Assignment 4 - 10 ECTS

Exercise (1)**.**

NGOs We once again consider a database that handles data for NGOs.

1. Given the following non-trivial functional dependency. What are the candidate keys for these three tables?
   - NGO: {Name} → {based_in, cause, director, phone, revenue} {Phone} → {name, based_in, cause, director, revenue} {director} → {name, based_in, cause, phone, revenue}
   - Supporter: {id} → {name, email, phone, city, ngo_name, birthday, volunteer, level}
   - Donations: {s_id, ngo_name, date, amount} → {activity}
2. Provide an example of a superkey that is not a candidate key based on the non-trivial functional dependency we provide above.

• • •

Solution. The candidate keys are just the left-hand sides of the functional dependencies, as they fully determine the respective relations, and are minimal. So for NGO $\{nae\}$, $\{phone\}$ and $\{director\}$ are candidate keys. For supporter $\{id\}$ is the candidate key. For donations $\{s\_id, ngo\_name, date, amount\}$ is the candidate key. We would need additional functional dependencies from the miniworld, to determine whether we any other candidate keys. An example of a superkey would be $\{name, phone\}$, which is a key, but it is not minimal.

Exercise (2)**.**

University Consider the relation R, which involves schedules of courses and sections at a university (shorthand notation written in parenthesis): R = {Course Number (C), Section Number (SN), Offering Department (D), Course Type (CT), Credit Hours (CH), Instructor ID (I), Semester (S), Year (Y), Day and Hours (DH), Room Number (R), Number of Students Enrolled (NS)}. Assume that the following set of functional dependencies F holds on R: F = { {C} → {D, CT}, {C, SN, S, Y} → {R, DH, NS}, {R, DH, S, Y} → {C, SN}, {I} → {D}, {CT} → {CH}, {D, CT} → {I} }

1. Find the 2 candidate keys of R based on F. Please give the steps and argument to find the candidate keys.

• • •

Solution. The candidate keys are $\{C, SN, S, Y\}$ and $\{R, DH, S, Y\}$. To see why, we computer their closures. We will be using transitivity a bunch of times.

$\{R, DH, S, Y\}$

$$\begin{aligned}\{R, DH, S, Y\}^+ &\sim \{R, DH, S, Y, C, SN\}^+ \\ &\sim \{R, DH, S, Y, C, SN, NS\}^+ \\ &\sim \{R, DH, S, Y, C, SN, NS, D, CT\}^+ \\ &\sim \{R, DH, S, Y, C, SN, NS, D, CT, I\}^+ \\ &\sim \{R, DH, S, Y, C, SN, NS, D, CT, I, CH\}^+ = R.\end{aligned}$$

Where I have the used the functional dependencies. So it is indeed a key, and a candidate key since it is minimal. We cant remove any elements.

$\{C, SN, S, Y\}$

$$\{C, SN, S, Y\}^+ \sim \{C, SN, S, Y, R, DH, NS\}^+ = R.$$

Using the functional dependencies, and reusing the functional dependencies. This key is minimal as well, so it is candidate key.

☕

Exercise (3)**.**

Miniworld to Functional Dependencies Consider the following relation: Order(Product_ID, Product_Name, Customer_ID, Customer_Name, Date, Item_price, Count, Total)

- Product ID: Each product is assigned a unique identifier, known as the Product ID.
- Product Name: Each product is given a distinct name, corresponding to its Product ID.
- Customer ID: Each customer is assigned a unique identifier, referred to as the Customer ID.
- Customer Name: Each customer is associated with a unique name, corresponding to their Customer ID.
- Date: The date of purchase made by the customer.
- Item Price: The cost of each individual product.
- Count: The total quantity of items purchased by a customer on a given day.
- Total: The aggregate expenditure made by a customer on a particular day.

OBS: If the same customer send multiple orders on the same day they are combined. Therefore, we only have one order per customer per day. Determine all non-trivial functional dependencies of this relation and provide your reasoning.

• • •

Solution. We will be arguing based on the miniworld. Since the product id determines the product, we should have that

$$\{product\_id\} \to \{product\_name, item\_price\}.$$

Since the product name is unique to its id, we should also have,

$$\{product\_name\} \to \{product\_id, item\_price\}.$$

The same logic holds for the customer id and the customer name,

$$\{customer\_id\} \to \{customer\_name, item\_price\}$$
$$\{customer\_name\} \to \{customer\_id, item\_price\}.$$

Since count and total, refer to the order made by a customer on a given day, it should hold that given date and either customer id or name, we can infer count and total,

$$\{customer\_name, date\} \to \{Count, Total\}$$
$$\{customer\_id, date\} \to \{Count, Total\}.$$

☕

Exercise (4)**.**

Consider the relation $R = \{A, B, C, D, E, F, G, H, I, J\}$ and the set of functional dependencies:

$$F = \begin{cases} \{A, B\} \to \{C\}, \\ \{A\} \to \{D, E\}, \\ \{B\} \to \{F\}, \\ \{C\} \to \{B\}, \\ \{F\} \to \{G, H\}, \\ \{D\} \to \{I, J\}. \end{cases}$$

a) Decompose R to 2NF, and then into 3NF. Please provide your steps.

b) Is the 3NF decomposition dependency preserving?

c) Is the 3NF decomposition non-additive?

• • •

Solution.

a)

We can start by noticing that A, B is candidate key for the relation. In order to have 2NF, no other attributes can be partially dependent on this key. This is violated since,

$$\{B\} \to \{F, G, H\}$$
$$\{A\} \to \{D, E, I, J\}$$

We can decompose our relation according to these partial dependencies, to get the following new relations,

$$R_1 = \{B, F, G, H\} \quad \{B\} \to \{F\} \to \{G, H\}$$
$$R_2 = \{A, D, E, I, J\} \quad \{A\} \to \{D, E\}, \{D\} \to \{I, J\}$$
$$R_3 = \{A, B, C\} \quad \{A, B\} \to \{C\}$$
$$.$$

Where I have listed the functional dependencies that hold for each of these relations. Now clearly, these relations are 2NF. R_1 and R_2 have single attribute candidate keys, which implies 2NF. And $\{C\} \in R_3$ depends on both A and B. To get to 3NF, we have to get rid of the transitive dependencies, therefore we just split the relations according to these,

$$R_1 = \{B, F\} \quad \{B\} \to \{F\}$$
$$R_2 = \{F, G, H\} \quad \{F\} \to \{G, H\}$$
$$R_3 = \{A, D, E\} \quad \{A\} \to \{D, E\}$$
$$R_4 = \{D, I, J\} \quad \{D\} \to \{I, J\}$$
$$R_5 = \{A, B, C\} \quad \{A, B\} \to \{C\}$$
$$.$$

This decomposition is dependency preserving since,

$$\bigcup_i F(R_i) = F.$$

I.e. if we combine the functional dependencies of the decomposed relations we get the functional dependencies of our original relation. For the decomposition to be non-additive, we have to check that differences are functionally dependent on intersections. In other words, for any two relations R_1 and R_2 it should hold that,

$$(R_1 \cap R_2) \to (R_1 - R_2) \in F \quad \text{or}$$
$$(R_1 \cap R_2) \to (R_2 - R_1) \in F.$$

It sufficient to check the relations that share attributes, we will go through them below,

$$R_3 \cap R_5 = \{A\} \to R_3 - R_5 = \{D, E\}$$
$$R_1 \cap R_2 = \{B\} \to R_2 - R_1 = \{G, H\}$$
$$R_3 \cap R_4 = \{D\} \to R_4 - R_3 = \{I, J\}$$
$$R_1 \cap R_5 = \{B\} \to R_1 - R_5 = \{F\}$$
$$.$$

So it is indeed loss-less

Exercise (5)**.**

Consider the relation $R = \{A,B,C,D,E\}$ and $F = \{A \to B, A \to C, B \to CD\}$. Decompose this into 3NF using the synthesis algorithm.

Lets start by noting that $\{A,E\}$ is a candidate key for the relation. The first step in the algorithm is to find a minimal cover. We do this by splitting up the right-hand side and eliminating redundancies,

$$\begin{aligned} F &= \{A \to B, A \to C, B \to CD\} \\ &\sim \{A \to B, A \to C, B \to C, B \to D\} \\ &\sim \{A \to B, B \to C, B \to D\} = G. \end{aligned}$$

We then decompose our relation according these FDs,

$$D_1 = (A,B) \quad \text{and} \quad D_2 = (B,C,D)\,.$$

Since none of these relations contain our key, we have to add an additional one,

$$D_3 = (A,E)\,.$$

Now, this is indeed a 3NF decomposition. There are no transitive dependencies, and no partial dependencies.