

# Beyond Patterns: Meaningful Functional Dependencies with Classical Algorithms and LLMs

Khalid Belhajjame

LAMSADE, Univ. Paris-Dauphine, PSL

# Functional Dependencies

## Reminder.

A functional dependency (FD) expresses a constraint between attributes:

$$X \rightarrow Y$$

If two tuples agree on  $X$ ,  
they must agree on  $Y$ .

## Example:

StudentID	Email	Program
1042	alice@dauphine.eu	CS
1042	alice@dauphine.eu	CS
2091	bob@dauphine.eu	Math
3177	clara@dauphine.eu	MIAGE

# Functional Dependencies

## Reminder.

A functional dependency (FD) expresses a constraint between attributes:

$$X \rightarrow Y$$

If two tuples agree on  $X$ , they must agree on  $Y$ .

## Example:

StudentID	Email	Program
1042	alice@dauphine.eu	CS
1042	alice@dauphine.eu	CS
2091	bob@dauphine.eu	Math
3177	clara@dauphine.eu	MIAGE

## (Some) Valid FDs:

StudentID  $\rightarrow$  Email

StudentID  $\rightarrow$  Program

## Why are FDs important?

- Schema design (3NF, BCNF)
- Data quality and validation
- Dataset profiling
- Query optimization
- Data integration / matching
- Key discovery

# Not All Functional Dependencies Are Meaningful

## Observation:

FD algorithms discover:

All dependencies that hold in the data

Those that **make sense** in the real world

## Reason:

FD discovery algorithms operate on:

- Values
- Equality patterns
- Attribute combinations

They do **not** use:

- Domain knowledge
- Semantics
- Causality
- Meaning

# Not All Functional Dependencies Are Meaningful

## Observation:

FD algorithms discover:

All dependencies that hold in the data

Those that **make sense** in the real world

## Reason:

FD discovery algorithms operate on:

- Values
- Equality patterns
- Attribute combinations

They do **not** use:

- Domain knowledge
- Semantics
- Causality
- Meaning

## Consequence:

An FD may be:

- True in the dataset
- Perfectly valid formally

Yet:

- Conceptually wrong
- Meaningless

## Example:

ZipCode  $\rightarrow$  Height

Holds in dataset But its Absurd in reality

# Classes of Meaningless Functional Dependencies

These FDs may hold in the dataset, but do not represent real-world rules.

**Accidental FDs** (coincidence in this dataset)

$\text{ZipCode} \rightarrow \text{Height}$

Example:  $(75016 \rightarrow 180), (75008 \rightarrow 175)$

# Classes of Meaningless Functional Dependencies

These FDs may hold in the dataset, but do not represent real-world rules.

**Accidental FDs** (coincidence in this dataset)

$\text{ZipCode} \rightarrow \text{Height}$

Example:  $(75016 \rightarrow 180), (75008 \rightarrow 175)$

**Degenerate FDs** (ID explains everything)

$\text{StudentID} \rightarrow \text{Email}, \text{Program}$

Example:  $1042 \rightarrow \text{alice@dauphine.eu}$

# Classes of Meaningless Functional Dependencies

These FDs may hold in the dataset, but do not represent real-world rules.

**Accidental FDs** (coincidence in this dataset)

$\text{ZipCode} \rightarrow \text{Height}$

Example:  $(75016 \rightarrow 180), (75008 \rightarrow 175)$

**Degenerate FDs** (ID explains everything)

$\text{StudentID} \rightarrow \text{Email}, \text{Program}$

Example:  $1042 \rightarrow \text{alice@dauphine.eu}$

**Encoding-based FDs** (information embedded in codes)

$\text{OrderID} \rightarrow \text{Country}$

Example:  $\text{ORD-2023-FR-0042} \rightarrow \text{FR}$

# Classes of Meaningless Functional Dependencies

These FDs may hold in the dataset, but do not represent real-world rules.

**Accidental FDs** (coincidence in this dataset)

$\text{ZipCode} \rightarrow \text{Height}$

Example:  $(75016 \rightarrow 180), (75008 \rightarrow 175)$

**Degenerate FDs** (ID explains everything)

$\text{StudentID} \rightarrow \text{Email}, \text{Program}$

Example:  $1042 \rightarrow \text{alice@dauphine.eu}$

**Encoding-based FDs** (information embedded in codes)

$\text{OrderID} \rightarrow \text{Country}$

Example:  $\text{ORD-2023-FR-0042} \rightarrow \text{FR}$

**Local FDs** (true only in subgroup)

$\text{Country} \rightarrow \text{VAT}$

Example: EU only:  $\text{FR} \rightarrow 20\%, \text{DE} \rightarrow 19\%$

# Classes of Meaningless Functional Dependencies

These FDs may hold in the dataset, but do not represent real-world rules.

**Accidental FDs** (coincidence in this dataset)

$\text{ZipCode} \rightarrow \text{Height}$

Example:  $(75016 \rightarrow 180), (75008 \rightarrow 175)$

**Degenerate FDs** (ID explains everything)

$\text{StudentID} \rightarrow \text{Email}, \text{Program}$

Example:  $1042 \rightarrow \text{alice@dauphine.eu}$

**Encoding-based FDs** (information embedded in codes)

$\text{OrderID} \rightarrow \text{Country}$

Example:  $\text{ORD-2023-FR-0042} \rightarrow \text{FR}$

**Local FDs** (true only in subgroup)

$\text{Country} \rightarrow \text{VAT}$

Example: EU only:  $\text{FR} \rightarrow 20\%, \text{DE} \rightarrow 19\%$

**Overfitted FDs** (excessive LHS)

$\text{Zip, Age, Gender} \rightarrow \text{Salary}$

Example: Holds only in sample

# Classes of Meaningless Functional Dependencies

These FDs may hold in the dataset, but do not represent real-world rules.

**Accidental FDs** (coincidence in this dataset)

$\text{ZipCode} \rightarrow \text{Height}$

Example:  $(75016 \rightarrow 180), (75008 \rightarrow 175)$

**Degenerate FDs** (ID explains everything)

$\text{StudentID} \rightarrow \text{Email}, \text{Program}$

Example:  $1042 \rightarrow \text{alice@dauphine.eu}$

**Encoding-based FDs** (information embedded in codes)

$\text{OrderID} \rightarrow \text{Country}$

Example:  $\text{ORD-2023-FR-0042} \rightarrow \text{FR}$

**Local FDs** (true only in subgroup)

$\text{Country} \rightarrow \text{VAT}$

Example: EU only:  $\text{FR} \rightarrow 20\%, \text{DE} \rightarrow 19\%$

**Overfitted FDs** (excessive LHS)

$\text{Zip}, \text{Age}, \text{Gender} \rightarrow \text{Salary}$

Example: Holds only in sample

**Spurious FDs** (random pattern)

$\text{ProductCode} \rightarrow \text{Weight}$

Example:  $\text{XJ-123} \rightarrow 300\text{g}$

# Classes of Meaningless Functional Dependencies

These FDs may hold in the dataset, but do not represent real-world rules.

**Accidental FDs** (coincidence in this dataset)

$\text{ZipCode} \rightarrow \text{Height}$

Example:  $(75016 \rightarrow 180), (75008 \rightarrow 175)$

**Degenerate FDs** (ID explains everything)

$\text{StudentID} \rightarrow \text{Email}, \text{Program}$

Example:  $1042 \rightarrow \text{alice@dauphine.eu}$

**Encoding-based FDs** (information embedded in codes)

$\text{OrderID} \rightarrow \text{Country}$

Example:  $\text{ORD-2023-FR-0042} \rightarrow \text{FR}$

**Local FDs** (true only in subgroup)

$\text{Country} \rightarrow \text{VAT}$

Example: EU only:  $\text{FR} \rightarrow 20\%, \text{DE} \rightarrow 19\%$

**Overfitted FDs** (excessive LHS)

$\text{Zip, Age, Gender} \rightarrow \text{Salary}$

Example: Holds only in sample

**Spurious FDs** (random pattern)

$\text{ProductCode} \rightarrow \text{Weight}$

Example:  $\text{XJ-123} \rightarrow 300\text{g}$

**Important:**

- All may be true in data.
- None are necessarily meaningful.
- Algorithms cannot distinguish these cases alone.

# Purpose of the Assignment

**Goal:** Move from discovering *all* functional dependencies to discovering *useful and meaningful* ones.

# Purpose of the Assignment

**Goal:** Move from discovering *all* functional dependencies to discovering *useful and meaningful* ones.

In this assignment, you will:

- Apply classical FD discovery algorithms (TANE / FastFD)
- Use LLMs as assistants (not solvers)
- Distinguish:
  - ▶ correct FDs vs false ones
  - ▶ meaningful FDs vs accidental ones
- Design a hybrid pipeline combining:
  - ▶ sampling
  - ▶ LLM-based reasoning
  - ▶ algorithmic verification

**Key idea:**

*Functional dependencies are constraints on data,  
not necessarily knowledge about the world.*

# Assignment Logistics

## Group Work:

- Groups of **up to 4 or 5 students**
- All members must contribute actively

# Assignment Logistics

## Group Work:

- Groups of **up to 4 or 5 students**
- All members must contribute actively

## Deliverables:

Each group must submit:

- A short written report (PDF)
- Source code / notebooks
- A list of prompts used with LLMs
- A summary of discovered FDs
- A reflection on failures and limitations

# Assignment Logistics

## Group Work:

- Groups of **up to 4 or 5 students**
- All members must contribute actively

## Deliverables:

Each group must submit:

- A short written report (PDF)
- Source code / notebooks
- A list of prompts used with LLMs
- A summary of discovered FDs
- A reflection on failures and limitations

## Presentation / Demonstration:

- Each group will give a short demonstration
- You will explain:
  - ▶ your pipeline design
  - ▶ your findings
  - ▶ your mistakes
- Demonstration will take place in the **next session**

# Task Set 1 — Interpreting Algorithmic FDs

## Objective:

Understand what an FD discovery algorithm produces before reasoning about meaning or LLM assistance.

For each dataset, you are given:

- A dataset (CSV)
- A list of minimal functional dependencies (FDs)

Your goal:

- Understand the structure of the discovered FDs
- Identify trivial and suspicious dependencies
- Analyze their characteristics

# Task Set 1 — What to Do

For each dataset:

- Read the provided list of FDs
- Compute:
  - ▶ number of FDs
  - ▶ average LHS size
  - ▶ attribute frequency (as LHS and RHS)
- Identify:
  - ▶ ID-based FDs
  - ▶ very large determinants
  - ▶ suspicious dependencies

**You do NOT run FD discovery algorithms.**

# Task Set 2 — LLM-Assisted Semantic FD Discovery

## Objective:

Use LLMs to reason about the *meaning* of functional dependencies, not to re-discover them.

In this task, you will:

- Use an LLM as a semantic assistant
- Judge plausibility and meaning of algorithmic FDs
- Compare human judgment with LLM judgment

## Reminder:

LLMs evaluate **meaning**, not **validity in data**.

# Task Set 2 — What to Do

For each dataset:

- Select at least:
  - ▶ 3 plausible FDs
  - ▶ 3 suspicious FDs
- For each FD, query the LLM:  
*“Does this dependency make sense in the real world?”*
- Classify each FD into:
  - ▶ meaningful
  - ▶ accidental
  - ▶ encoding-based
  - ▶ degenerate
  - ▶ unlikely

## **Important:**

Do not use LLMs to extract FDs from the dataset.

# Task Set 2 — Required Output

For each analyzed FD, produce a table:

FD	LLM judgment	Your judgment	Agreement?
$A \rightarrow B$	meaningful	accidental	No

## Additionally:

- Report at least 2 disagreements
- Explain:
  - ▶ why the LLM is wrong, or
  - ▶ why the algorithm is misleading

# Task Set 3 — Sampling and FD Hypotheses

## Objective:

Study how functional dependencies suggested by LLMs on samples can differ from those holding in the full dataset.

In this task, you will:

- Sample the dataset
- Use LLMs to suggest FDs from limited data
- Compare hypotheses with algorithmic FDs

## Key idea:

Sampling creates hypotheses, not truth.

# Task Set 3 — What to Do

For each dataset:

- Create at least:
  - ▶ one random sample (max 50 rows)
  - ▶ one stratified or biased sample
- For each sample:
  - ▶ show it to the LLM
  - ▶ ask for likely FDs
- Collect all candidate FDs

## **Prompt constraint:**

Do *not* show the full dataset or FD list.

# Task Set 3 — Validation

For each FD proposed by the LLM:

- Check manually or with code whether it holds on:
  - ▶ the sample
  - ▶ the full dataset
- Report:
  - ▶ violations
  - ▶ approximate validity (if any)

**Answer:**

- Which FDs are false positives?
- Which are not minimal?
- Which are misleading but “look right”?

# Task Set 3 — Insight

This task should convince you that:

- Sampling hides violations
- Samples may reverse dependencies
- LLMs generalize from tiny evidence

## Key realization:

Empirical patterns on samples are not constraints.

# Task Set 4 — Hybrid FD Discovery

## Objective:

Design a system that combines:

- algorithmic FD discovery output
- LLM-based reasoning

to improve usefulness of results.

In this task, you will:

- invent a hybrid design
- implement part of it
- evaluate its output

# Task Set 4 — What to Build

Design a pipeline that includes at least:

- one LLM-based component
- one verification-based component

Choose at least one role for the LLM:

- FD semantic filter
- Candidate ranking
- Hypothesis generator
- Suspicious FD detector

**You must draw your pipeline diagram.**

# Task Set 4 — Evaluation

For your hybrid method, report:

- reduced noise level (subjective)
- loss of potentially valid FDs

## **Discuss:**

- What did you gain?
- What did you lose?

# Provided Datasets and Functional Dependencies

All datasets and their corresponding minimal FD files are provided in the assignment ZIP file.

The FDs were generated using the **TANE** algorithm.

Dataset	Source	Cols	Rows	Size	#FDs
iris	UCI	5	150	5 KB	4
balance-scale	UCI	5	625	7 KB	1
chess	UCI	7	28,056	519 KB	1
abalone	UCI	9	4,177	187 KB	137
nursery	UCI	9	12,960	1 MB	1
breast-cancer-wisconsin	UCI	11	699	20 KB	46
bridges	UCI	13	108	6 KB	142
echocardiogram	UCI	13	132	6 KB	538
adult	UCI	14	48,842	3.5 MB	78
hepatitis	UCI	20	155	8 KB	8,250
horse	UCI	27	300	25 KB	128,726

# Tool Support for Task Set 4 — Metanome

## Important:

For Task Set 4, you will need to run an FD discovery tool.

Although FDs are provided for Tasks 1–3, you will generate new results in Task Set 4.

## Recommended Tool: Metanome

- Research-grade data profiling system
- Includes FD algorithms TANE and FastFD that we have seen in the course.

## Documentation:

- [Metanome Algorithms Page](#)

## Download:

- [Metanome v1.2 Binary \(with Tomcat\)](#)

## Run:

- Requires Java 1.8
- Launch `run.sh` / `run.bat`
- Open browser at: `http://localhost:8080`

## Support:

Live demo in class.