Database Programming and Management

Tutorial 8: Normalization

Biswadeep Sen
School of Computing
National University of Singapore
biswadeep@u.nus.edu

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

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

1.(a) Is R with Σ in 3NF?

A relation R is in **3NF** if, for every functional dependency $X \rightarrow Y$ in Σ:

1. The dependency is **trivial** (i.e, $Y \subseteq X$),

   OR

2. X is a **superkey** for R,

   OR

3. Every attribute in Y is **prime** (i.e., part of some candidate key).

📌 *NOTE:* Satisfying any one of the conditions will suffice and hence **OR**.

1.(a) Is R with Σ in 3NF?

Consider $\{A, B\} \rightarrow \{D\}$.

1. The dependency is **trivial** (i.e, $Y \subseteq X$),

   OR

2. X is a **superkey** for R,

   OR

3. Every attribute in Y is **prime**

📌 *NOTE:* To prove something is NOT in 3NF, all 3 conditions must be False.

1.(a) Is R with Σ in 3NF?

Consider $\{A, B\} \rightarrow \{D\}$.

1. The dependency is **trivial** (i.e, Y ⊆ X),

   OR

2. X is a **superkey** for R,

   OR

3. Every attribute in Y is **prime**

1.(a) Is R with Σ in 3NF?

Consider $\{A, B\} \rightarrow \{D\}$.

- It is non-trivial (i.e., $\{D\} \not\subseteq \{A, B\}$).

1. The dependency is **trivial** (i.e, Y ⊆ X),

   OR

2. X is a **superkey** for R,

   OR

3. Every attribute in Y is **prime**

1.(a) Is R with Σ in 3NF?

1. The dependency is **trivial** (i.e, Y ⊆ X),

 OR

2. X is a **superkey** for R,

 OR

3. Every attribute in Y is **prime**

Consider $\{A, B\} \rightarrow \{D\}$.

- It is non-trivial (i.e., $\{D\} \not\subseteq \{A, B\}$).
- $\{A, B\}$ is not a key (i.e., $\{A, B\}^+ = \{A, B, C, D, E\} \subset R$).

  $\{A, B\}$ is *also* not a superset of a key (keys are $\{A, B, F\}$ and $\{B, C, E, F\}$).

  $\Rightarrow$ This is a simpler way to check superkey if we have computed keys.

1.(a) Is R with Σ in 3NF?

1. The dependency is **trivial** (i.e, Y ⊆ X),

   OR

2. X is a **superkey** for R,

   OR

3. Every attribute in Y is **prime**

Consider $\{A, B\} \rightarrow \{D\}$.

- It is non-trivial (i.e., $\{D\} \not\subseteq \{A, B\}$).
- $\{A, B\}$ is not a key (i.e., $\{A, B\}^+ = \{A, B, C, D, E\} \subset R$).

  $\{A, B\}$ is *also* not a superset of a key (keys are $\{A, B, F\}$ and $\{B, C, E, F\}$).

  $\Rightarrow$ This is a simpler way to check superkey if we have computed keys.
- D is not a prime attribute.

  Prime attributes are $\{A, B, C, E, F\}$.

NO! NOT in 3NF!

1.(b) Is R with Σ in BCNF?

A relation R is in **BCNF** if, for every functional dependency $X \rightarrow Y$ in Σ:

1. The dependency is **trivial** (i.e, $Y \subseteq X$),

   OR

2. X is a **superkey** for R,

📌 *NOTE:*

- Every BCNF relation is in 3NF, but not every 3NF relation is in BCNF
- BCNF is stricter!

1.(b) Is R with Σ in BCNF?

From Question 1a, we know that R with Σ is not in 3NF. Therefore, it cannot be in BCNF. However, let us verify this from the definition of BCNF. Obviously, we can consider $\{A, B\} \rightarrow \{D\}$, but let us consider a different functional dependency. Consider $\{A\} \rightarrow \{C\}$.

NO! NOT in BCNF!

1.(b) Is R with Σ in BCNF?

From Question 1a, we know that R with Σ is not in 3NF. Therefore, it cannot be in BCNF. However, let us verify this from the definition of BCNF. Obviously, we can consider $\{A, B\} \rightarrow \{D\}$, but let us consider a different functional dependency. Consider $\{A\} \rightarrow \{C\}$.

- It is non-trivial (i.e., $\{C\} \not\subseteq \{A\}$).
- $\{A\}$ is not a superkey (i.e., $\{A\}^+ = \{A, C, E\} \subset R$).

  $\{A\}$ is *also* not a superset of a key (keys are $\{A, B, F\}$ and $\{B, C, E, F\}$).

  $\Rightarrow$ This is a simpler way to check superkey if we have computed keys.

NO! NOT in BCNF!

2.(a) Decompose R with Σ into a lossless-join 3NF.

Algorithm : 3NF Synthesis (Bernstein Algorithm)

When a relation is not in 3NF, we can *synthesize* a schema in 3NF from a *canonical cover* of the set of functional dependencies.

- For each functional dependency $X \rightarrow Y$ in the minimal cover, create a relation

  $$R_i = X \cup Y$$

  Unless it already exists or is *subsumed* by another relation
- If none of the created relations contain one of the keys, pick any candidate key and create a relation with that candidate key.

2.(a) Decompose R with Σ into a lossless-join 3NF.

We can start from a canonical cover directly.

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

Algorithm : 3NF Synthesis (Bernstein Algorithm)

When a relation is not in 3NF, we can *synthesize* a schema in 3NF from a *canonical cover* of the set of functional dependencies.

- For each functional dependency $X \rightarrow Y$ in the minimal cover, create a relation

$$R_i = X \cup Y$$

  Unless it already exists or is *subsumed* by another relation

- If none of the created relations contain one of the keys, pick any candidate key and create a relation with that candidate key.

2.(a) Decompose R with Σ into a lossless-join 3NF.

We can start from a canonical cover directly.

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

For each functional dependency, we synthesize a fragment.

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

2.(a) Decompose R with Σ into a lossless-join 3NF.

We can start from a canonical cover directly.

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

For each functional dependency, we synthesize a fragment.

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

If there is any fragments that can be *subsumed*, we remove them from the result.

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

2.(a) Decompose R with Σ into a lossless-join 3NF.

We can start from a canonical cover directly.

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

For each functional dependency, we synthesize a fragment.

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

If there is any fragments that can be *subsumed*, we remove them from the result.

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

If no candidate keys is present as a subset of any fragment, we add. Luckily, the key $\{B, C, E, F\}$ is a subset of $\{B, C, E, F, G\}$. So, we do not have to add another relation.

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

If no key is present

We know the keys of R are *{A,B,F}* and *{B,C,E,F}*

Suppose after subsumption we ended up with this:

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

Add any of the keys! That’s it

NOTE:

- If the questions asks you to just make fragments - Make fragments using the **3NF synthesis algorithm** and you can stop there.
- Skip **FD projections** and **BCNF checks** unless the question **explicitly asks** or you need to verify extras.

Projection steps

5 steps to compute projection of R with Σ onto X.

1. Find all subset X' of attributes of X.
2. For each subset X', compute the closure (i.e., $\varphi_1 := \texttt{AttrClose}(X', \Sigma)$).
3. Keep only the relevant attributes (i.e., $\varphi_2 := \varphi_1 \cap X$).
4. Remove attributes that does not contribute new information (i.e., $\varphi_3 := \varphi_2 - X'$).
5. If φ_3 is not empty, form a functional dependency $X' \rightarrow \varphi_3$.

📌**Question:**
Calculate the projection R with Σ onto X = {A, C, E}

Calculating Projections

Do this for all possible subsets

1 = choose subset $X' \subseteq X$

2 = $\varphi_1 := \mathrm{AttrClose}(X', \Sigma)$

3 = $\varphi_2 := \varphi_1 \cap X$

4 = $\varphi_3 := \varphi_2 - X'$

5 = if $\varphi_3 \neq \varnothing$, emit $X' \rightarrow \varphi_3$

Onto $X = \{A, C, E\}$ (call this Σ_1)

Step 1 X'	**Step 2 φ_1 (closure)**	**Step 3 φ_2**	**Step 4 φ_3**	**Step 5 (FD)**
{A}	{A,C,E} (by A→CE)	{A,C,E}	{C,E}	**A→CE**
{C,E}	{A,C,E} (by CE→A)	{A,C,E}	{A}	**CE→A**
{A,C}	{A,C,E}	{A,C,E}	{E}	AC→E (*implied by A→CE*)
{A,E}	{A,C,E}	{A,C,E}	{C}	AE→C (*implied by A→CE*)

All other X' give $\varphi_3 = \varnothing$. Keep the bold FDs for a minimal projection: **Σ_1 = {A→CE, CE→A}**.

Calculating Projections

Exhaustive check (all 8 subsets of X)

Step 1 X'	**Step 2** $\varphi_1 = (X')^+$	**Step 3** $\varphi_2 = \varphi_1 \cap X$	**Step 4** $\varphi_3 = \varphi_2 \setminus X'$	**Step 5 (FD if any)**
∅	∅	∅	∅	—
{E}	{E}	{E}	∅	—
{C}	{C}	{C}	∅	—
{A}	{A,C,E} (by $A \to CE$)	{A,C,E}	{C,E}	**A → CE**
{C,E}	{A,C,E} (by $CE \to A$)	{A,C,E}	{A}	**CE → A**
{A,E}	{A,C,E}	{A,C,E}	{C}	AE → C *(implied by $A \to CE$)*
{A,C}	{A,C,E}	{A,C,E}	{E}	AC → E *(implied by $A \to CE$)*
{A,C,E}	{A,C,E}	{A,C,E}	∅	—

1 = choose subset $X' \subseteq X$

2 = $\varphi_1 := \mathrm{AttrClose}(X', \Sigma)$

3 = $\varphi_2 := \varphi_1 \cap X$

4 = $\varphi_3 := \varphi_2 - X'$

5 = if $\varphi_3 \neq \emptyset$, emit $X' \to \varphi_3$

Calculating Projections

Do this for all possible subsets

1 = choose subset $X' \subseteq X$

2 = $\varphi_1 := \mathrm{AttrClose}(X', \Sigma)$

3 = $\varphi_2 := \varphi_1 \cap X$

4 = $\varphi_3 := \varphi_2 - X'$

5 = if $\varphi_3 \neq \varnothing$, emit $X' \to \varphi_3$

Onto $X = \{F, H\}$ (Σ_2)

Step 1 X'	**Step 2** φ_1	**Step 3** φ_2	**Step 4** φ_3	**Step 5**
{F}	{F,H} (by F→H)	{F,H}	{H}	**F→H**

Other subsets yield $\varphi_3 = \varnothing$. So **Σ_2 = {F→H}**.

📌**Early Exit Rule:**
Let S = AttrClose(X′, Σ). If φ_2 = X′, stop for this X′ (no FD).

Calculating Projections

Do this for all possible subsets

1 = choose subset $X' \subseteq X$

2 = $\varphi_1 := \mathrm{AttrClose}(X', \Sigma)$

3 = $\varphi_2 := \varphi_1 \cap X$

4 = $\varphi_3 := \varphi_2 - X'$

5 = if $\varphi_3 \neq \varnothing$, emit $X' \to \varphi_3$

Onto $X = \{B, C, D, E\}$ (Σ_3)

Step 1 X'	**Step 2 φ_1**	**Step 3 φ_2**	**Step 4 φ_3**	**Step 5**
{B,C,E}	{B,C,E,D} (by **BCE→D**)	{B,C,E,D}	{D}	**BCE→D**

All other subsets inside $\{B, C, D, E\}$ do not gain D (or add nothing beyond themselves), so $\varphi_3 = \varnothing$.

Thus **Σ_3 = {BCE→D}**.

Calculating Projections

Do this for all possible subsets

1 = choose subset $X' \subseteq X$

2 = $\varphi_1 := \mathrm{AttrClose}(X', \Sigma)$

3 = $\varphi_2 := \varphi_1 \cap X$

4 = $\varphi_3 := \varphi_2 - X'$

5 = if $\varphi_3 \neq \varnothing$, emit $X' \to \varphi_3$

Onto $X = \{B, C, E, F, G\}$ **(Σ_4)**

Step 1 X'	**Step 2** φ_1	**Step 3** φ_2	**Step 4** φ_3	**Step 5**
{B,C,E,F}	{B,C,E,**F**,G} (by **BCEF→G**)	{B,C,E,F,G}	{G}	**BCEF→G**

Others give $\varphi_3 = \varnothing$. Hence **Σ_4 = {BCEF→G}**.

Final Projections

Do this for all possible subsets

1 = choose subset $X' \subseteq X$

2 = $\varphi_1 := \mathrm{AttrClose}(X', \Sigma)$

3 = $\varphi_2 := \varphi_1 \cap X$

4 = $\varphi_3 := \varphi_2 - X'$

5 = if $\varphi_3 \neq \varnothing$, emit $X' \rightarrow \varphi_3$

- $\{A, C, E\}$ with $\Sigma_1 = \{ \{A\} \rightarrow \{C\}, \ \{A\} \rightarrow \{E\}, \{C, E\} \rightarrow \{A\} \}$
- $\{F, H\}$ with $\Sigma_2 = \{ \{F\} \rightarrow \{H\} \}$
- $\{B, C, D, E\}$ with $\Sigma_3 = \{ \{B, C, E\} \rightarrow \{D\} \}$
- $\{B, C, E, F, G\}$ with $\Sigma_4 = \{ \{B, C, E, F\} \rightarrow \{G\} \}$

Mistake : "Keep only FDs that use attributes from X"

R(A, B, C), Σ = { A→B, B→C }, project onto X = {A, C}.

Naive keeps none (since both FDs mention B).

Correct (by the algorithm):

- For X′ = {A}: A^+ = {A, B, C} ⇒ (A^+ ∩ X) − {A} = {C} ⇒ **A→C**

  Projection is **{ A→C }**.

  The naive method **misses** A→C (transitive via B).

📌 *NOTE:*
Using shortcuts can lead to error in **MANY different ways!**
This is not the only one!

Projection – Caution Points ⚠️

- Projection is **error-prone** → don't skip steps!
- Shortcuts can omit valid **derived FDs**.
- Always follow all **5 steps** for correctness ✅ .
- Tedious ⏳, but ensures **accurate results**.
- ❌ Shortcuts **don't guarantee** correctness.

(b) Is the result dependency preserving?

Definition

- Let R be a relation schema and Σ a set of functional dependencies (FDs).
- Decompose R into $\Delta = \{R_1, \ldots, R_n\}$.
- For each piece R_i, let $\Sigma|_{R_i}$ be the projection of Σ onto R_i.
- The decomposition is *dependency-preserving* iff

$$\Sigma^+ = \left(\Sigma|_{R_1} \cup \cdots \cup \Sigma|_{R_n}\right)^+.$$

Note (how to check)

- For every FD $X \to Y$ in Σ, compute X^+ using $\Sigma|_{R_1} \cup \cdots \cup \Sigma|_{R_n}$.
- If $Y \subseteq X^+$ for all FDs in Σ, the decomposition is dependency-preserving.

(b) Is the result dependency preserving?

Yes. This is guaranteed by the algorithm. 3NF synthesis algorithm produces lossless-join dependency-preserving decomposition in 3NF.

(c) Is the result in BCNF?

(c) Is the result in BCNF?

The 3NF synthesis only guarantees that the result is in 3NF, it may not be in BCNF.

(c) Is the result in BCNF?

The 3NF synthesis only guarantees that the result is in 3NF, it may not be in BCNF.

- $\{A, C, E\}$ with $\Sigma_1 = \{ \{A\} \rightarrow \{C\},\ \{A\} \rightarrow \{E\}, \{C, E\} \rightarrow \{A\} \}$

📌 *NOTE:*
All non-trivial FDs here (A→C, A→E, CE→A, and A→CE) have LHS ∈ {A, CE} (keys) ⇒ BCNF

(c) Is the result in BCNF?

The 3NF synthesis only guarantees that the result is in 3NF, it may not be in BCNF.

- $\{A, C, E\}$ with $\Sigma_1 = \{\ \{A\} \rightarrow \{C\},\ \{A\} \rightarrow \{E\}, \{C, E\} \rightarrow \{A\}\ \}$
- $\{F, H\}$ with $\Sigma_2 = \{\ \{F\} \rightarrow \{H\}\ \}$

(c) Is the result in BCNF?

The 3NF synthesis only guarantees that the result is in 3NF, it may not be in BCNF.

- $\{A, C, E\}$ with $\Sigma_1 = \{ \{A\} \rightarrow \{C\},\ \{A\} \rightarrow \{E\}, \{C, E\} \rightarrow \{A\} \}$
- $\{F, H\}$ with $\Sigma_2 = \{ \{F\} \rightarrow \{H\} \}$
- $\{B, C, D, E\}$ with $\Sigma_3 = \{ \{B, C, E\} \rightarrow \{D\} \}$

(c) Is the result in BCNF?

The 3NF synthesis only guarantees that the result is in 3NF, it may not be in BCNF.

- $\{A, C, E\}$ with $\Sigma_1 = \{\ \{A\} \rightarrow \{C\},\ \{A\} \rightarrow \{E\}, \{C, E\} \rightarrow \{A\}\ \}$
- $\{F, H\}$ with $\Sigma_2 = \{\ \{F\} \rightarrow \{H\}\ \}$
- $\{B, C, D, E\}$ with $\Sigma_3 = \{\ \{B, C, E\} \rightarrow \{D\}\ \}$
- $\{B, C, E, F, G\}$ with $\Sigma_4 = \{\ \{B, C, E, F\} \rightarrow \{G\}\ \}$

YES! It is in BCNF!

(c) Is the result in BCNF?

The 3NF synthesis only guarantees that the result is in 3NF, it may not be in BCNF.

- $\{A, C, E\}$ with $\Sigma_1 = \{ \{A\} \rightarrow \{C\}, \{A\} \rightarrow \{E\}, \{C, E\} \rightarrow \{A\} \}$
- $\{F, H\}$ with $\Sigma_2 = \{ \{F\} \rightarrow \{H\} \}$
- $\{B, C, D, E\}$ with $\Sigma_3 = \{ \{B, C, E\} \rightarrow \{D\} \}$
- $\{B, C, E, F, G\}$ with $\Sigma_4 = \{ \{B, C, E, F\} \rightarrow \{G\} \}$

Note that it is not always the case that we are lucky to obtain a BCNF decomposition using 3NF synthesis, but it may happen.

🏁 One Last Lap

💙 **One last class to go!** (BCNF + Extra problems + doubt clearing) — hope to all of you there! 📚✨

💬 If you enjoyed the module, please take a moment to leave your **feedback/rating** after the course — it really helps me grow as a teacher! 🙏💫

👋 I'll see you all around campus — don't hesitate to say hi or wave! 😄🎓

Thank you for joining!

Got questions? Post them on the forum or email me:

biswadeep@u.nus.edu

(I reply **within 2 working days** — *faster if coffee is strong* ☕)

Because your learning matters to me! 😊

NUS

National University of Singapore

You can generate all implied FDs by running the above closure for **every** subset X⊆UX \subseteq UX⊆U (U = all attributes):

```
Σplus := ∅

for each X ⊆ U:

    S := AttrClosure(X, Σ)

    for each attribute A in (S - X):

        add FD  X → A  to Σplus   // (often we ignore trivial A∈X)
```

Subsumption

Keep

In some cases like $R = \{A, B, C\}$ with $\Sigma = \{ \{A,B\} \rightarrow \{C\}, \{C\} \rightarrow \{B\} \}$, we ***cannot remove*** $R_2 = \{B, C\}$ even when it is subsumed by $R_1 = \{A, B, C\}$.

R2(B, C)

```sql
CREATE TABLE R2 (
  B  INT,
  C  INT
    UNIQUE,
  PRIMARY KEY (B, C)

);
```

R1(A, B, C)

```sql
CREATE TABLE R1 (
  A  INT,
  B  INT,
  C  INT,
  PRIMARY KEY (A, B),
  FOREIGN KEY (B, C) REFERENCES R2(B, C)
);
```

Lemma #1: Lossless-Join Binary Decomposition

A **binary decomposition** of R into R_1 and R_2 is ***lossless-join*** if $R = R_1 \cup R_2$ and $(R_1 \cap R_2) \rightarrow R_1$ or $(R_1 \cap R_2) \rightarrow R_2$

Lemma #2: Lossless-Join Decomposition

A **decomposition** of R into R_1, R_2, ..., R_n is ***lossless-join*** if there exists **at least one sequence** of binary lossless-join decomposition that generates that decomposition.

Note

If $(R_1 \cap R_2)$ is the primary key of one of the two tables, then it can be a foreign key in the other table referencing the primary key.

0) Pick an anchor that contains a key

Compute $(BCEF)^+$ under Σ:

- BCEF ⟶ G (BCEF→G) ⇒ add G
- BCE ⟶ D (BCE→D) ⇒ add D
- CE ⟶ A (CE→A) ⇒ add A
- F ⟶ H (F→H) ⇒ add H

So $(BCEF)^+$ = **A B C D E F G H ⇒ BCEF is a key**.
Fragment **R_4(BCEFG)** contains BCEF ⇒ use **R_4** as the anchor.

1) Join R_4(BCEFG) with R_3(BCDE)

- Intersection: **$R_4 \cap R_3$ = {B,C,E}**.
- Check binary test: does {B,C,E} ⟶ R_3 ?
  Yes: {B,C,E} ⟶ D (given) and trivially ⟶ {B,C,E}, so {B,C,E} ⟶ **BCDE**.

✅ Step 1 is **lossless**.

2) Join the result with R_1(ACE)

(Current attributes after step 1: {B,C,E,F,G,D})

- Intersection: **{C,E}**.
- Check: {C,E} ⟶ R_1 ?
  Yes: {C,E} ⟶ A (given) and trivially ⟶ {C,E} ⇒ {C,E} ⟶ **ACE**.

✅ Step 2 is **lossless**.

3) Join the result with R_2(FH)

(Current attributes after step 2: {A,B,C,D,E,F,G})

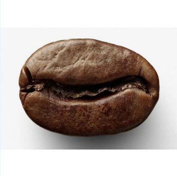
- Intersection: **{F}** (H isn't in the current set yet).
- Check: {F} ⟶ R_2 ?
  Yes: {F} ⟶ H (given) and trivially ⟶ F ⇒ {F} ⟶ **FH**.

✅ Step 3 is **lossless**.

Problem Overview

Design a relational schema for the management of coffee bean, drinks and cafes

The Coffee Bean Entity

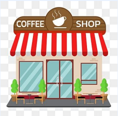


- Identified by *BrandName* **OR** *(Cultivar, Region)*
- One bean → many drinks
- Attributes: *(BrandName, Cultivar, Region)*
- PK: *(BrandName)* or *(Cultivar, Region)*

Drink Entity

- Made from **one coffee bean**
- Name unique **per bean**
- Attributes: *(BeanID, DrinkName, Price)*
- PK: *(BeanID, DrinkName)*

Branch



- Represents a **physical coffee shop branch**
- Each branch has a **unique name**
- **Attributes:** (BranchName, Address)
- **PK:** (BranchName)