Tutorial: Functional Dependencies

Your company, Apasaja Private Limited, is commissioned by Toko Kopi Luwak to design relational schema for the management of their coffee beans, drinks, and cafes.

A coffee bean is fully identified by its unique brand name or a combination of its cultivar and region (since the same cultivar can be grown in different region). For instance, we may have a coffee bean named "The Waterfall" which comes from a Tabi cultivar grown in Colombia.

A drink can be made utilizing a particular coffee bean. The name of the drink is only unique for a particular coffee bean. This means, we can have an "Espresso" made with "The Waterfall" or "La Bella" (which is a Pacamara cultivar grown in Guatemala). The price of the drink is also recorded.

A branch identified by its branch name may then sell the drink. A drink may be sold by zero or more branches. A branch may sell zero or more drinks. Additionally, the address of the branch is also recorded. Lastly, for each drink sold by a branch, we record the quantity sold to see which branch is the most profitable.

We are only given an abstract schema for this application as follows.

$$R = \{A, B, C, D, E, F, G, H\}$$

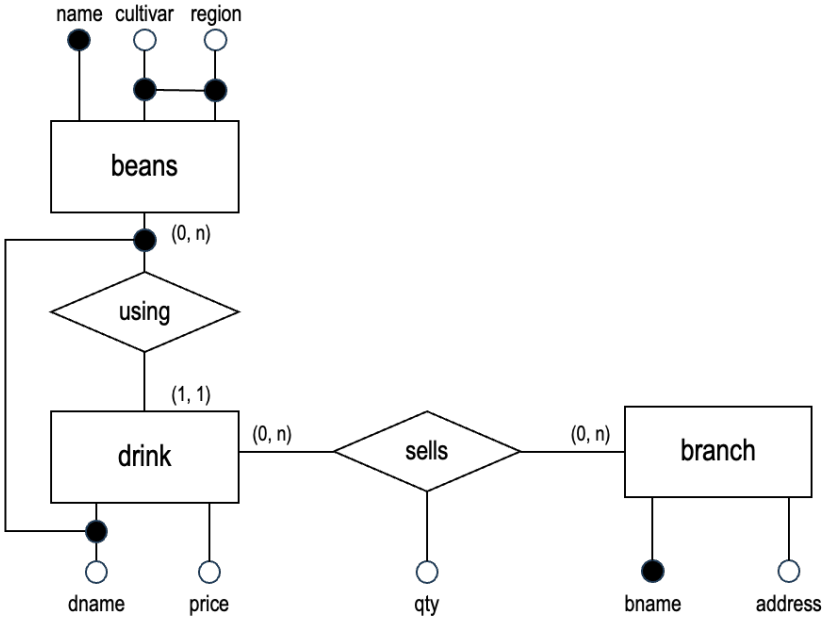
$$\Sigma = \{\ \{A\} \to \{C, E\},\ \ \{A, B\} \to \{D\},\ \ \{F\} \to \{H\},\ \ \{C, E\} \to \{A\},\ \ \{B, C, E\} \to \{D\},\ \ \{A, B, F\} \to \{D, G\},\ \ \{B, C, E, F\} \to \{G\}\ \}$$

Questions

Not all questions will be discussed during tutorial. You are expected to attempt them before coming to the tutorial. You may be randomly called to present your answer during tutorial. You are encouraged to discuss them on Canvas Discussion.

1. **Functional Dependencies**.
   (a) From the functional dependencies in Σ and the text description of the application, can you figure out the mapping of the attributes and the letters?

   > **Comments:**
   > While not necessary, it might be easier to draw an entity-relationship diagram based on the text description of the application and figure out the mapping from there.



Alternatively, once you have obtained the entity-relationship diagram, you may translate the diagram into schema. Then, the functional dependencies from each table can be inferred from the `UNIQUE` and `PRIMARY KEY` constraints in the schema. Looking at it another way, the functional dependencies required by the application (i.e., those given by Σ) have been enforced by decomposing R into smaller tables and applying `UNIQUE` or `PRIMARY KEY` constraints to the new tables. This is the main idea behind database normalization.

Attribute	Character
`name`	A
`dname` (drink name)	B
`cultivar`	C
`price`	D

Attribute	Character
`region`	E
`bname` (branch name)	F
`qty` (quantity)	G
`address`	H

In general, functional dependencies may come from multiple sources including entity-relationship diagram, schema, "common sense", or text description of the problem. We list *some* direct sources of functional dependencies below. There may be others as the list below is not exhaustive.

- Entity-Relationship Diagram
  - Key attributes functionally determine the rest of the attributes.
  - Entity set participating with a (0,1) or (1,1) constraint functionally determines the attributes of the relationship set and the attributes of the other participating entity set.
    $\Rightarrow$ An exception is the (1,1) constraint for weak entity set.
  - If all of the entity sets are participating with (0,n) or (1,n) constraint, their combined key attributes determine the attributes of the relationship set.
- Schema
  - `PRIMARY KEY` and `UNIQUE` attributes functionally determine the rest of the attributes (as we assume we are not dealing with `NULL`).

(b) Compute the attribute closures of the subset of attributes of R with Σ in order to find the candidate keys of R with Σ.

Comments:
Since B and F do not appear on the right-hand side of any functional dependencies, they must appear in all keys. We compute the attribute closure for all attributes starting from the smallest set containing B and F.

- Sets of 2 attributes.
  - $\{B, F\}^+ = \{B, F, H\}$ (*not a key*)
- Sets of 3 attributes superset of $\{B, F\}$.
  - $\{\mathbf{A}, B, F\}^+ = \{A, B, C, D, E, F, G, H\}$ (**candidate key**)
  - $\{B, \mathbf{C}, F\}^+ = \{B, C, F, H\}$ (*not a key*)
  - $\{B, \mathbf{D}, F\}^+ = \{B, D, F, H\}$ (*not a key*)
  - $\{B, \mathbf{E}, F\}^+ = \{B, E, F, H\}$ (*not a key*)
  - $\{B, F, \mathbf{G}\}^+ = \{B, F, H, G\}$ (*not a key*)
  - $\{B, F, \mathbf{H}\}^+ = \{B, F, H\}$ (*not a key*)
- Sets of 4 attributes superset of $\{B, F\}$ but not superset of $\{A, B, F\}$.
  - $\{B, \mathbf{C}, \mathbf{D}, F\}^+ = \{B, C, D, F, H\}$ (*not a key*)
  - $\{B, \mathbf{C}, \mathbf{E}, F\}^+ = \{A, B, C, D, E, F, G, H\}$ (**candidate key**)
  - $\{B, \mathbf{C}, F, \mathbf{G}\}^+ = \{B, C, F, G, H\}$ (*not a key*)
  - $\{B, \mathbf{C}, F, \mathbf{H}\}^+ = \{B, C, F, H\}$ (*not a key*)
  - $\{B, \mathbf{D}, \mathbf{E}, F\}^+ = \{B, D, E, F, H\}$ (*not a key*)
  - $\{B, \mathbf{D}, F, \mathbf{G}\}^+ = \{B, D, F, G, H\}$ (*not a key*)
  - $\{B, \mathbf{D}, F, \mathbf{H}\}^+ = \{B, D, F, H\}$ (*not a key*)
  - $\{B, \mathbf{E}, F, \mathbf{G}\}^+ = \{B, E, F, G, H\}$ (*not a key*)
  - $\{B, \mathbf{E}, F, \mathbf{H}\}^+ = \{B, E, F, H\}$ (*not a key*)
  - $\{B, F, \mathbf{G}, \mathbf{H}\}^+ = \{B, F, G, H\}$ (*not a key*)
- Sets of 5 attributes superset of $\{B, F\}$ but not superset of $\{A, B, F\}$ or $\{B, C, E, F\}$.
  - $\{B, \mathbf{C}, F, \mathbf{G}, \mathbf{H}\}^+ = \{B, C, F, G, H\}$ (*not a key*)
  - $\{B, \mathbf{E}, F, \mathbf{G}, \mathbf{H}\}^+ = \{B, E, F, G, H\}$ (*not a key*)
  - *...other sets, also not a key.*
- Convince yourself that no set with 6 or 7 attributes that is a superset of $\{B, F\}$ but not superset of $\{A, B, F\}$ or $\{B, C, E, F\}$, is a candidate key.

Hence, keys = $\{\mathbf{A}, \mathbf{B}, \mathbf{F}\}$ and $\{\mathbf{B}, \mathbf{C}, \mathbf{E}, \mathbf{F}\}$.

(c) Find the prime attributes of R with Σ.

Comments:
Since the keys are $\{A, B, F\}$ and $\{B, C, E, F\}$, the prime attributes are $\{A, B, F\} \cup \{B, C, E, F\} = \{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{E}, \mathbf{F}\}$.

2. **Minimal Cover**.

(a) Compute a minimal cover of R with Σ.

Comments:

We start from Σ.

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

Step 1: Simplify the right-hand side.

$\{A\} \to \{C\}$ (*split*)
$\{A\} \to \{E\}$ (*split*)
$\{A, B\} \to \{D\}$
$\{F\} \to \{H\}$
$\{C, E\} \to \{A\}$
$\{B, C, E\} \to \{D\}$
$\{A, B, F\} \to \{D\}$ (*split*)
$\{A, B, F\} \to \{G\}$ (*split*)
$\{B, C, E, F\} \to \{G\}$

Step 2: Simplify the left-hand side.

$\{A\} \to \{C\}$
$\{A\} \to \{E\}$
$\{A, B\} \to \{D\}$
$\{F\} \to \{H\}$
$\{C, E\} \to \{A\}$
$\{B, C, E\} \to \{D\}$
$\{A, B, \cancel{F}\} \to \{D\}$ ($\{A, B\} \to \{D\}$)
$\{A, B, F\} \to \{G\}$
$\{B, C, E, F\} \to \{G\}$

Step 3: Simplify the set.

$\{A\} \to \{C\}$
$\{A\} \to \{E\}$
~~$\{A, B\} \to \{D\}$~~
$\{F\} \to \{H\}$
$\{C, E\} \to \{A\}$
$\{B, C, E\} \to \{D\}$
~~$\{A, B\} \to \{D\}$~~
~~$\{A, B, F\} \to \{G\}$~~
$\{B, C, E, F\} \to \{G\}$

One possible minimal cover is shown below.

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

There can be other minimal covers such as the following.

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

Note that our algorithm may not be able to obtain all possible minimal covers. This is because we work by *eliminating redundancy*. Hence, we cannot add new functional dependencies that were not originally in Σ. If the algorithm starts from the closure of Σ (i.e., Σ^+), then the algorithm can compute all possible minimal covers.

(b) Compute a canonical cover of R with Σ.

Comments:

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

There are alternative answers but they are left as an exercise.

Comments:

It is important to note that the attribute closure algorithm is the most important algorithm for functional dependencies and normalization. To illustrate its importance, let us give it a name: `AttrClose(attrs, sigma)`. This algorithm computes the attribute closure of the set of attributes `attrs` with respect to the set of functional dependencies `sigma`.

We can now solve the rest of the problems with this algorithm.

Q1b: For each subset of attributes X still considered, we do the following.

(i) Compute $\varphi := \texttt{AttrClose}(X, \Sigma)$.

(ii) Check if $\varphi = R$.

Q2a: We use the algorithm in different ways.

Step 2: Consider $X \to \{A\}$. To remove attribute $B \in X$, we need to show that $(X - \{B\}) \to \{A\}$ can be derived from Σ. This is done by computing $(X - \{B\})^+$ and checking if it contains A. In other words, we do the following.

(i) Compute $\varphi := \texttt{AttrClose}((X - \{B\}), \Sigma)$.

(ii) Check if $A \in \varphi$.

Step 3: Consider $X \to \{A\}$ in Σ. To remove $X \to \{A\}$ from Σ, we need to show that $X \to \{A\}$ can be derived from Σ without using $X \to \{A\}$. This is done by computing X^+ with respect to $\Sigma - \{X \to \{A\}\}$ and checking if it contains A. In other words, we do the following.

(i) Compute $\varphi := \texttt{AttrClose}(X, \Sigma - \{X \to \{A\}\})$.

(ii) Check if $A \in \varphi$.

When answering questions about functional dependencies and normalizations, it is advisable to always compute (i) the candidate keys and (ii) the minimal cover. These will simplify the rest of the computations.

References

[1] S. Bressan and B. Catania. *Introduction to Database Systems*. McGraw-Hill Education, 2006. ISBN: 9780071246507.

[2] Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. *Database Systems: The Complete Book*. 2nd ed. Prentice Hall Press, 2008. ISBN: 9780131873254.

[3] Raghu Ramakrishnan and Johannes Gehrke. *Database Management Systems*. 2nd. USA: McGraw-Hill, Inc., 2000. ISBN: 0072440422.