NSA (Part 2)
Analytical SQL and Big Data

Big Data Engineering
(formerly Informationssysteme)

Prof. Dr. Jens Dittrich

bigdata.uni-saarland.de

May 11, 2023

Last week:

1. Concrete application: NSA

- Snowden
- global surveillance disclosures
- the book by Eschbach
- data vs metadata
- links for further reading

NSA (Part 2)

2. What are the data management and analysis issues behind this?

Question 1

How do we phrase more complex queries?

- SQL!
- simple examples in Jupyter with DuckDB

NSA (Part 2)

Now:

4. Transfer of the basics to the concrete application

Afterwards:

Question 2

... and what ethical problems arise from these kind of data collections? How do we deal with them?

next week:

Question 3

... and what if the data gets bigger? How do we actually get from SQL to an efficient program?

The Fatal Basic Idea of One of the Queries in the Book "NSA"

Query Idea 1

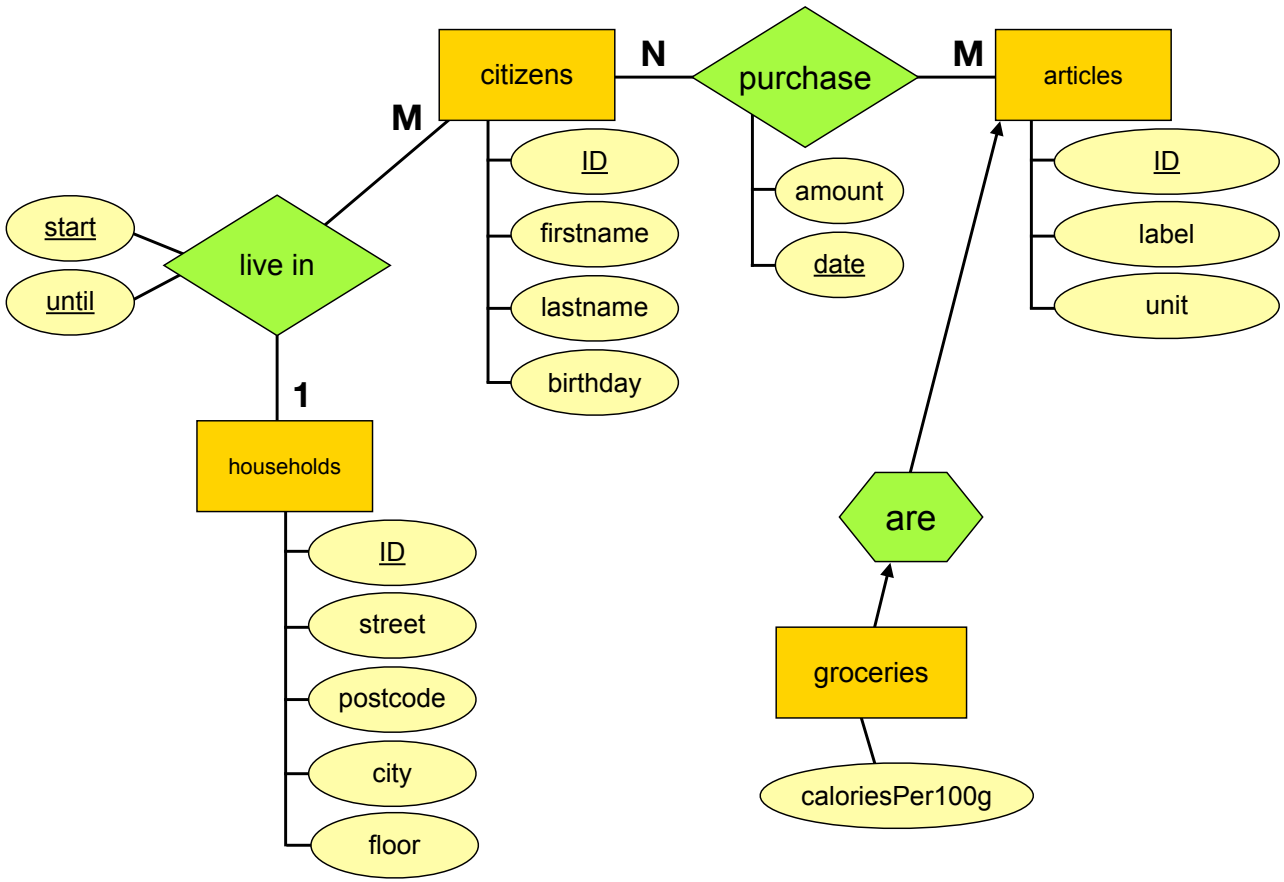
On average, everyone consumes about the same amount of calories. If someone buys on average much more calories than the average, this may indicate that she/he is secretly helping someone else (in other words: hiding!).

So, to answer this query we have to:

1. estimate the average calorie consumption of a person
2. determine per household how many calories are purchased on average
3. determine the average calorie consumption per person in that household
4. output the result list in descending order: highest consumption per person first, only the most conspicuous households
5. in the book, the next step is carried out by the SS...

At which step does the ethical responsibility of us computer scientists and data scientists begin?

Part of the ER Model implied by the Book (Already Improved)

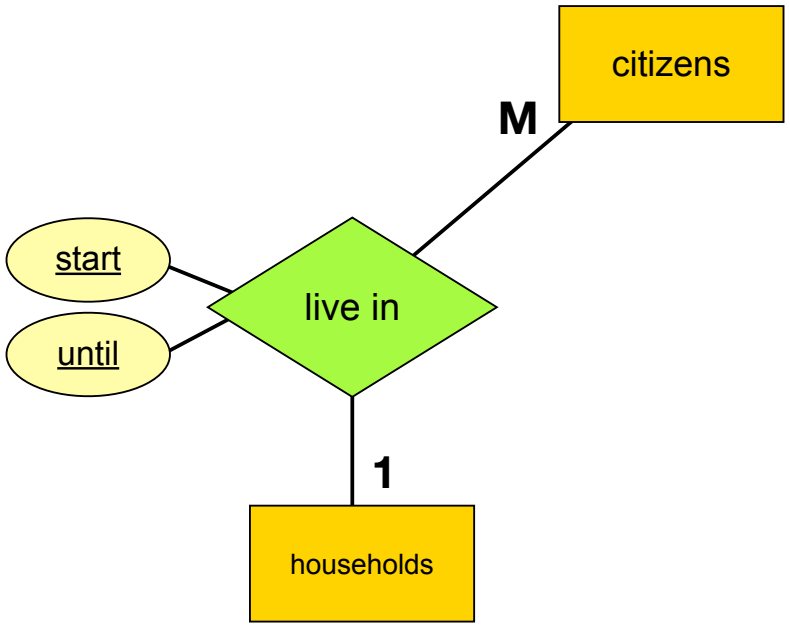


Attention

Time is modelled in "live in" as an additional part of the key!

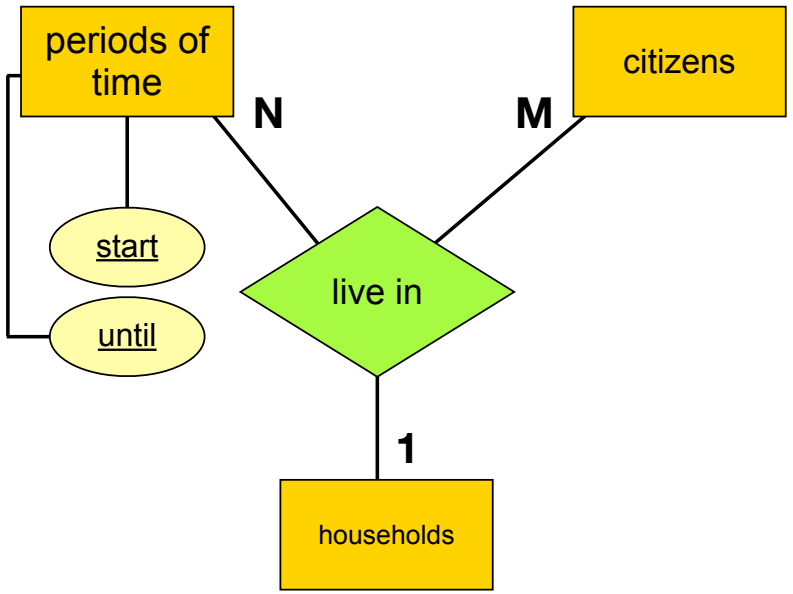
Why? Attributes vs Separate Entity Type

(1) Attributes



Either:
Time periods modelled as key attributes of the relationship type

(2) Separate Entity Type



Or:
Time periods modelled as a separate entity type (with the same key attributes)

Translation to the Relational Model

Translation of (1)

[live in] : {[
citizen_id:(citizens),
start:timestamp,
until:timestamp,
household_id:(households)
]}

Notice that we used a separate relation even though this is a 1:N-relationship!

Translation of (2)

As in (1) plus:

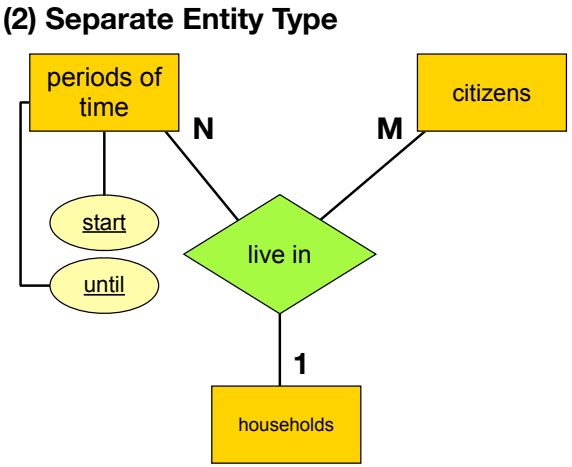
[periods of time] : {[
start:timestamp,
until:timestamp,
]}

An entity type where all its attributes are key attributes is equivalent to a domain!

How to Read n-ary Relationship Types (Exercise)

periods of time × citizens ↦ households

A period of time and a citizen **together determine** the household: for a given period of time and citizen there is either no household or one household but not several. A citizen can only be registered in one household for a given period of time.



households × periods of time ↦ citizens

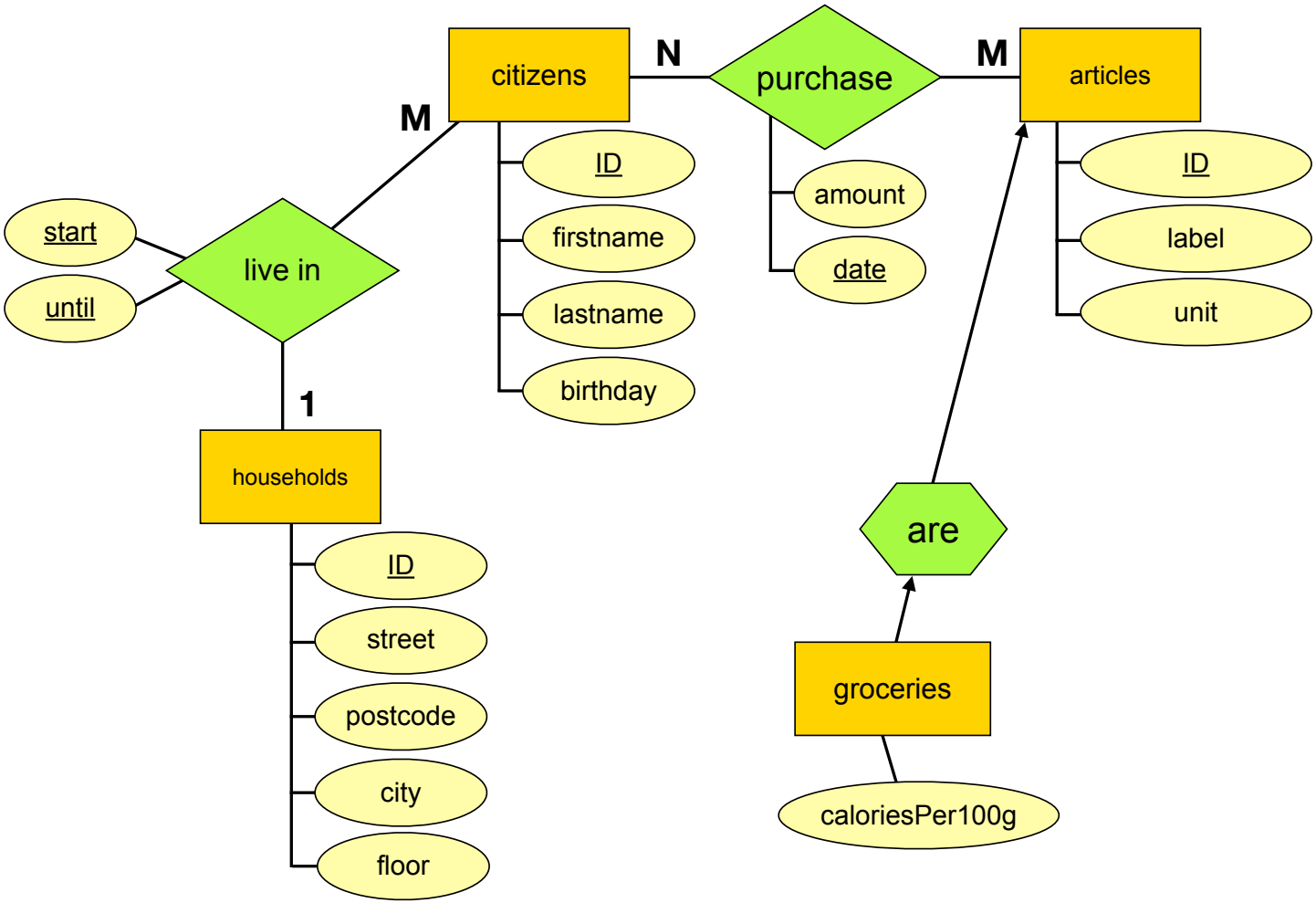
A household and a period of time **together do not determine** the citizen: for a given household and a period of time there are any number of citizens. In the same period of time, several citizens can be registered in the same household.

citizens × households ↦ periods

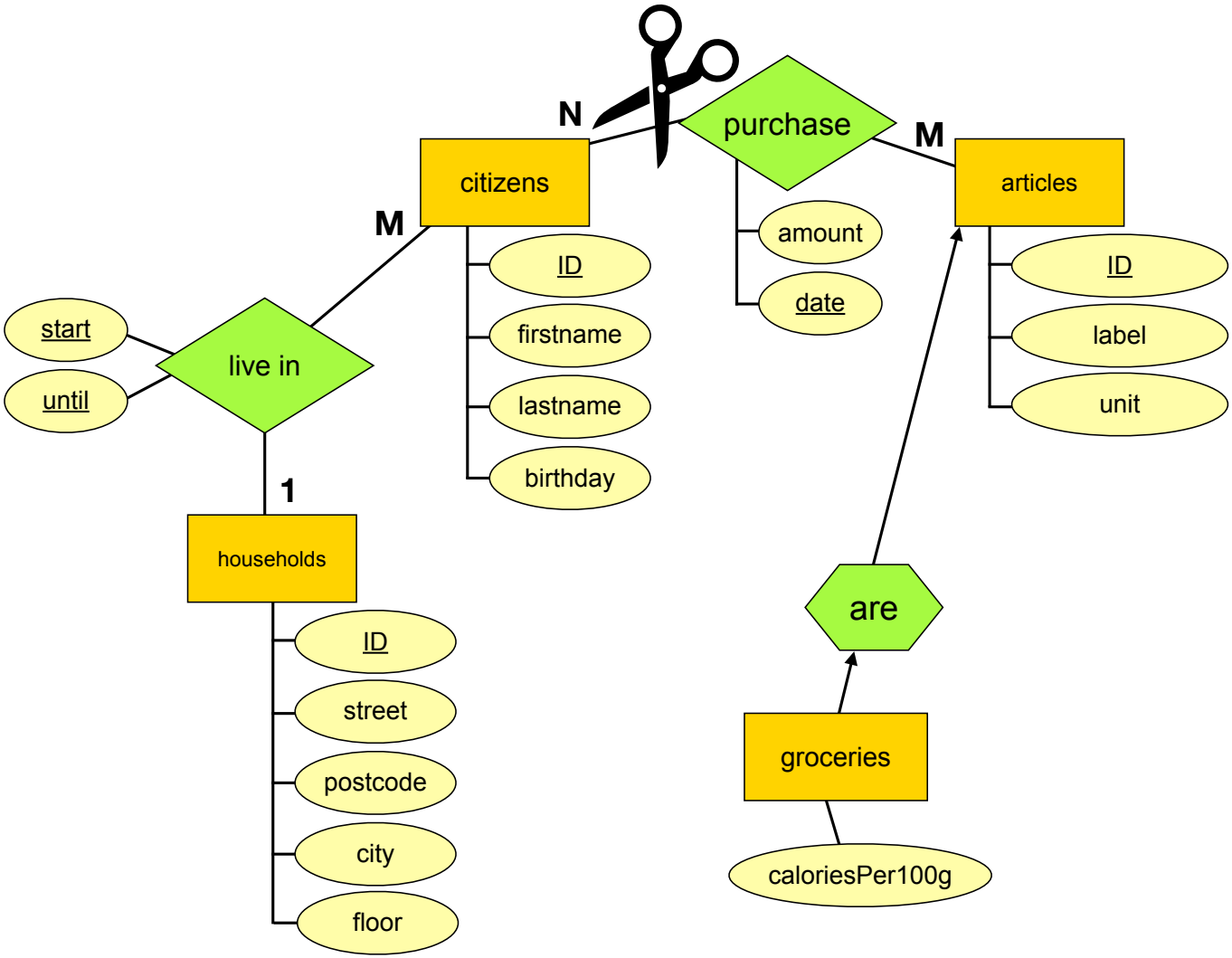
A citizen and a household **together do not determine** the period of time: for a given citizen and household there are any number of periods of time. The same citizen can be registered in the same household at several periods of time.

P.S.: But what actually happens, if [start;until]-periods of time overlap? Oops!

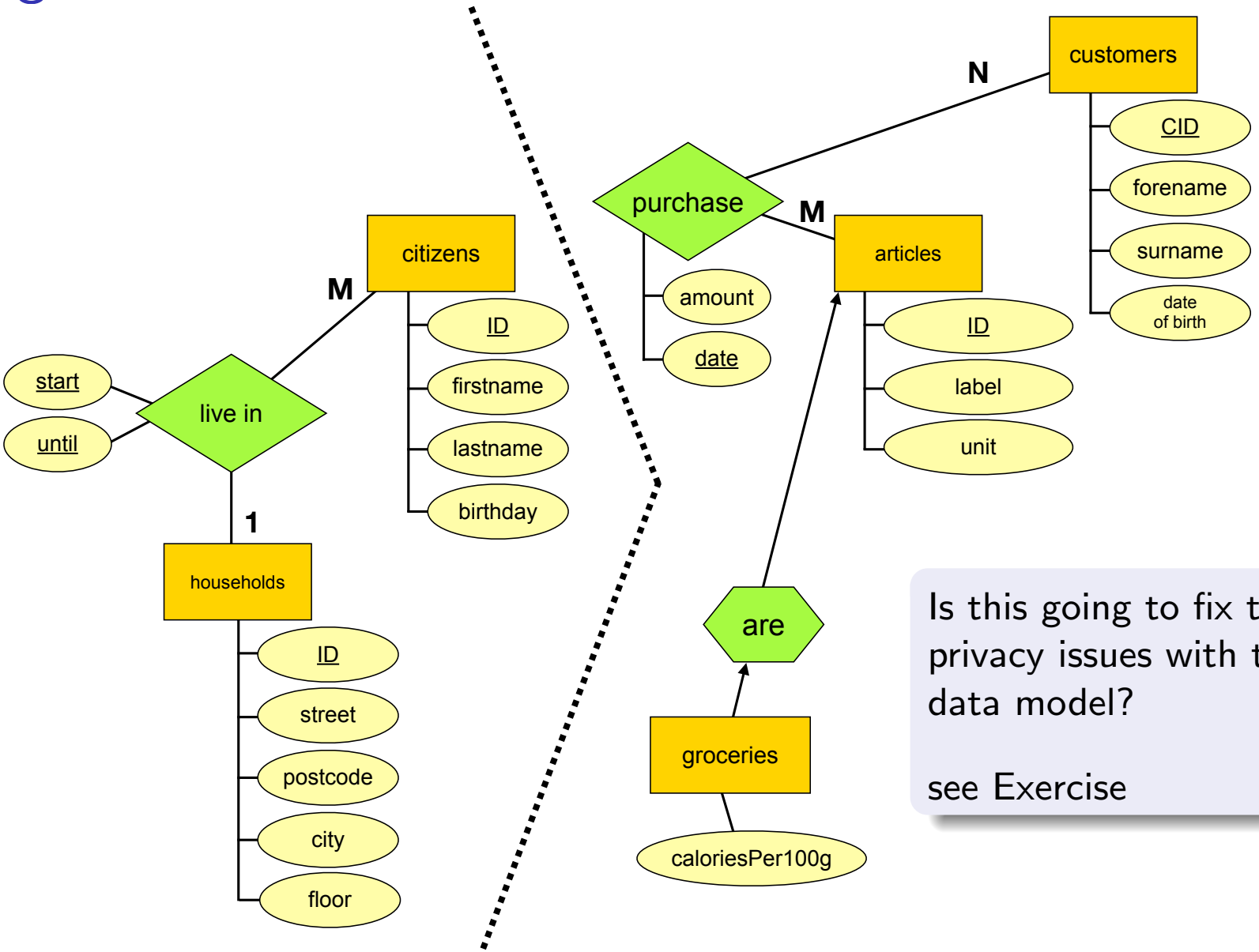
Part of the ER Model from the Book (Already Improved)



Splitting up the ER Model for the Exercise: What can Possibly go Wrong?



Split ER Model for the Exercise: What can Possibly go Wrong?



Is this going to fix the privacy issues with this data model?

see Exercise

Relational Model for the ER Model from the Book

[households] : {[
    ID:int,
    street:str,
    postcode:int,
    city:str,
    floor:int
]}

[citizens] : {[
    ID:int,
    firstname:str,
    lastname:str,
    birthday:timestamp
]}

[articles] : {[
    ID:int,
    label:str,
    unit:str
]}

[groceries] : {[
    ID:(articles),
    caloriesPer100g:int
]}

[live in] : {[
    citizen_ID:(citizens),
    start:timestamp,
    until:timestamp
    household_ID:(households)
]}

[purchases] : {[
    article_ID:(articles),
    citizen_ID:(citizens),
    date:timestamp,
    amount:int
]}

The NSA Scenario from the Book in Python

Citizens & Households

In a first step, we want to show the citizens and the households they currently live in. The current household can be determined by looking at the "until" attribute. If it is `NULL`, the citizen currently lives in this household. If it is not `NULL`, the citizen lived here in the past.

```
duckdb.sql("""
SELECT citizens.firstname, citizens.lastname, households.street, households.postcode, households.city
FROM citizens
    JOIN livingIn ON citizens.id = livingIn.citizen_id
    JOIN households ON households.id = livingIn.household_id
WHERE livingIn.until IS NULL
ORDER BY citizens.firstname
LIMIT 10;""")
```

Count Inhabitants

To search for hidden persons, we first need to know the number of (official) inhabitants of each household. This can be achieved by grouping over the id of the household and counting the number of citizen_id within each group.

```
duckdb.sql("DROP VIEW IF EXISTS inhabitantsPerHousehold;")
duckdb.sql("""
CREATE VIEW inhabitantsPerHousehold AS
    SELECT livingIn.household_id AS household_id,
        COUNT(*) AS numInhabitants
    FROM livingIn
    WHERE livingIn.until is NULL
    GROUP BY livingIn.household_id;""")
```

https://github.com/BigDataAnalyticsGroup/bigdataengineering/blob/master/NSA.ipynb

NSA (Part 1&2)

Question 2

... and what ethical problems arise from these kind of data collections? How do we deal with them?

Ethical Problems

In Eschbach's book, the main character Helene, an NSA employee, conceives various SQL ("SAS" in the book) queries and thus helps to find hidden people.

Effects of Helene's SQL queries

- Without Helene's work, many of these people would probably never have been found.
- What does this mean for Helene's work?

To a large extent, this is already explained in Eschbach's book. With her various queries, Helene also finds her lover, who is also hiding.

Key Question

Key Question

- In this scenario, what would have prevented these people from being found?
- What could Helene have done?

Attention

None of the following information is to be understood as instructions for action. Some of these actions are punishable by law. We are discussing here in principle what possibilities for action Helene or anyone else would have had in the context of civil disobedience/civil resistance in a comparable situation.

Technical Approaches (1/6)

Data economy (in German: Datensparsamkeit)

Example: Cash vs electronic payment transactions

Problem: all in all, very unrealistic by now, but basically helpful; cash is banned in the book, whether this can be used to prevent a black market is questionable

Store only a few attributes

Examples: to show that I have bought something, I do not need to disclose what I have bought. In an application system, do I have to ask for gender or race or religion? (of course not)

Problem: Can I restrict the schema at all beforehand? That's what we do in data modelling anyway! Still, could certain attributes become problematic later on?

Technical Approaches (2/6)

Anonymise data

Example: Anonymising search queries in a search engine

Problem: often de-anonymisable again by joins and mostly only **pseudo-anonymous**, e.g. AOL search queries, [NYT article]

Encrypting parts of the data

at attribute, tuple, table or database level

Problem: join with (unknown) plaintext database possibly breaks this encryption, but certainly better than not encrypting at all (effort for the attacker increases)
Important: make it as difficult as possible for an attacker

Technical Approaches (3/6)

Make joins difficult, remove possible join keys

Example: No uniform citizen ID, rather use temporary keys

The exact opposite is happening right now in Germany: see Bürger-Identifikationsnummer.

Problem: possibly still reconstructible, but only with massively higher effort

Example of the use of temporary keys: The Corona Warn App

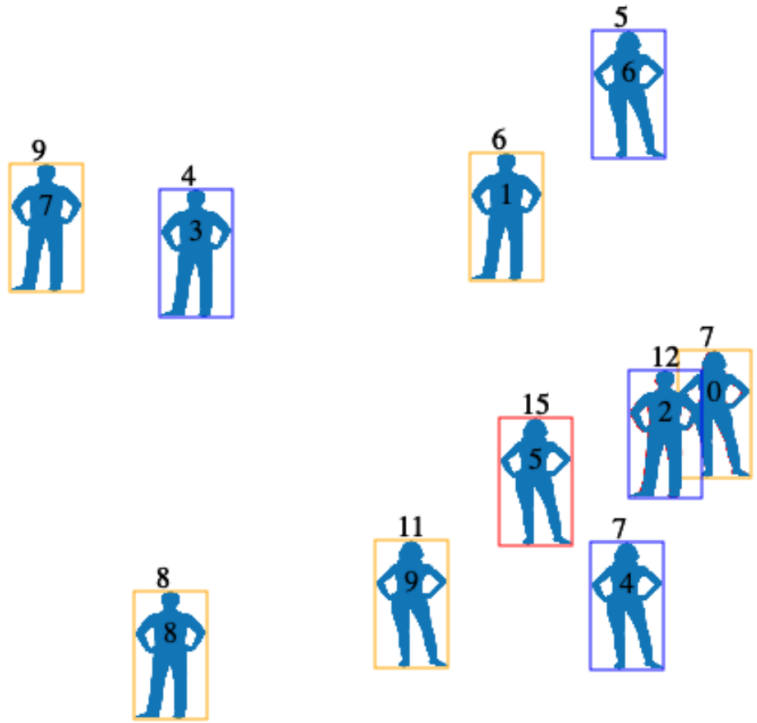
The Corona Warn App

- the Corona Warn App (official, wiki) is a contact tracing app available since 2020
- developed by SAP SE and Deutsche Telekom
- decentralised approach, developed after an initially centralised approach was discarded after heavy criticism

The Corona Warn App: Idea (Simplified)

- smartphones measure distance to other smartphones via bluetooth
- if two smartphones fall below a distance for a set period of time, both smartphones record this as an encounter
- does not use persistent, personal keys, but rather temporal keys
- every smartphone generates a new pseudo-random key every 15 minutes, only these are exchanged
- every user who has tested positive reports this to a central server and sends his keys from the last two weeks to the server
- all users regularly download the keys of all those tested positive and intersect them **locally** against their own keys to determine risk (details)

The Principle of the Corona Warn App



https://github.com/BigDataAnalyticsGroup/bigdataengineering/blob/master/CoronaWarnAppPrinciple.ipynb

Technical Approaches (4/6)

Differential Privacy, Noise in the data

add small errors to the data so that analyses are only partially falsified; analyses then (hopefully) only work for groups of tuples (k-anonymity) but no longer for identifying individual tuples

Problem: how much noise so as not to falsify analyses? Is this really safe? (see security lectures)

Change/Manipulate the data

Change data so that queries do not (no longer) show certain results

Problem: actually technically easy to discover (also in the book)

Technical Approaches (5/6)

Anti-hardware Administration (manipulate/disrupt/slow down hardware)

"unfortunately, the server is down again, I don't know what the problem is!"

Examples:

- in the book there is always the one "data silo" that is slow
- workload peaks
- water damage
- ...

Problem: also easy to discover with a little thought...

Technical Approaches (6/6)

Anti-Software Engineering (Manipulate/disrupt/slow down software)

"there's nothing we can do about it, it simply takes that long!":

Examples:

- produce unmaintainable spaghetti code (most of them do this all by ourselves anyway...)
- include hard-to-find errors (concurrency is your friend)
- write complex queries that overwhelm the query optimiser
- use the wrong database system
- configure database system incorrectly
- use NoSQL, JSON or XML
- write code that has quadratic or worse runtime:
  "Unfortunately, a join always has quadratic complexity."
- do not document the code at all or document it incorrectly

Social Approaches

- "this query is more complicated than I thought, I need more time" (used in the book)
- "it's not that simple, we should still consider aspect XY..." (used in the book)
- "we don't have the right software/hardware for it, we first have to buy XY and organise training, deploy XY, ..." (but XY is the wrong tool anyhow)
- "for this we should call in Mr X, he is an expert in this field..." (in truth Mr X is an idiot and will most likely slow down/ruin the project)
- "we should delegate this to department Y..." (they didn't get anything right in the past either...)
- "the project is too big, we should divide it into sub-projects..." (so that afterwards no one has an overview and we lose time in pointless meetings....)

Legal Approaches

Prevent joins, make access to multiple data sources difficult, no data cartels

Basically:

- strongly restrict access to "sensitive" data sources
- prevent certain types of data sources from being brought (aka joined) together, cf. Technical Approaches 3/6

Problem:

What are sensitive data sources?

- Is temporal data sensitive?
  Very likely.
- Is spatial data sensitive?
  Very likely.
- And other data less?
  Hmmmm.

The Criminal Case

The investigation

A crime was committed in X on 27.4.2019. A witness observed someone for whom a sketch was made. A query of the phantom picture to a picture database shows Mr M as a possible candidate. Mr M is registered in X. But was Mr M in X at all on 27.4. ?

- The GPS data from his (now intercepted) mobile phone says no.
- M's voice was recorded by another mobile phone, this mobile phone was located 200 km away from city X.
- The electricity and water meters of M's flat in X clearly show that someone was at home in the morning until 10 am and in the evening from 6 pm. The electricity profiles look the same as for almost every other day.
- Was it M?

Commissioner Equi-Join Starts With the Investigation

What do we know about Mr M that is suitable as a join predicate?
In other words: how can we enrich the information on Mr M?

Mr M collects sneakers

Mr M maintains a private little relation in which all his sneakers are listed.

```
[sneakers] : {[
        ID:int,
        label:str,
        size:float
]}
```

Note:

There are no time and/or place attributes in this example.

What's the big deal? It's completely harmless, isn't it?
"Mr M has nothing to hide."

A Department Store

[shoes] : {[
    ID:int,
    label:str,
    size:float
]}

[purchases] : {[
    shoe_ID:(shoes),
    amount:int,
    date:timestamp
]}

This is only a small simplified section of the scheme. Of course, the department store has information on all items sold.

The Sneaker Scenario in Python

Tracking wihout space and time attributes

This notebook showcases an example where a personal database does not contain time or space attributes. Still that data may be critical when combined with other data sources that have spatial and/or temporal attributes.


```
import duckdb
```

Define the schema and load the data of **Mr. M's** personal database:

```
# personal sneakers "database" (in fact just one tiny relation) of Mr. M:

duckdb.sql("DROP TABLE IF EXISTS sneakers;")

duckdb.sql("""
CREATE TABLE sneakers (
    id INTEGER PRIMARY KEY,
    label VARCHAR,
    size FLOAT
);""")
```

```
# load csv data into tables:
duckdb.sql("COPY sneakers FROM './data/sneakers/MrM_sneakers_no_header.csv' (FORMAT CSV, DELIMITER ',');")
```

https://github.com/BigDataAnalyticsGroup/bigdataengineering/blob/master/Sneakers.ipynb

NSA (Part 2)

Question 2

... and what ethical problems arise from these kind of data collections? How do we deal with them?

1. There are many possible countermeasures, both legal and civil disobedience.
2. In principle, the joint evaluation of several data sources should be much more strongly regulated.
3. Monitoring through metadata — which is also often processed video/audio etc data — can reveal surprising correlations.
4. These effects can be used for bad (see examples above) as well as for good (e.g. data journalism, Panama Papers, sneakers example, acquittal by data).
5. The underlying principle problem is very difficult to solve: what if data gets into the wrong hands?

Outlook for ~~Next Week~~ in two Weeks: Query Optimisation

Question 3

... and what if the data gets bigger? How do we actually get from SQL to an efficient program?

Further Material (in German and English)

- see various links in 03 NSA (Part 1)
- Edward Snowden. Permanent Record: A Memoir of a Reluctant Whistleblower. amz
- Glenn Greenwald. No Place to Hide: Edward Snowden, the NSA, and the U.S. Surveillance State. amz
- Josef Foschepoth. Überwachtes Deutschland: Post- und Telefonüberwachung in der alten Bundesrepublik. amz (in German)
- Tom Hillenbrand. Des Königs NSA: 1684 statt 1984. tomhillenbrand.de, youtube (in German)

General Bibliography for this lecture:
https://www.infomath-bib.de/tmp/vorlesungen/info-basic_big-data-engineering.html