Data, Data Storage, Data Collection

Lecture 9: Normalization

Romain Pascual

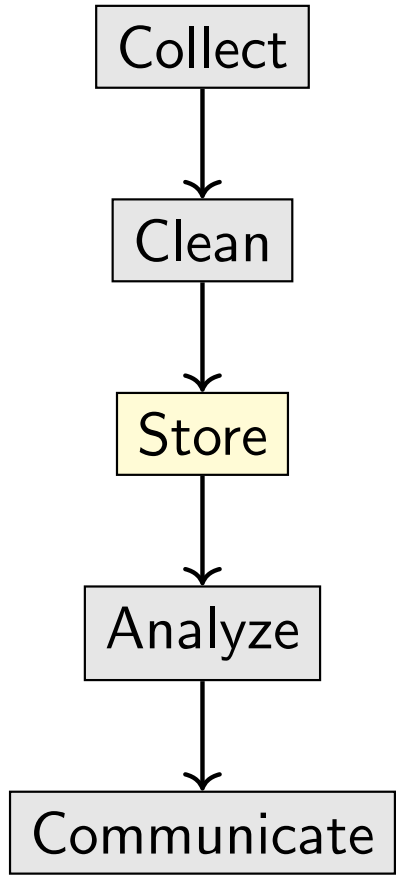
MICS, CentraleSupélec, Université Paris-Saclay

Introduction

Recap: From Data Analysis to Communication

1. **EDA is driven by questions:** it is iterative, and each answer can raise a new question.
2. **Meaning, not encoding:** the feature type decides the summary and the chart.
3. **One feature, then several:** examine distributions before relationships.
4. **An observation is not an interpretation:** check it against noise, confounding and your cleaning choices.
5. **Communicate for an audience:** clear, minimal charts, tailored to the decision at hand.
6. **Show the data as it is:** state uncertainty and limitations, never hide them.

Within The Lifecycle



Session Objectives

By the end of this session, you should be able to:

- Diagnose the design flaw behind an insertion, update or deletion **anomaly**, and explain why normalizing removes it
- Identify the **keys** and **functional dependencies** of a table
- Determine the highest **normal form** of a table, and justify it
- Decompose a table into 3NF or BCNF **without losing information**
- Recognize multivalued and join dependencies (4NF, 5NF)

Bad Data[1]

Flatmate Expenses

Expense	Amount	Paid by	Month
Pizza	25	Sam	2025-11
Internet	40	Alex	2025-11
Coffee	12	Mia	2025-11
Time Machine	250	Bill	2025-11

[1] Running examples adapted from: Decomplexify. **Learn Database Normalization: 1NF, 2NF, 3NF, 4NF, 5NF**. YouTube video. Nov. 2021. URL: https://www.youtube.com/watch?v=GFQaEYEc8_8

Bad Data[1]

Flatmate Expenses

Expense	Amount	Paid by	Month
Pizza	25	Sam	2025-11
Internet	40	Alex	2025-11
Coffee	12	Mia	2025-11
Time Machine	250	Bill	2025-11

⚠ The Time Machine row is almost certainly wrong.

A **validity** problem, found by checking values against the world, not the table's structure.

A table can hold a wrong value without contradicting itself.

[1] Running examples adapted from: Decomplexify. **Learn Database Normalization: 1NF, 2NF, 3NF, 4NF, 5NF**. YouTube video. Nov. 2021. URL: https://www.youtube.com/watch?v=GFQaEYEc8_8

Data Integrity

Flatmate Expenses

Expense	Amount	Paid by	Month
Pizza	25	Sam	2025-11
Internet	40	Alex	2025-11
Coffee	12	Mia	2025-11
Dishwasher	250	Bill	2025-11
Pizza	25	Alex	2025-11
Internet	40	Bill	2025-11

Two new rows arrive. Is either of them wrong?

Data Integrity

Flatmate Expenses

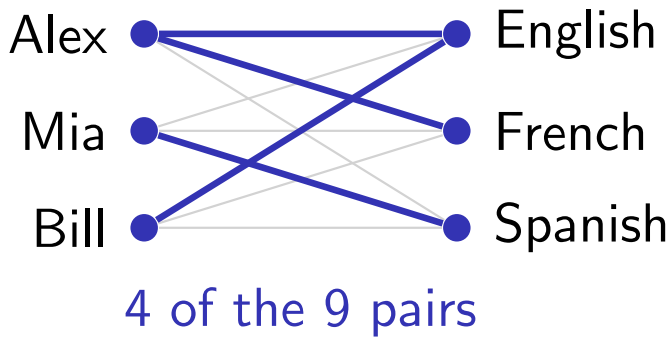
ExpenseID	Expense	Amount	Paid by	Month
1842	Pizza	25	Sam	2025-11
9571	Internet	40	Alex	2025-11
6511	Coffee	12	Mia	2025-11
3427	Dishwasher	250	Bill	2025-11
8689	Pizza	25	Alex	2025-11
9571	Internet	40	Bill	2025-11

Two copies of one fact contradict each other: an **inconsistency**.
No constraint refused it: a failure of **data integrity**.

This is not "just" bad data: it is bad **database design**.

Refresher: What Is a Relation?

A relation between two sets A and B is a subset $R \subseteq A \times B$.



Name	Language
Alex	English
Alex	French
Mia	Spanish
Bill	English

← **schema**

instance

Definition (Relation)

A **relation** $R(A_1, \ldots, A_n)$, with attributes A_i over domains D_i, is a subset $R \subseteq D_1 \times \cdots \times D_n$: a set of tuples.

Like a set, the rows of a relational table (the SQL tables of Storage) have no order and no duplicates.

Data Normalization

Data Normalization

Flatmate Expenses

ExpenseID	Expense	Amount	Paid by	Month
9571	Internet	40	Alex	2025-11
9571	Internet	40	Bill	2025-11

Definition (Normalization)

The process of **structuring** a relational database to reduce data redundancy and improve data integrity.

- How do we **detect** a fact stored several times?
- How do we **fix** it?

Store each fact once.

The Normal Forms

- How do we normalize?
- When have we normalized "enough"?

The Normal Forms

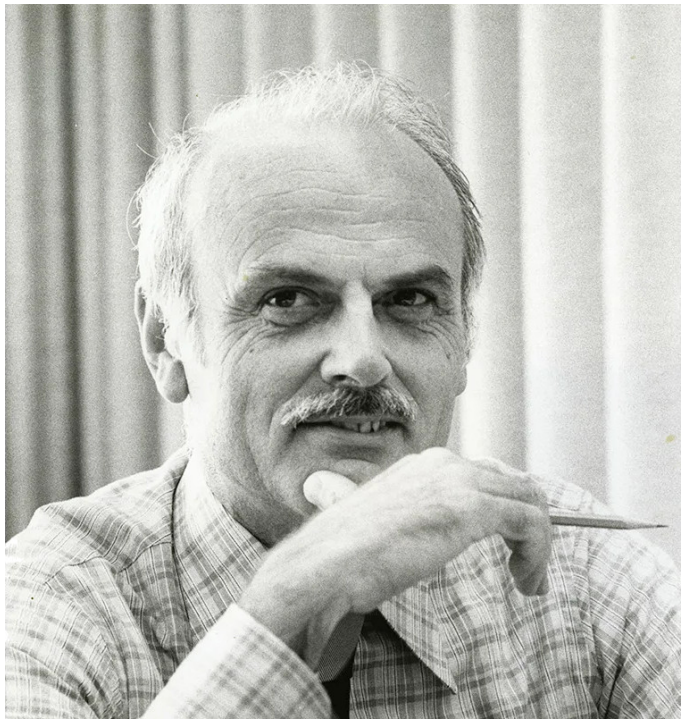
- How do we normalize?
- When have we normalized "enough"?

Normal Forms

Criteria to assess redundant data.

- Proposed by **Edgar F. Codd**[1,2] (relational model)
- Progressive levels of data integrity

1NF $\rightarrow$ **2NF** $\rightarrow$ **3NF** $\rightarrow$ **4NF** $\rightarrow$ **5NF**



Edgar F. Codd
(1923-2003)

[1] Edgar Codd. "A Relational Model of Data for Large Shared Data Banks". In: **Commun. ACM** 13.6 (1970)

[2] Edgar Codd. **Further Normalization of the Data Base Relational Model**. Tech. rep. IBM Research Report RJ909. 1971

Shared Items in the Flat

Who owns what in the flat?

Flat_Members_Items

Name	Age	Shared Items
Alex	23	Microwave, 2 Lamps
Mia	20	Blender, 2 Chairs
Bill	26	Gaming Console
Sam	21	TV



Poll

Does **Shared Items** store single values?

Shared Items in the Flat

Who owns what in the flat?

Flat_Members_Items

Name	Age	Shared Items
Alex	23	Microwave, 2 Lamps
Mia	20	Blender, 2 Chairs
Bill	26	Gaming Console
Sam	21	TV

Poll

Does **Shared Items** store single values? **Depends on the record.**

Definition (Repeating Group)

A collection of values in the field of a record.

Varying sizes and mixed types.

First Normal Form (1NF)

1NF – Codd (1970)

A table is in **1NF** if every value is **atomic**: a single element of its domain, no collection (e.g., lists, sets), nor nested table.

For Codd, 1NF is part of the model: the other NF are guidelines.

Several values

Name	Shared Items
Alex	Microwave, 2 Lamps
Mia	Blender, 2 Chairs
Bill	Gaming Console
Sam	TV

First Normal Form (1NF)

1NF – Codd (1970)

A table is in **1NF** if every value is **atomic**: a single element of its domain, no collection (e.g., lists, sets), nor nested table.

For Codd, 1NF is part of the model: the other NF are guidelines.

Several values

Name	Shared Items
Alex	Microwave, 2 Lamps
Mia	Blender, 2 Chairs
Bill	Gaming Console
Sam	TV

Flat_Items

Name	Item	Qty
Alex	Microwave	1
Alex	Lamp	2
Mia	Blender	1
Mia	Chairs	2
Bill	Gaming Console	1
Sam	TV	1

The **unit of observation** (person, item) becomes the **primary key**.

Adding the Age?

We add each flatmate's age as a new column.

Flat_Info

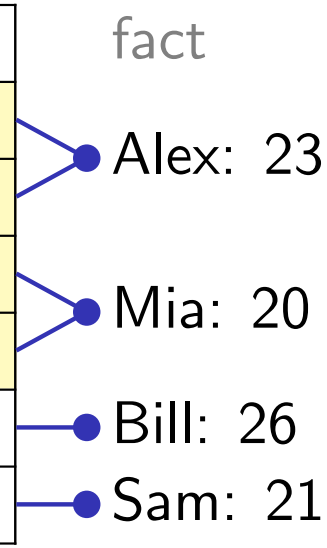
Name	**Item**	**Qty**	**Age**
Alex	Microwave	1	23
Alex	Lamp	2	23
Mia	Blender	1	20
Mia	Chairs	2	20
Bill	Gaming Console	1	26
Sam	TV	1	21

Adding the Age?

We add each flatmate's age as a new column.

Flat_Info

Name	**Item**	**Qty**	**Age**
Alex	Microwave	1	23
Alex	Lamp	2	23
Mia	Blender	1	20
Mia	Chairs	2	20
Bill	Gaming Console	1	26
Sam	TV	1	21



Still in 1NF, but each age is stored once **per item**.

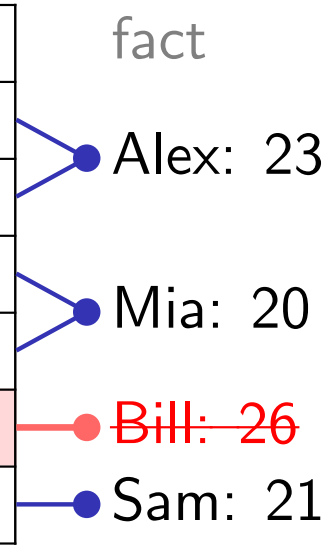
Redundancy: every copy is a chance to disagree.

Adding the Age?

Bill sells his console, so we delete its row.

Flat_Info

Name	**Item**	**Qty**	**Age**
Alex	Microwave	1	23
Alex	Lamp	2	23
Mia	Blender	1	20
Mia	Chairs	2	20
~~Bill~~	~~Gaming Console~~	~~1~~	~~26~~
Sam	TV	1	21



Deletion Anomaly

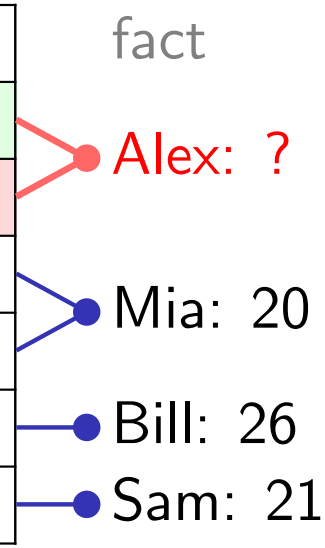
Deleting one fact (Bill owns a console) deletes another: **Bill's age**.

Adding the Age?

Alex turns 24: one birthday, two rows to update.

Flat_Info

Name	**Item**	**Qty**	**Age**
Alex	Microwave	1	24
Alex	Lamp	2	23
Mia	Blender	1	20
Mia	Chairs	2	20
Bill	Gaming Console	1	26
Sam	TV	1	21



Update Anomaly

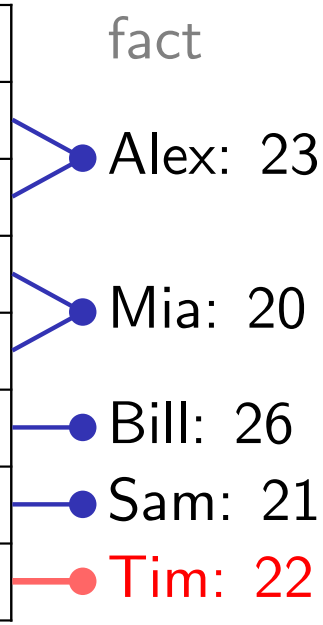
Update only one: Alex is both 23 and 24, an **inconsistency**.

Adding the Age?

Tim joins the flat, but does not bring any new item.

Flat_Info

Name	**Item**	**Qty**	**Age**
Alex	Microwave	1	23
Alex	Lamp	2	23
Mia	Blender	1	20
Mia	Chairs	2	20
Bill	Gaming Console	1	26
Sam	TV	1	21
Tim	?		22



Insertion Anomaly

No item, so no complete key: we cannot insert **Tim's age**.

What Went Wrong?

Deletion, update, insertion: the same cause each time.

Poll

Does each column depend on the entire key?

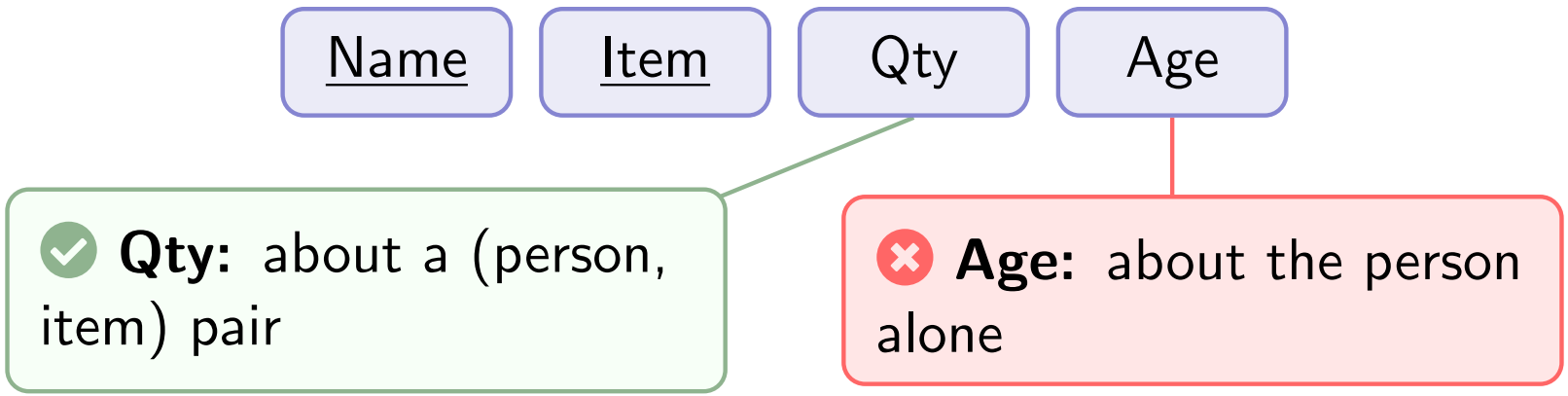
Name	Item	Qty	Age

What Went Wrong?

Deletion, update, insertion: the same cause each time.

Poll

Does each column depend on the entire key?



To state this precisely, on any table, we need some notions defined formally: **functional dependencies** and **keys**.

Functional Dependencies (FDs)

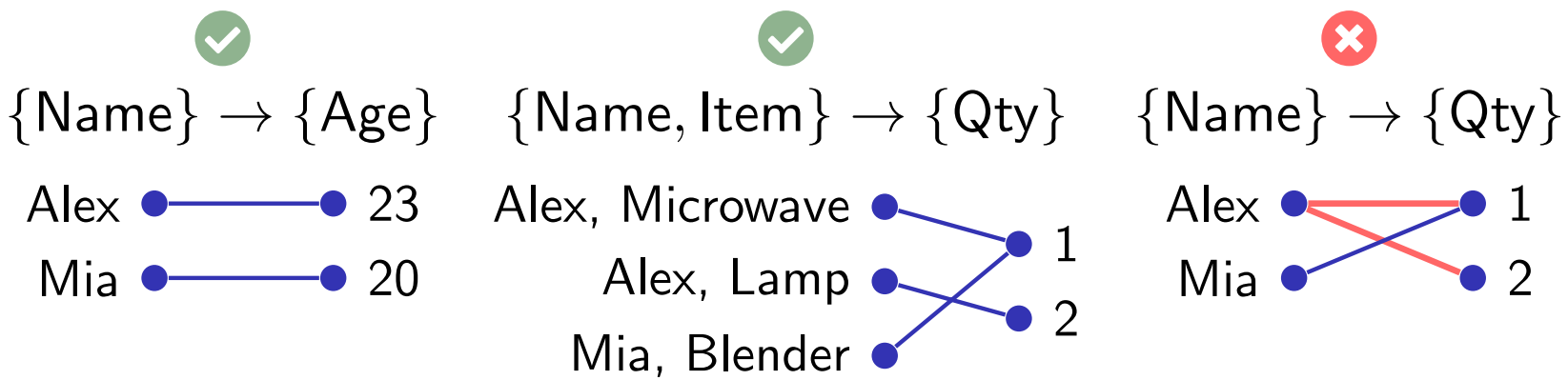
Definition (Functional dependency $X \to Y$)

$X \to Y$ holds if any two rows that agree on X also agree on Y.

X **determines** Y: on X and Y, the rows form a function.

Flat_Info (excerpt)

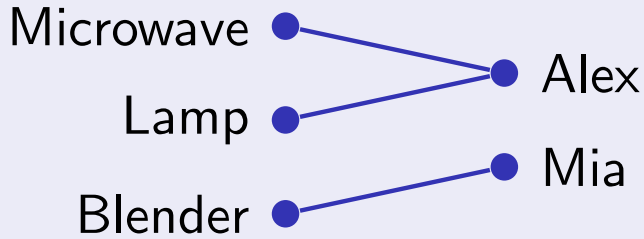
Name	**Item**	**Qty**	**Age**
Alex	Microwave	1	23
Alex	Lamp	2	23
Mia	Blender	1	20



Where Do FDs Come From?

From the table

✓ $\{\text{Item}\} \rightarrow \{\text{Name}\}$

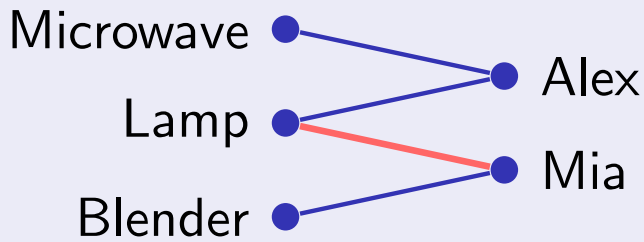


No item has two owners, today.

Where Do FDs Come From?

From the table

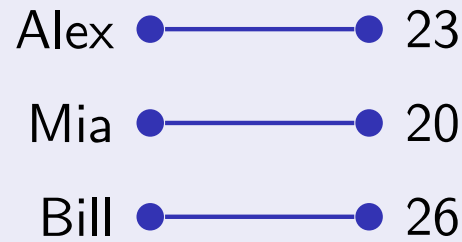
✗ $\{\text{Item}\} \to \{\text{Name}\}$



No item has two owners, today.
Tomorrow, Mia buys a lamp too.

From the domain

✓ $\{\text{Name}\} \to \{\text{Age}\}$



A person has one age:
a rule, for every instance.

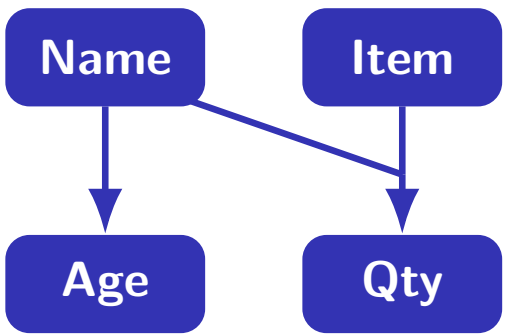
The rows can refute an FD; only the domain can assert it.

Superkey and Candidate Key

Definition (Superkey)

A set of attributes X such that $X \rightarrow$ (all attributes).

Follow the FDs from X: is every attribute reached?



✅ superkey

Superkeys of Flat_Info

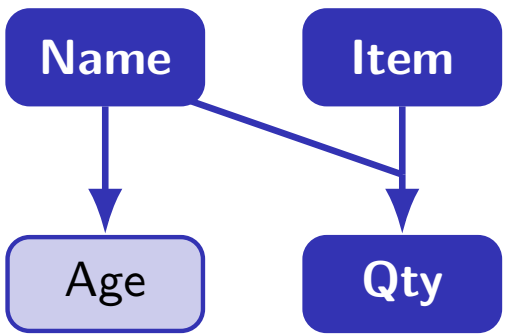
{Name, Item, Qty, Age}

Superkey and Candidate Key

Definition (Superkey)

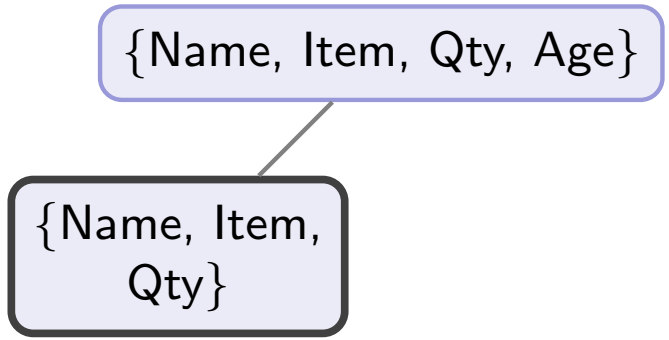
A set of attributes X such that $X \rightarrow$ (all attributes).

Follow the FDs from X: is every attribute reached?



superkey

Superkeys of Flat_Info

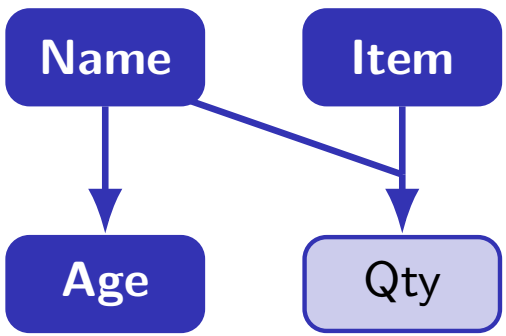


Superkey and Candidate Key

Definition (Superkey)

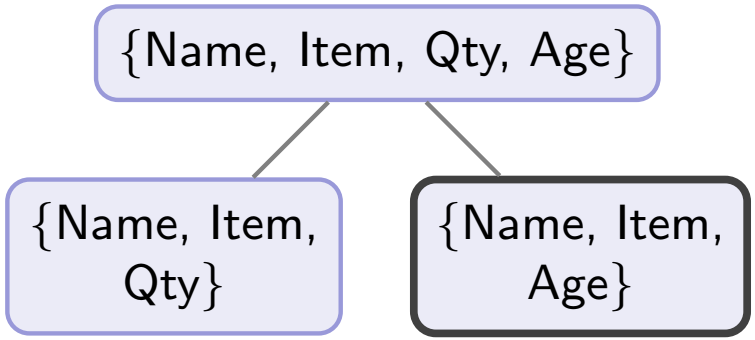
A set of attributes X such that $X \rightarrow$ (all attributes).

Follow the FDs from X: is every attribute reached?



superkey

Superkeys of Flat_Info

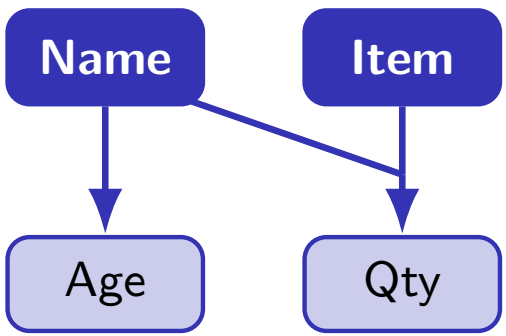


Superkey and Candidate Key

Definition (Superkey)

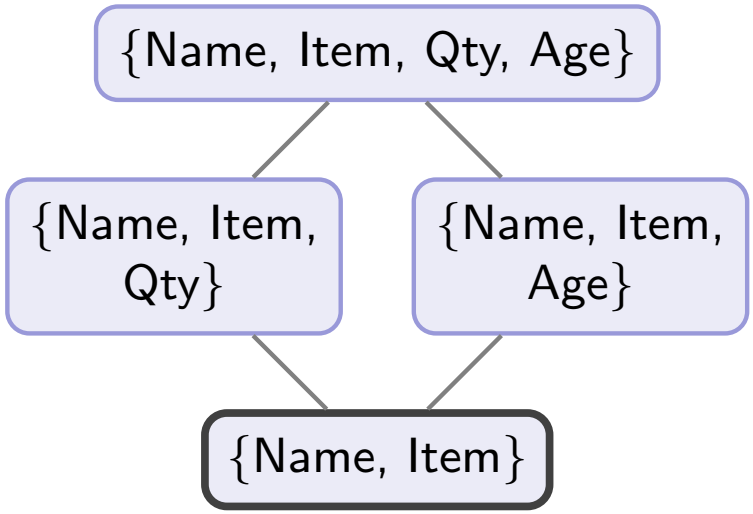
A set of attributes X such that $X \rightarrow$ (all attributes).

Follow the FDs from X: is every attribute reached?



superkey

Superkeys of Flat_Info

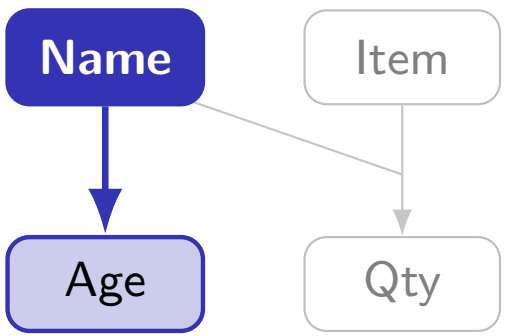


Superkey and Candidate Key

Definition (Superkey)

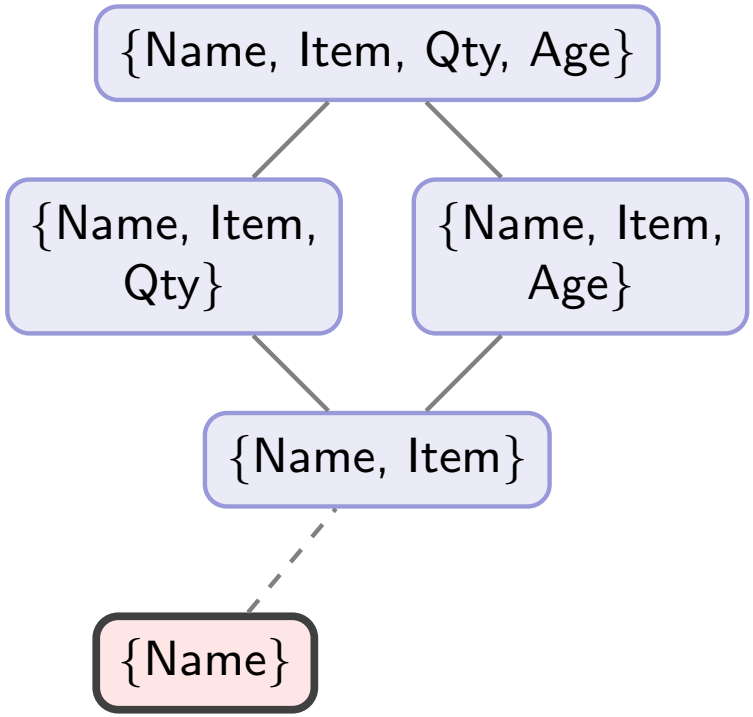
A set of attributes X such that $X \rightarrow$ (all attributes).

Follow the FDs from X: is every attribute reached?



❌ not a superkey

Superkeys of Flat_Info

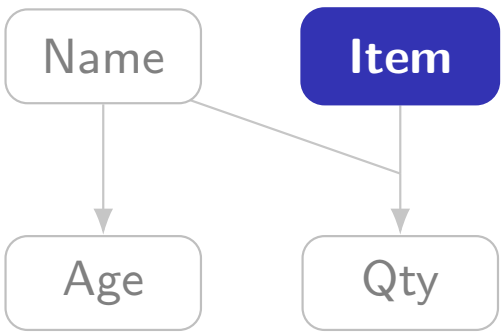


Superkey and Candidate Key

Definition (Superkey)

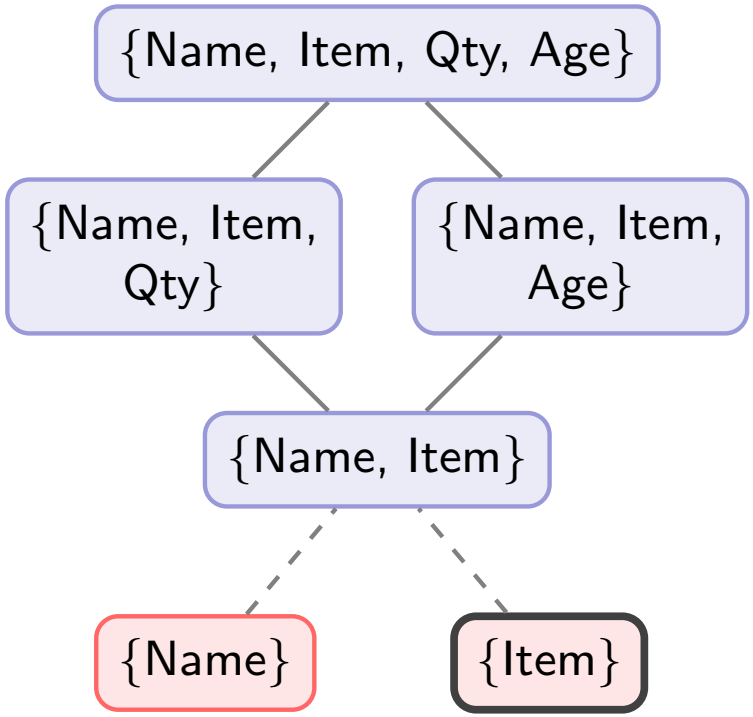
A set of attributes X such that $X \rightarrow$ (all attributes).

Follow the FDs from X: is every attribute reached?



✗ not a superkey

Superkeys of Flat_Info

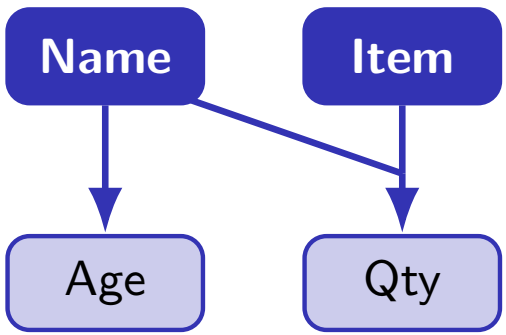


Superkey and Candidate Key

Definition (Superkey)

A set of attributes X such that $X \rightarrow$ (all attributes).

Follow the FDs from X: is every attribute reached?

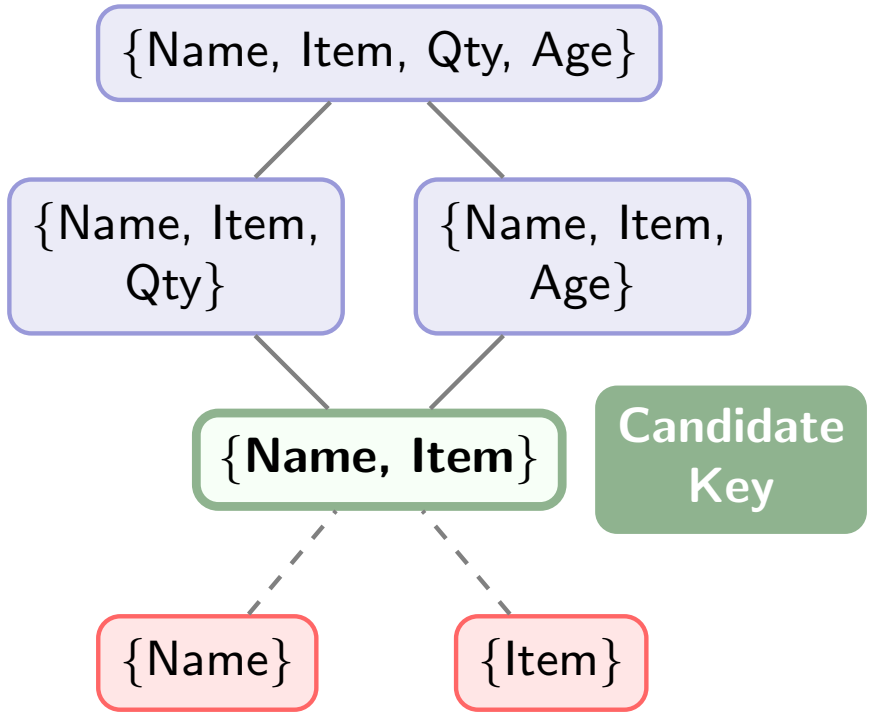


✅ **candidate key**

Definition (Candidate key)

A **minimal** superkey.

Superkeys of Flat_Info



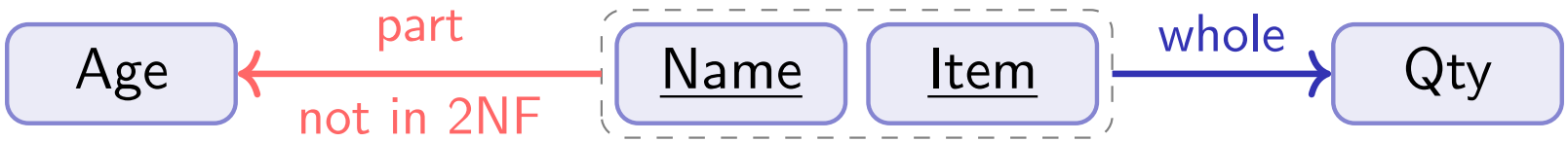
⚠ Minimal does not mean smallest.

2NF

A **prime** attribute belongs to at least one candidate key.

Definition (2NF)

A table in 1NF is in **2NF** if no non-prime attribute depends on a **proper subset** of a candidate key.

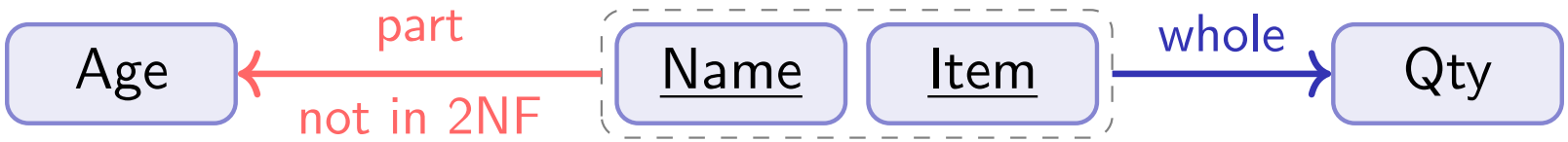


2NF

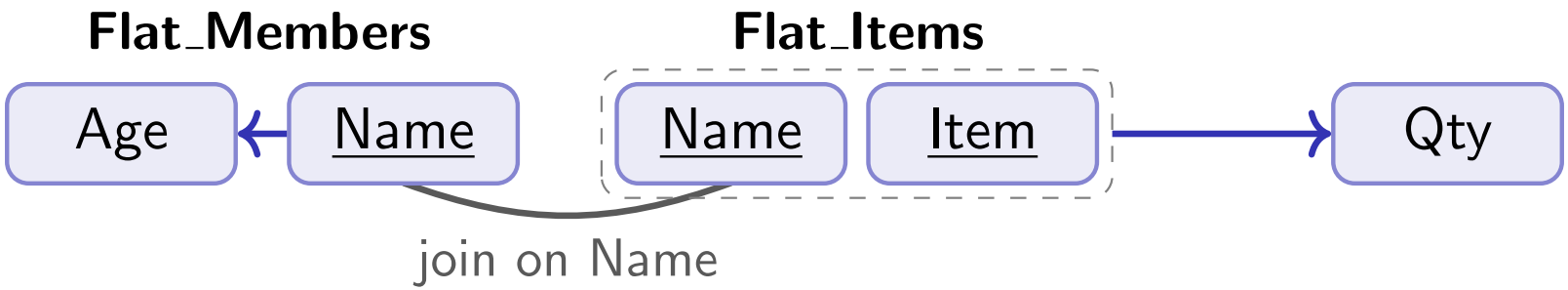
A **prime** attribute belongs to at least one candidate key.

Definition (2NF)

A table in 1NF is in **2NF** if no non-prime attribute depends on a **proper subset** of a candidate key.



Split along the offending functional dependency:



Every non-prime attribute depends on **the whole key**.

Achieving 2NF

Split along $\{\text{Name}\} \rightarrow \{\text{Age}\}$:

Flat_Members

Name	**Age**
Alex	23
Mia	20
Bill	26
Sam	21

Flat_Items

Name	**Item**	**Qty**
Alex	Microwave	1
Alex	Lamp	2
Mia	Blender	1
Mia	Chairs	2
Bill	Gaming Console	1
Sam	TV	1

Joining on Name gives back **Flat_Info**: a **lossless** decomposition.

Achieving 2NF

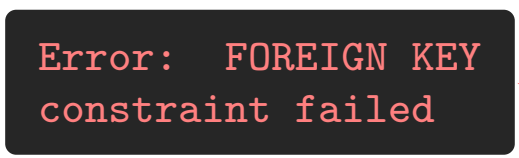
Split along $\{\text{Name}\} \to \{\text{Age}\}$:

Flat_Members

Name	**Age**
Alex	23
Mia	20
Bill	26
Sam	21

Flat_Items

Name	**Item**	**Qty**
Alex	Microwave	1
Alex	Lamp	2
Mia	Blender	1
Mia	Chairs	2
Bill	Gaming Console	1
Sam	TV	1
Tim	Kettle	1



Joining on Name gives back **Flat_Info**: a **lossless** decomposition.

Definition (Foreign key)

A column whose values must appear as a key of another table.

Adding Rent Information

We add room size and rent; the landlord sets the rent by size.

Flat_Members

Name	**Age**	**RoomSize**	**Rent**
Alex	23	Large	500
Mia	20	Small	300
Bill	26	Large	500
Sam	21	Medium	400

Table in 2NF: the key {Name} is a single attribute, so nothing depends on part of it.

Adding Rent Information

Bill swaps rooms with Mia: we move the flatmates.

Flat_Members

Name	**Age**	**RoomSize**	**Rent**
Alex	23	Large	500
Mia	20	Large	300
Bill	26	Small	500
Sam	21	Medium	400

Update Anomaly

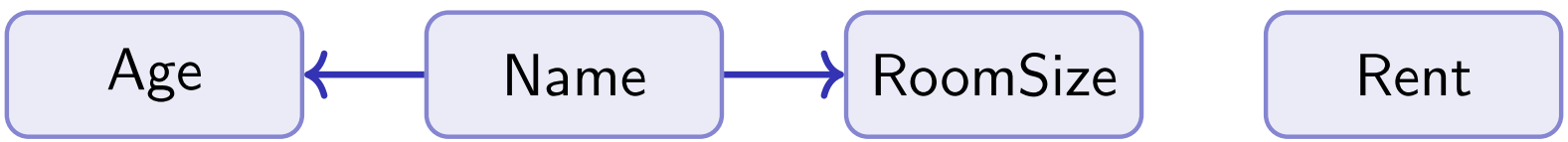
A large room now rents for 500 **and** 300.

Rent Depends on the Room

Which FDs hold in **Flat_Members**?

Flat_Members

Name	**Age**	**RoomSize**	**Rent**
Alex	23	Large	500
Mia	20	Small	300
Bill	26	Large	500
Sam	21	Medium	400

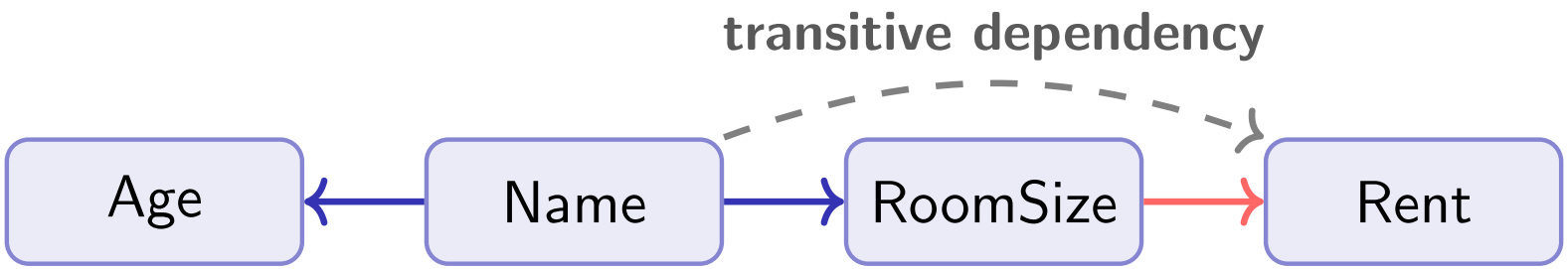


Rent Depends on the Room

Which FDs hold in **Flat_Members**?

Flat_Members

Name	**Age**	**RoomSize**	**Rent**
Alex	23	Large	500
Mia	20	Small	300
Bill	26	Large	500
Sam	21	Medium	400

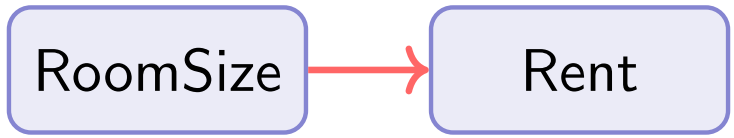


3NF

An FD $X \to A$ is **trivial** if $A \in X$ (it always holds).

Definition (3NF[1])

A table is in **3NF** if for every non-trivial FD $X \to A$, X is a superkey **or** A is prime.



RoomSize is not a superkey and Rent is not prime: not in 3NF.

A non-key field must provide a fact about the key, the whole key, and nothing but the key.[2]

[1] Definitions of 3NF and BCNF as in: Abraham Silberschatz, Henry F. Korth, and S. Sudarshan. **Database System Concepts**. 7th ed. McGraw-Hill, 2019

[2] Quote from: William Kent. "A Simple Guide to Five Normal Forms in Relational Database Theory". In: **Commun. ACM** 26.2 (1983), pp. 120–125

Achieving 3NF

Split along $\{\text{RoomSize}\} \rightarrow \{\text{Rent}\}$: the fact moves to a table keyed by RoomSize.

Flat_Members

Name	**Age**	**RoomSize**
Alex	23	Large
Mia	20	Small
Bill	26	Large
Sam	21	Medium

Room_Rent

RoomSize	**Rent**
Small	300
Medium	400
Large	500

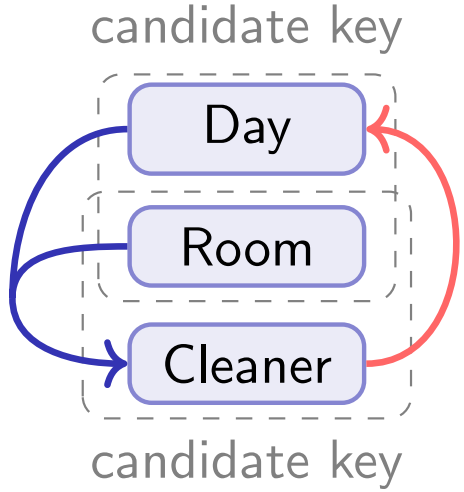
foreign key: no room size without a rent

A Cleaning Schedule

One cleaner per room per day. Each flatmate has a fixed day.

Cleaning_Schedule

Room	**Day**	**Cleaner**
Kitchen	Mon	Alex
Bathroom	Mon	Bill
Kitchen	Sat	Mia
Bathroom	Sat	Mia
Living Room	Mon	Alex



All attributes prime: **3NF**.

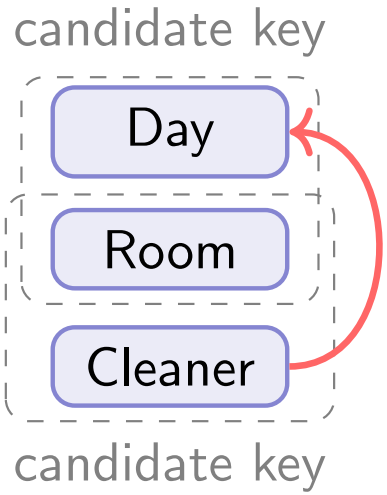
Update Anomaly

"Mia cleans on Saturday" is stored twice: if she moves to Sunday, two rows must change.

Boyce–Codd Normal Form (BCNF)

Definition (BCNF)

A table is in **BCNF** if for every non-trivial FD $X \to A$, X is a superkey.

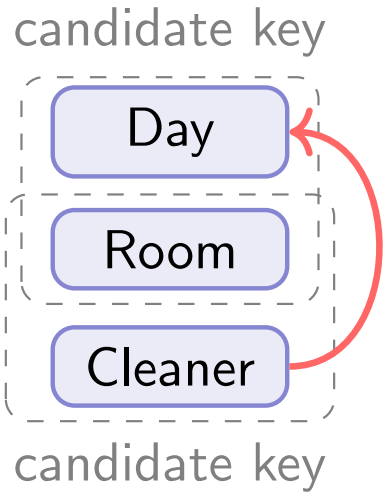


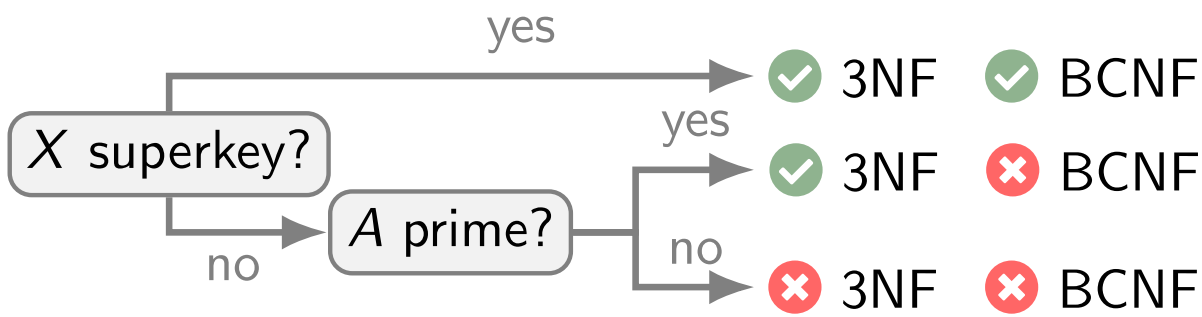
Every attribute depends on the key,
the whole key, and nothing but the key.

Boyce–Codd Normal Form (BCNF)

Definition (BCNF)

A table is in **BCNF** if for every non-trivial FD $X \to A$, X is a superkey.



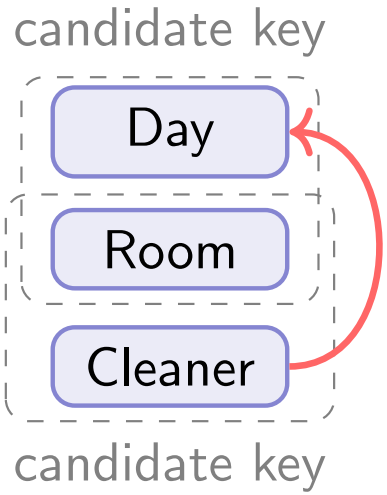


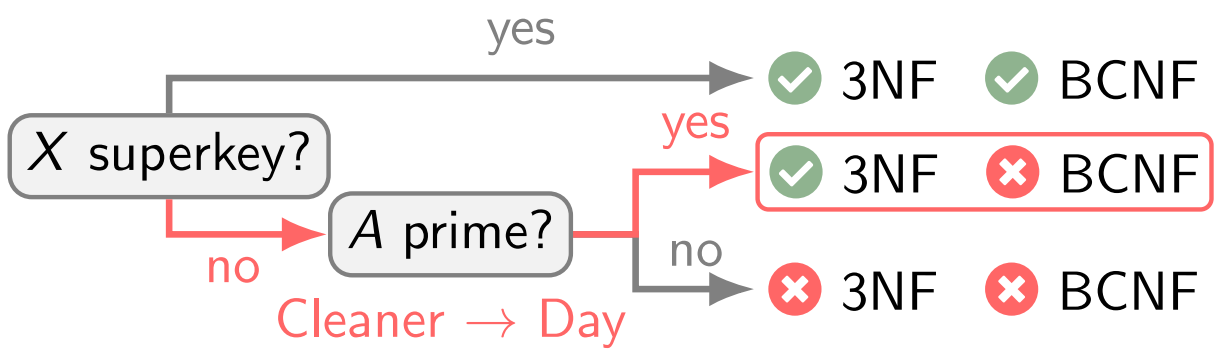
Every attribute depends on the key,
the whole key, and nothing but the key.

Boyce–Codd Normal Form (BCNF)

Definition (BCNF)

A table is in **BCNF** if for every non-trivial FD $X \to A$, X is a superkey.





Every attribute depends on the key,
the whole key, and nothing but the key.

3NF or BCNF?

Split along $\{\text{Cleaner}\} \rightarrow \{\text{Day}\}$:

Cleaner_Day

Cleaner	**Day**
Alex	Mon
Bill	Mon
Mia	Sat

Room_Cleaner

Room	**Cleaner**
Kitchen	Alex
Bathroom	Bill
Kitchen	Mia
Bathroom	Mia
Living Room	Alex

3NF or BCNF?

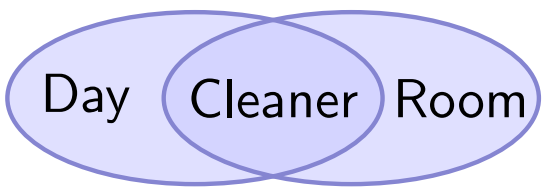
Split along $\{\text{Cleaner}\} \to \{\text{Day}\}$:

Cleaner_Day

Cleaner	**Day**
Alex	Mon
Bill	Mon
Mia	Sat

Room_Cleaner

Room	**Cleaner**
Kitchen	Alex
Bathroom	Bill
Kitchen	Mia
Bathroom	Mia
Living Room	Alex
Kitchen	Bill



⚠ Not **dependency-preserving**: $\{\text{Room}, \text{Day}\} \to \{\text{Cleaner}\}$ fits in no table.

In practice: aim for BCNF; if a dependency is lost, stay in 3NF.

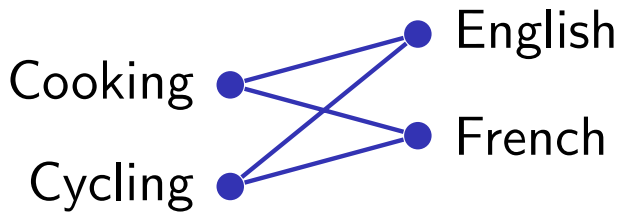
Hobbies and Languages

We record each flatmate's hobbies and the languages they speak.

Flat_Profiles

Name	**Hobby**	**Language**
Alex	Cooking	English
Alex	Cycling	English
Alex	Cooking	French
Alex	Cycling	French
Bill	Cooking	English

Alex's rows



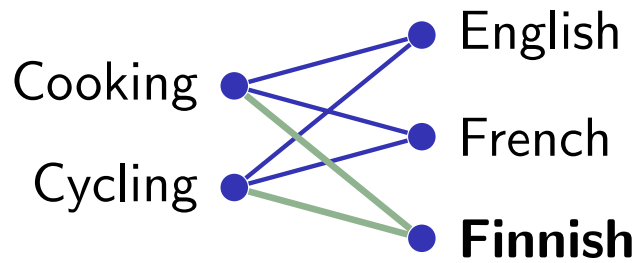
Hobbies and Languages

We record each flatmate's hobbies and the languages they speak.

Flat_Profiles

Name	Hobby	Language
Alex	Cooking	English
Alex	Cycling	English
Alex	Cooking	French
Alex	Cycling	French
Bill	Cooking	English
Alex	Cooking	Finnish
Alex	Cycling	Finnish

Alex's rows



Alex also speaks Finnish.

⚠ Without the second row, Alex speaks Finnish only while cooking.

Update Anomaly

One new fact, several rows to change.

Too Much Independence

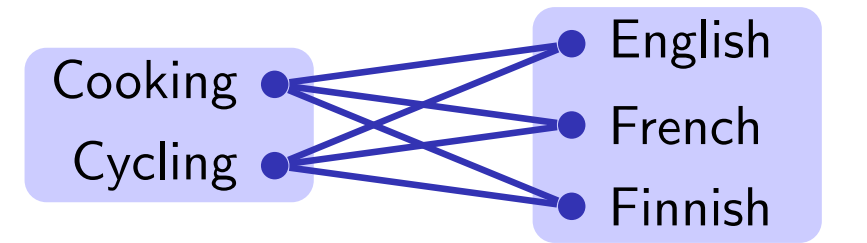
✋ Poll

Which non-trivial FDs hold in **Flat_Profiles**?
Highest normal form: 2NF, 3NF or BCNF?

Too Much Independence

Poll

Which non-trivial FDs hold in **Flat_Profiles**? **None.**
Highest normal form: 2NF, 3NF or BCNF? **BCNF**.

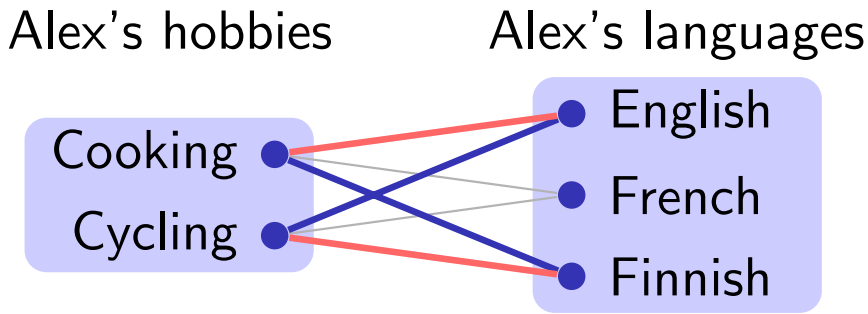


Two independent facts: all their combinations.

Too Much Independence

Poll

Which non-trivial FDs hold in **Flat_Profiles**? **None.**
Highest normal form: 2NF, 3NF or BCNF? **BCNF**.



Two independent facts: all their combinations.

A **value** of X gives a **set** of Y values, **whatever** the other columns.

Definition (Multivalued dependency (MVD) $X \twoheadrightarrow Y$)

In $R(X, Y, Z)$, any two rows that agree on X can swap their Y values and stay in R: for all x, y_i, z_i,

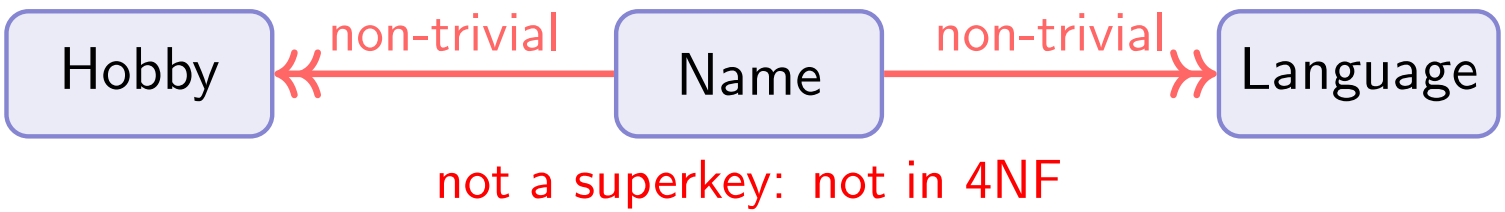
$$(x, y_1, z_1),\ (x, y_2, z_2) \in R \implies (x, y_1, z_2),\ (x, y_2, z_1) \in R$$

4NF

An MVD $X \twoheadrightarrow Y$ is **trivial** if $Y \subseteq X$ or $X \cup Y$ is all of R.

Definition (4NF[1])

A table is in **4NF** if for every non-trivial MVD $X \twoheadrightarrow Y$, X is a superkey.



Same rule as BCNF, with $\twoheadrightarrow$ in place of $\rightarrow$:
independent facts about a key go in separate tables.

[1] Definition as in: William Kent. "A Simple Guide to Five Normal Forms in Relational Database Theory". In: **Commun. ACM** 26.2 (1983), pp. 120–125

Achieving 4NF

Split along $\{\text{Name}\} \twoheadrightarrow \{\text{Hobby}\}$: one table per independent fact.

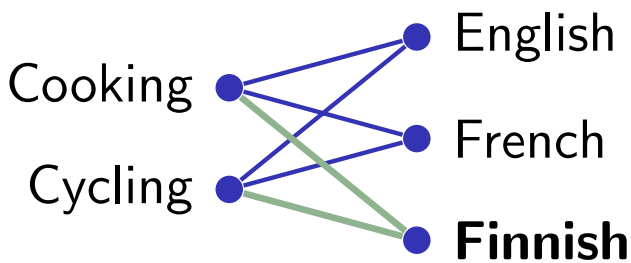
Member_Hobbies

Name	Hobby
Alex	Cooking
Alex	Cycling
Bill	Cooking

Member_Languages

Name	Language
Alex	English
Alex	French
Alex	Finnish
Bill	English

Join on Name: Alex



The join on Name rebuilds every combination, and only those.

✅ Both tables are in 4NF: with two columns, every MVD is trivial.

Flatmate Chores

Sam

I am available for Decorating and Cleaning. I can use a Brush, a Hammer and a Vacuum.

Tools

Brush	**Hammer**	**Vacuum**
Cleaning, Decorating	Repairs	Cleaning

Flatmate Chores

Sam

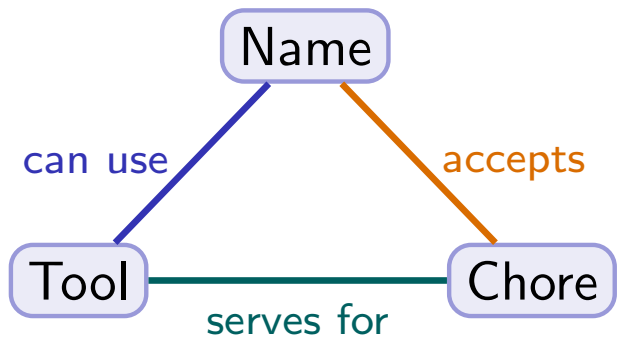
I am available for Decorating and Cleaning. I can use a Brush, a Hammer and a Vacuum.

Tools

Brush	**Hammer**	**Vacuum**
Cleaning, Decorating	Repairs	Cleaning

Flat_Chores

Name	**Tool**	**Chore**
Alex	Brush	Cleaning
Mia	Hammer	Repairs
Sam	Brush	Cleaning
Sam	Brush	Decorating
Sam	Vacuum	Cleaning



No FD and no MVD: **4NF**.

Flatmate Chores

Sam

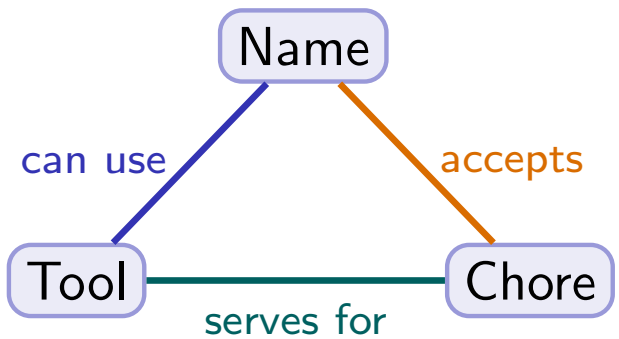
I am available for Decorating and Cleaning. I can use a Brush, a Hammer and a Vacuum.

Tools

Brush	**Hammer**	**Vacuum**
Cleaning, Decorating	Repairs	Cleaning

Flat_Chores

Name	**Tool**	**Chore**
Alex	Brush	Cleaning
Mia	Hammer	Repairs
Sam	Brush	Cleaning
Sam	Brush	Decorating
Sam	Vacuum	Cleaning



No FD and no MVD: **4NF**.

⚠ The table loses Sam's Hammer: Sam does not accept Repairs.

Decomposition into Pairwise Relations

Three pieces of information, so we should have built 3 tables.
When needed, we retrieve the information by joining the tables.

Preferred_Chores

Name	Chore
Alex	Cleaning
Mia	Repairs
Sam	Cleaning
Sam	Decorating

Can_Use

Name	Tool
Alex	Brush
Mia	Hammer
Sam	Brush
Sam	Hammer
Sam	Vacuum

Can_Be_Used_For

Tool	Chore
Brush	Decorating
Brush	Cleaning
Hammer	Repairs
Vacuum	Cleaning

Decomposition into Pairwise Relations

Three pieces of information, so we should have built 3 tables.
When needed, we retrieve the information by joining the tables.

Preferred_Chores

Name	Chore
Alex	Cleaning
Mia	Repairs
Sam	Cleaning
Sam	Decorating

Can_Use

Name	Tool
Alex	Brush
Mia	Hammer
Sam	Brush
Sam	Hammer
Sam	Vacuum

Can_Be_Used_For

Tool	Chore
Brush	Decorating
Brush	Cleaning
Hammer	Repairs
Vacuum	Cleaning

Decomposition into Pairwise Relations

Three pieces of information, so we should have built 3 tables.
When needed, we retrieve the information by joining the tables.

Preferred_Chores

Name	Chore
Alex	Cleaning
Mia	Repairs
Sam	Cleaning
Sam	Decorating

Can_Use

Name	Tool
Alex	Brush
Mia	Hammer
Sam	Brush
Sam	Hammer
Sam	Vacuum

Can_Be_Used_For

Tool	Chore
Brush	Decorating
Brush	Cleaning
Hammer	Repairs
Vacuum	Cleaning

1st join, on Tool

(Alex, Brush, Decorating)

Decomposition into Pairwise Relations

Three pieces of information, so we should have built 3 tables.
When needed, we retrieve the information by joining the tables.

Preferred_Chores

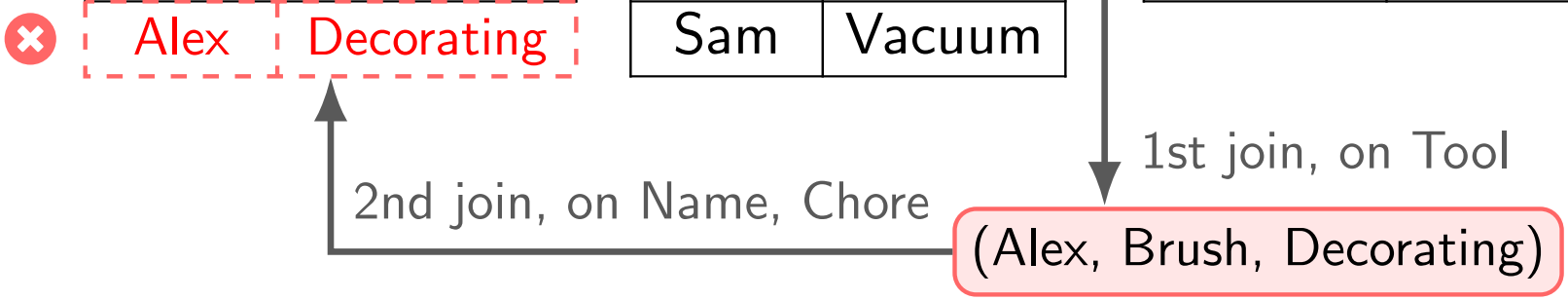
Name	Chore
Alex	Cleaning
Mia	Repairs
Sam	Cleaning
Sam	Decorating
Alex	Decorating

Can_Use

Name	Tool
Alex	Brush
Mia	Hammer
Sam	Brush
Sam	Hammer
Sam	Vacuum

Can_Be_Used_For

Tool	Chore
Brush	Decorating
Brush	Cleaning
Hammer	Repairs
Vacuum	Cleaning

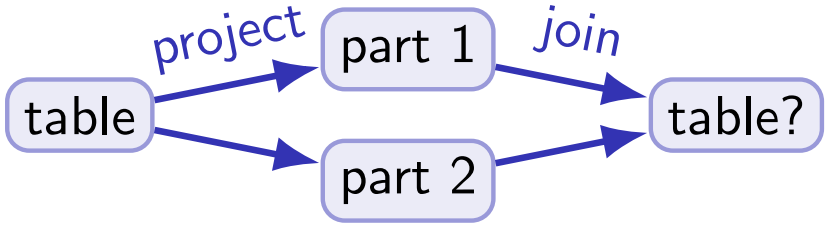


Only the join of all three tables gives back **Flat_Chores**.

Lossless Decomposition and 5NF

Definition (Lossless decomposition)

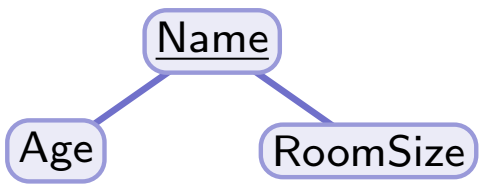
A split of a table is **lossless** if the join of its parts gives back exactly the table.



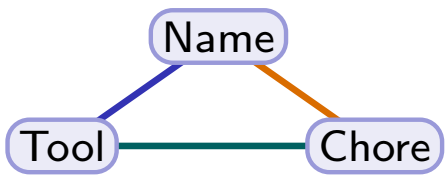
no row lost,
no **spurious row**

Definition (5NF – Projection-Join Normal Form)

A table is in **5NF** if it cannot be described as the logical result of joining other tables, unless they all share a candidate key.



✅ Parts share the key



❌ Three parts, no shared key:
a **join dependency**, not in 5NF

Conclusion

The Normal Forms, in One Line

Each form includes the one above it.

1NF One value per field.

2NF Every non-key attribute depends on the whole key.

3NF . . . and on nothing but the key.

BCNF Only keys determine other attributes.

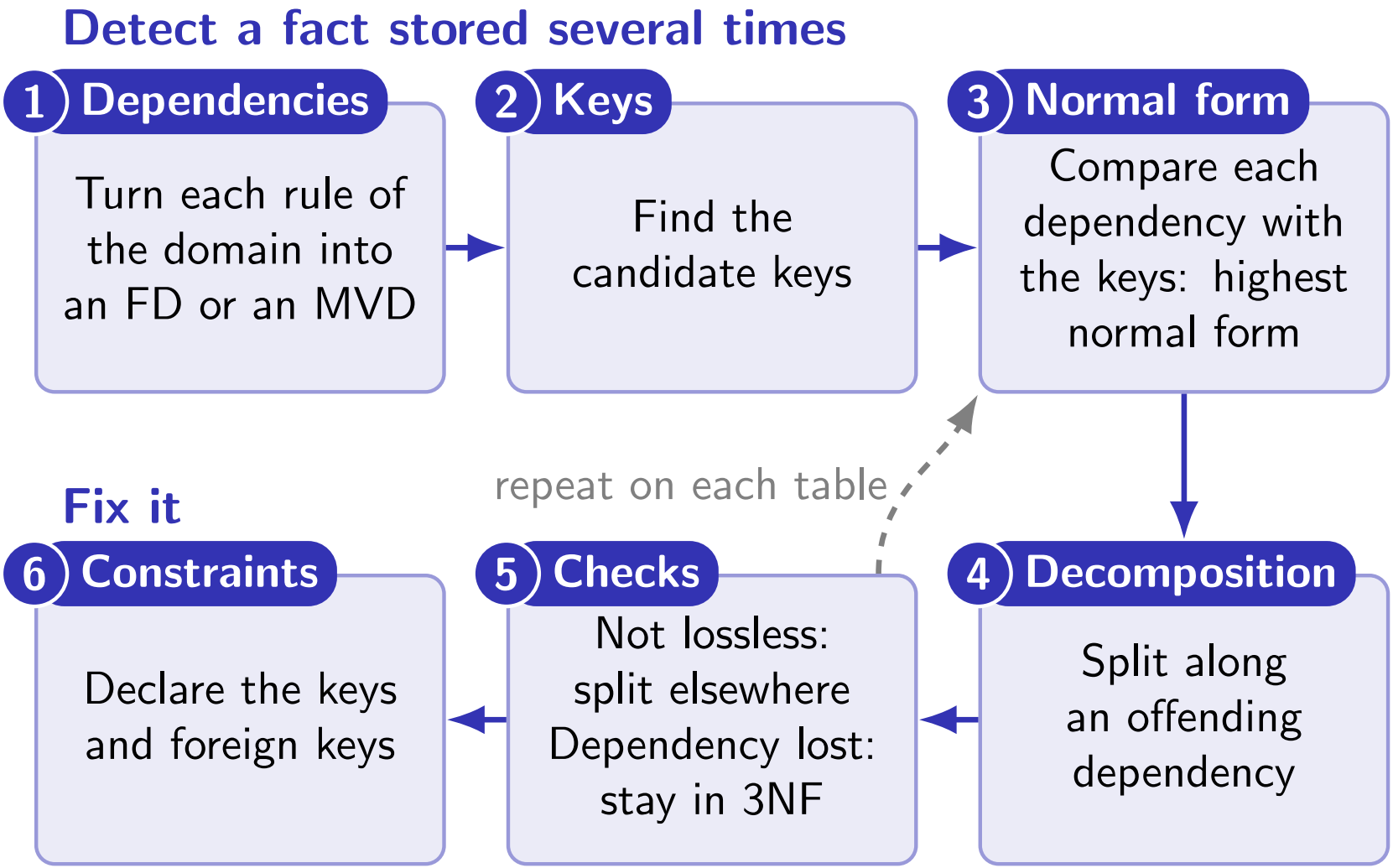
4NF Independent facts about a key go in separate tables.

5NF Not a join of smaller tables, unless on a key.

In practice, aim for BCNF:

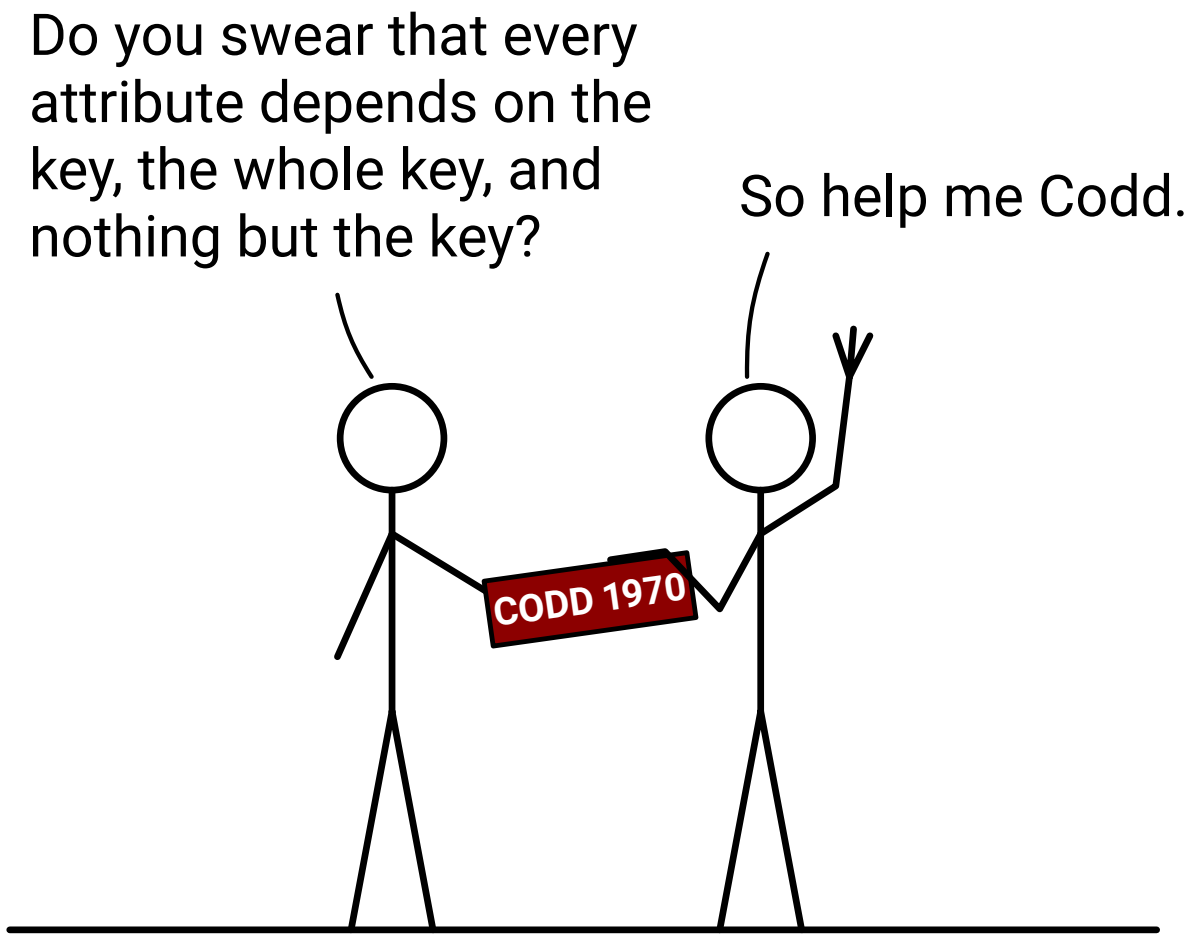
> *Every attribute depends on the key,*
> *the whole key, and nothing but the key.*

Normalizing on Paper



Takeaways: Normalization

1. **Redundancy causes anomalies:** a fact stored with another cannot be inserted, deleted or updated on its own.
2. **Dependencies are rules of the domain:** the rows can refute a dependency, never prove it.
3. **Keys decide the normal form:** find the candidate keys, then check each non-trivial dependency against them.
4. **Normalize by lossless splits:** split along a dependency; the join gives the table back, with no spurious rows.
5. **Not every redundancy is an FD:** independent facts stored as combinations (4NF), several kinds of facts in one table (5NF).
6. **BCNF may not preserve dependencies:** some rules can no longer be checked inside one table.



Store each fact once.

Appendix: formal explanations about 5NF (not required at the exam)

Join Dependencies and 5NF

Definition (Join dependency $\bowtie (R_1, \ldots, R_n)$)

For $R_1 \cup \cdots \cup R_n = R$, it holds in R if R is the join of its projections on $R_1, \ldots, R_n$. It is **trivial** if some R_i is all of R.

Definition (5NF[1])

A table is in **5NF** if every non-trivial join dependency is implied by its candidate keys.

For instance, a join dependency is implied when all the R_i contain the **same** candidate key.

The rule of **Flat_Chores** is
$\bowtie (\{\text{Name}, \text{Chore}\}, \{\text{Name}, \text{Tool}\}, \{\text{Tool}, \text{Chore}\})$.

[1] Definition as in: William Kent. "A Simple Guide to Five Normal Forms in Relational Database Theory". In: **Commun. ACM** 26.2 (1983), pp. 120–125

Flat_Chores: Its Three Projections

A **projection** keeps some columns and removes the duplicate rows.
Flat_Chores (5 rows) has three projections on two columns:

(Name, Tool)

Name	Tool
Alex	Brush
Mia	Hammer
Sam	Brush
Sam	Vacuum

(Tool, Chore)

Tool	Chore
Brush	Decorating
Brush	Cleaning
Hammer	Repairs
Vacuum	Cleaning

(Name, Chore)

Name	Chore
Alex	Cleaning
Mia	Repairs
Sam	Cleaning
Sam	Decorating

 No Sam's Hammer here: **Flat_Chores** does not record it.

Does joining them give back exactly the 5 rows?

Flat_Chores: Joining Two Projections

Join (Name, Tool) and (Tool, Chore) on Tool: each flatmate gets every chore of each of their tools.

The join

Name	Tool	Chore
Alex	Brush	Cleaning
Alex	Brush	Decorating
Mia	Hammer	Repairs
Sam	Brush	Cleaning
Sam	Brush	Decorating
Sam	Vacuum	Cleaning

6 rows: the one in red is not in **Flat_Chores**, so it is **spurious**. The other two pairs of projections also give 6 rows, with one spurious row each.

Two projections are not enough.

Flat_Chores: Joining the Third Projection

Now join (Name, Chore): a row stays only if its (Name, Chore) pair is in that projection.

Spurious row	**Its pair**	**In projection?**
(Alex, Brush, Decorating)	(Alex, Decorating)	No: removed

The other 5 rows stay: they are the rows of **Flat_Chores**, so their pairs are in its projection.

This is a **non-trivial** join dependency: no projection has all three columns.

Joined, the three projections give back exactly **Flat_Chores**.

Flat_Chores: Why It Is Not in 5NF

5NF requires every non-trivial join dependency to be **implied by the candidate keys**: the keys alone must guarantee that the split is lossless.

Implied by a key

Flat_Members(Name, Age, RoomSize), key Name.

Both parts of (Name, Age) and (Name, RoomSize) contain the key.

Not implied

Flat_Chores, key: all three columns.

No part contains the key: only a rule of the domain makes the split lossless.

The rule: a flatmate who accepts a chore and can use a tool for it does that chore with it.

The key does not express that rule: not in 5NF.