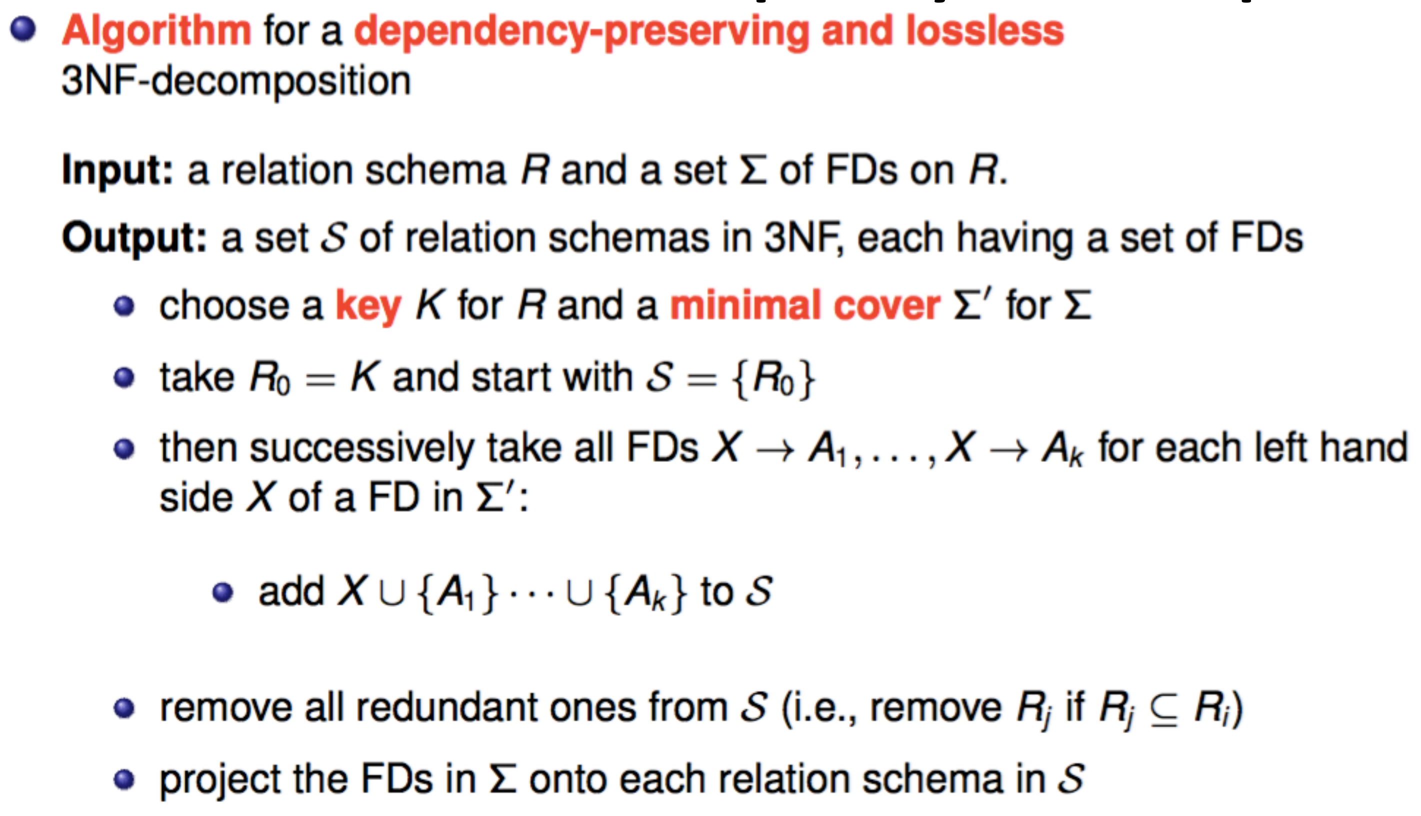
FD6, FD2, FD4,
or
FD6, FD1, FD4

Question 4.3

4.3 Is the Job Application relation schema in 3NF? If not, identify a lossless and de- pendency preserving 3NF decomposition for the current relation schema. You need to explain why the relation schema Job Application is or is not in 3NF, and include the steps used for identifying your 3NF decomposition.

Question 4.3

3NF: A relation schema R is in **3NF** if whenever a non-trivial FD *X* -> *A* holds in R, then *X* is a **superkey** or *A* is a **prime attribute** .



- **Algorithm** for a **dependency-preserving and lossless** 3NF-decomposition

  Input: a relation schema R and a set Σ of FDs on R.

  Output: a set $\mathcal{S}$ of relation schemas in 3NF, each having a set of FDs

  - choose a **key** K for R and a **minimal cover** Σ' for Σ
  - take $R_0 = K$ and start with $\mathcal{S} = \{R_0\}$
  - then successively take all FDs $X \rightarrow A_1, \ldots, X \rightarrow A_k$ for each left hand side X of a FD in Σ':
    - add $X \cup \{A_1\} \cdots \cup \{A_k\}$ to $\mathcal{S}$
  - remove all redundant ones from $\mathcal{S}$ (i.e., remove R_j if $R_j \subseteq R_i$)
  - project the FDs in Σ onto each relation schema in $\mathcal{S}$

Question 4.3

Choose a key K for R and a minimal cover E’:

- K = {Candidate_Email, Job_No}
- E’ = ?

Question 4.3 – Minimal Cover

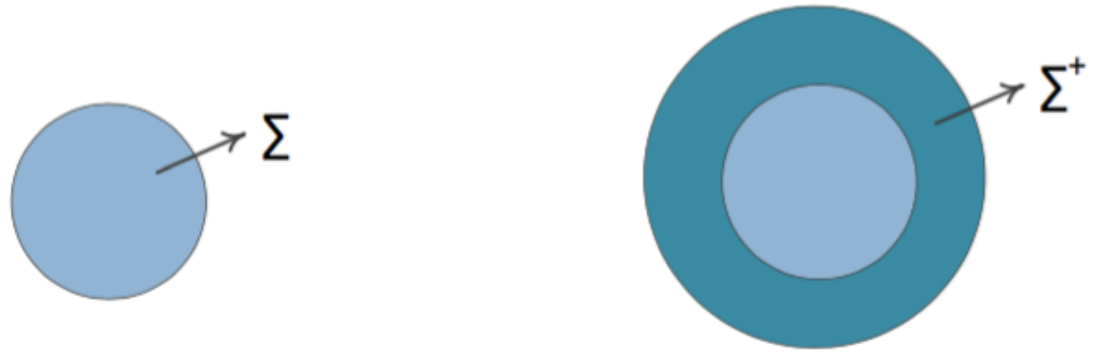
Algorithm[7]:

Input: a set Σ of FDs on R.
Output: a minimal cover Σ_m for Σ on R.

- **Step 1:** Start with $\Sigma_m = \Sigma$
- **Step 2:** Replace each FD $X \rightarrow \{A_1, \ldots, A_k\}$ in Σ_m with $X \rightarrow A_1, \ldots, X \rightarrow A_k$ (i.e., the right hand side is single attribute).
- **Step 3:** For each FD $X \rightarrow A$ in Σ_m, check each attribute B of X:
  - if we can replace $X \rightarrow A$ with $(X - B) \rightarrow A$ in Σ_m, and still remain equivalence of Σ_m and Σ, then we replace it.
- **Step 4:** For each remaining FD $X \rightarrow A$ in Σ_m:
  - if we can remove $X \rightarrow A$ and Σ_m is still equivalent to Σ, then we remove it.

Question 4.3 – Equivalence of sets of FDs

- We write Σ^+ for all functional dependencies that are implied by those in Σ.



- It is possible for $\Sigma_1^+ = \Sigma_2^+$ even if $\Sigma_1 \neq \Sigma_2$. When $\Sigma_1^+ = \Sigma_2^+$, we say that Σ_1 and Σ_2 are **equivalent**.

  Example: Let $\Sigma_1 = \{X \rightarrow Y, Y \rightarrow Z\}$ and $\Sigma_2 = \{X \rightarrow Y, Y \rightarrow Z, X \rightarrow Z\}$. In this case, we have $\Sigma_1 \neq \Sigma_2$ but $\Sigma_1^+ = \Sigma_2^+ = \{X \rightarrow Y, Y \rightarrow Z, X \rightarrow Z\}$. Hence, Σ_1 and Σ_2 are equivalent.

Question 4.4

Should we consider 3NF instead of BCNF when redesigning Job Application? Why or why not?

Idea: Difference between 3NF and BCNF, what problems can be solved by each normal form.

Notes

3NF ensure preservation of functional dependencies but not necessarily the absence of redundancy while BCNF eliminates redundant but not necessarily guarantee the preservation of functional dependencies. Therefore, we may want to consider 3NF instead of BCNF if it is more important to ensure that none of the data requirements corresponding to the functional dependencies are violated than to guarantee the absence of redundancy. Conversely, we may want to consider BCNF instead of 3NF if it is more important to guarantee the absence of redundancy than to ensure that none of the data requirements corresponding to the functional dependencies are violated.