Diffix: Enabling (Aggregate) Data Markets with Anonymization

Paul Francis

MAX PLANCK INSTITUTE
FOR SOFTWARE SYSTEMS

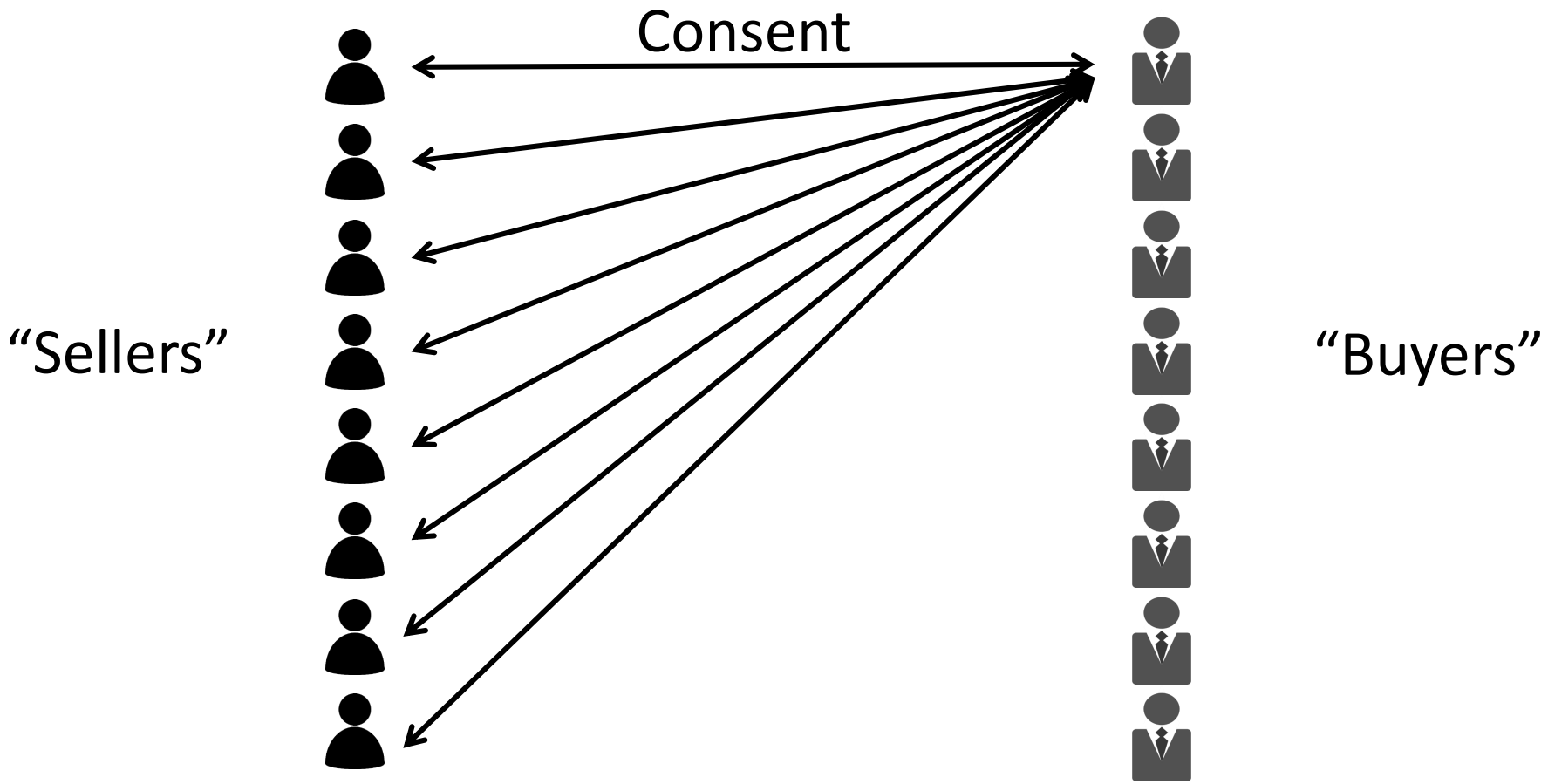
MyData 2017 Session:
Roadmap for Personal Data Markets

Markets and Scale

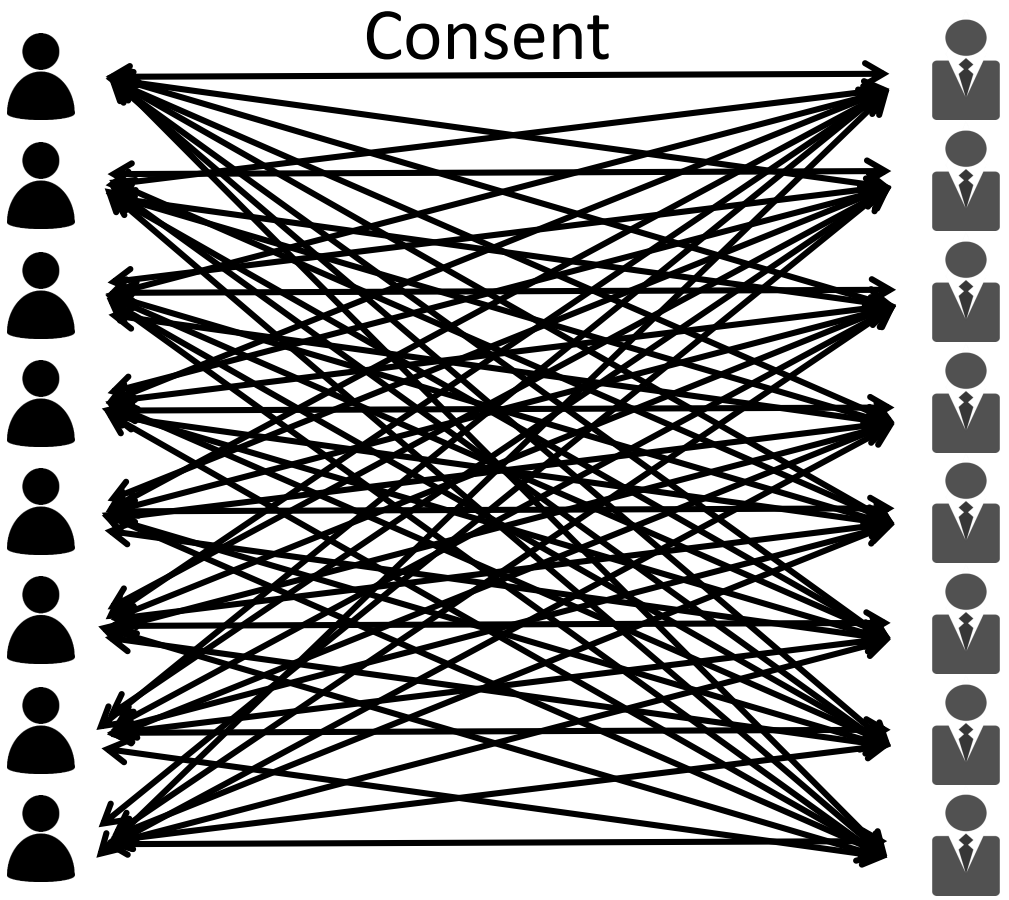
- Market: *“a regular gathering of people for the purchase and sale of provisions, livestock, and other commodities.”*

- Markets don’t scale without “middlemen” of various sorts
- A “middleman” is antithetical to the MyData concept of individual control over data
- To scale a *data market*, individual needs to give broad consent to middleman
  - Individual cannot be expected to give consent every transaction, or even track every transaction

Data Market Without Middleman



Data Market Without Middleman



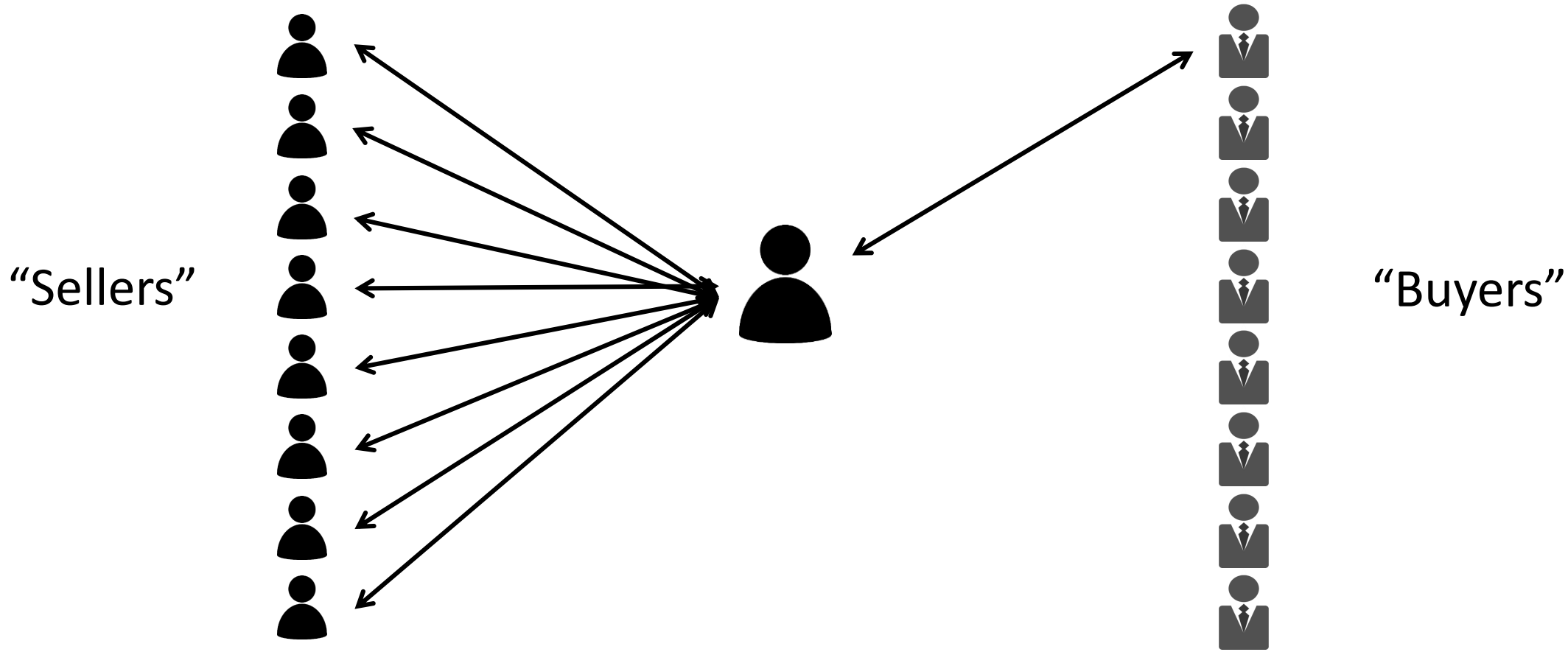
“Sellers”

“Buyers”

MAX PLANCK INSTITUTE
FOR SOFTWARE SYSTEMS

Middleman, but per-transaction consent:

- Still doesn't scale for seller

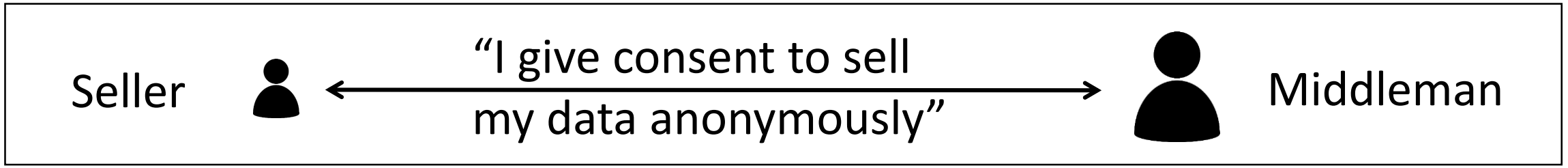


MyData Enables and Hinders Data Markets

- MyData *enables* data markets in many ways:
  - Diverse types of data joined together through identity management
  - Diverse data leads to much better analytics
    - Knowledge of diet ***and*** exercise together more valuable than knowledge of diet or exercise separately
- But MyData concept of individual control of data *hinders* data markets

Anonymization is key enabler

- Most often, data “buyer” is interested in *aggregate* data, not *individual* data
- If aggregate data is *anonymous*, then consent is much simpler:



- Because anonymous data is not “personal” data
- Safe to distribute

Conclusions from GDPR-track session in Tallinn

- People don’t trust claims of anonymity
  - Too many failures in the past

- People don’t even trust DPA “certifications” of anonymity
  - Too much expertise required, too complex

Synopsis

- Buyers in a data market may be interested in only *aggregate data*
- Historically GDPR-level anonymity very hard to achieve
- *Diffix* is a technical breakthrough in anonymity
  - GDPR-level anonymity
  - Minimal data distortion
  - Simple configuration
  - Rich query semantics

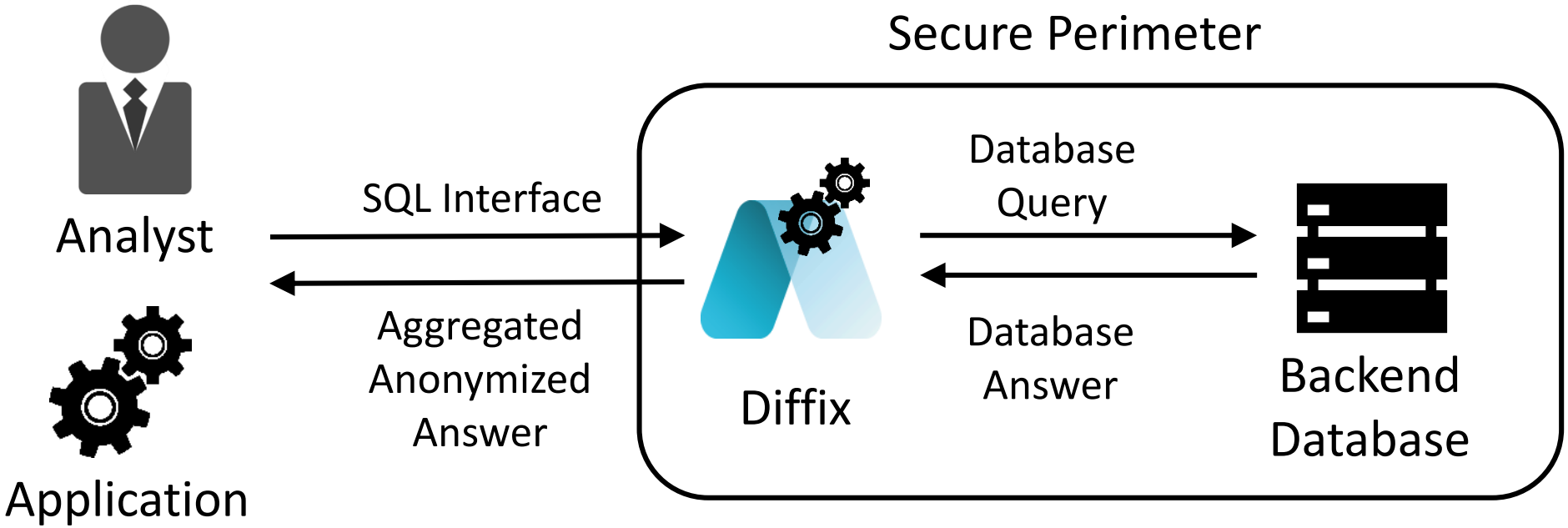
MAX PLANCK INSTITUTE
FOR SOFTWARE SYSTEMS

- Need transparency to build confidence

Strong anonymization (has been) very hard

- Requires substantial expertise
  - For example, ARX:
    - classify data as identifying, quasi-identifying, sensitive, and insensitive;
    - create masking-based, interval-based, or order-based generalization hierarchies;
    - understand and configure privacy models such as δ-presence, l-diversity, t-closeness, δ-disclosure, k-Anonymity, k-Map, (ϵ,δ)-differential privacy;
    - risk-based privacy models for prosecutor, journalist and marketer risks
- Re-think for each new use case
- *Often not possible,* even for a single data set with single use case!
- *Definately not possible* if goal is to join diverse data in a market

Diffix: Breakthrough in anonymity



Deployed as a “box” in front of an unmodified (raw) database

Diffix Configuration is Simple

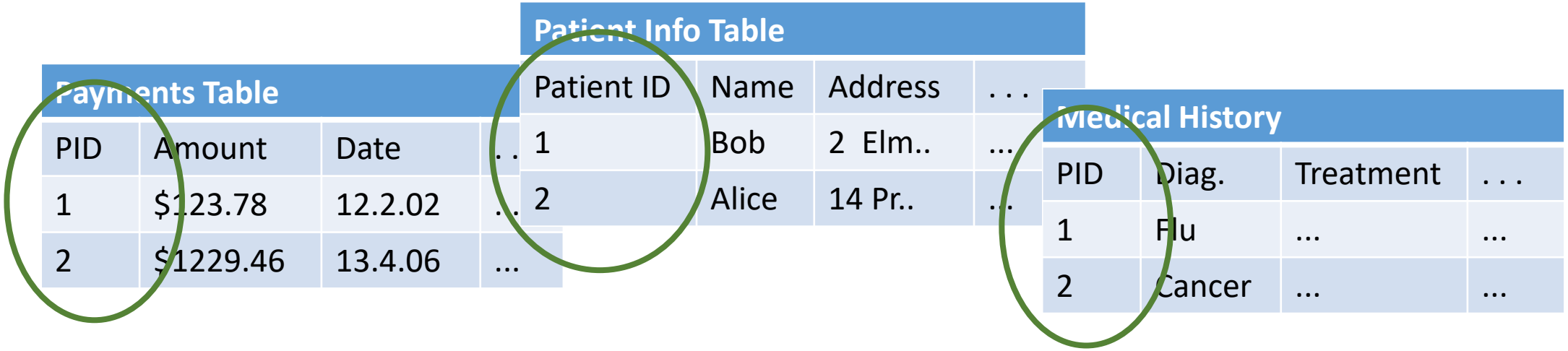
Payments Table			
PID	Amount	Date	. .
1	$123.78	12.2.02	...
2	$1229.46	13.4.06	...

Patient Info Table			
Patient ID	Name	Address	. . .
1	Bob	2 Elm..	...
2	Alice	14 Pr..	...

Medical History			
PID	Diag.	Treatment	. . .
1	Flu	...	...
2	Cancer	...	...

Diffix Configuration is Simple

- No change to database
- Simply identify and configure key fields that identify users and link user tables
- Example: 30 minutes to configure clinical database with 120 distinct user tables



Payments Table

PID	Amount	Date	. . .
1	$123.78	12.2.02	...
2	$1229.46	13.4.06	...

Patient Info Table

Patient ID	Name	Address	. . .
1	Bob	2 Elm..	...
2	Alice	14 Pr..	...

Medical History

PID	Diag.	Treatment	. . .
1	Flu	...	...
2	Cancer	...	...

```
SELECT [DISTINCT]
  field_expression [, ...]
  FROM from_expression [, ...]
  [ WHERE where_expression [AND ...] ]
  [ GROUP BY column_expression | position [, ...] ]
  [ HAVING having_expression [AND ...] ]
  [ ORDER BY column_name [ASC | DESC] | position [, ...] [ LIMIT
amount ] [ OFFSET amount ] ]

field_expression :=
  * | table_name.* | column_expression [AS alias]

column_expression :=
  [table_name.]column_name |
  aggregation_function([DISTINCT] column_name) |
  function(column_expression) |
  column_expression binary_operator column_expression |
  column_expression::data_type

binary_operator :=
  + | - | * | / | ^ | %

data_type :=
  integer | real | text | boolean | datetime | date | time

from_expression :=
  table | join

table :=
  table_name [[AS] alias] | (select_expression) [AS] alias
```

```
join :=
  table CROSS JOIN table |
  table { [INNER] | { LEFT | RIGHT } [OUTER] } JOIN table ON where_expression

aggregation_function :=
  COUNT | SUM | AVG | MIN | MAX | STDDEV | MEDIAN

where_expression :=
  column_expression equality_operator (value | column_expression) |
  column_expression inequality_operator (numerical_value | datetime_value) |
  column_expression BETWEEN value AND value |
  column_expression IS [NOT] NULL |
  column_expression [NOT] IN (constant [, ...])
  column_expression [NOT] LIKE | ILIKE string_pattern [ESCAPE escape_string]

having_expression :=
    column_expression comparison_operator (value | column_expression)

comparison_operator :=
    equality_operator | inequality_operator

equality_operator :=
    = | <>

inequality_operator :=
    > | >= | < | <=
```

Much of SQL

banking Online

```
1 SELECT left(birth_number, 2) AS birth_year, count(*), count_noise(*)
2 FROM accounts
3 GROUP BY 1
```

Run or Ctrl + Enter

```
1 SELECT left(birth_number, 2) AS birth_year, count(*), count_noise(*)
2 FROM accounts
3 GROUP BY 1
```

birth_year	count	count_noise
*	22	1.4
38	73	1.4
60	80	1.4
19	48	1.4
80	93	1.4
49	85	1.4
20	57	1.4
72	86	1.4

Tables and views New view

Filter columns

− accounts

Column	Type
account_id	integer
acct_district_id	integer
frequency	text
acct_date	integer
client_id	integer
disp_type	text
birth_number	text
cli_district_id	integer
lastname	text

+ cards

+ clients

+ disp

+ loans

+ orders

banking Online

```
1 SELECT operation, count(*), count_noise(*)
2 FROM transactions
3 GROUP BY 1
```

Run or Ctrl + Enter

```
1 SELECT operation, count(*), count_noise(*)
2 FROM transactions
3 GROUP BY 1
```

operation	count	count_noise
	218210	100
PREVOD NA UCET	254653	190
PREVOD Z UCTU	81705	58
VKLAD	182005	120
VYBER	517162	210
VYBER KARTOU	9275	34

Download as CSV | Show chart

6 rows.

Tables and views New view

Filter columns

- accounts
- cards
- clients
- disp
- loans
- orders
- transactions

Column	Type
trans_id	integer
account_id	integer
trans_date	text
trans_type	text
operation	text
amount	real
balance	real
k_symbol	text

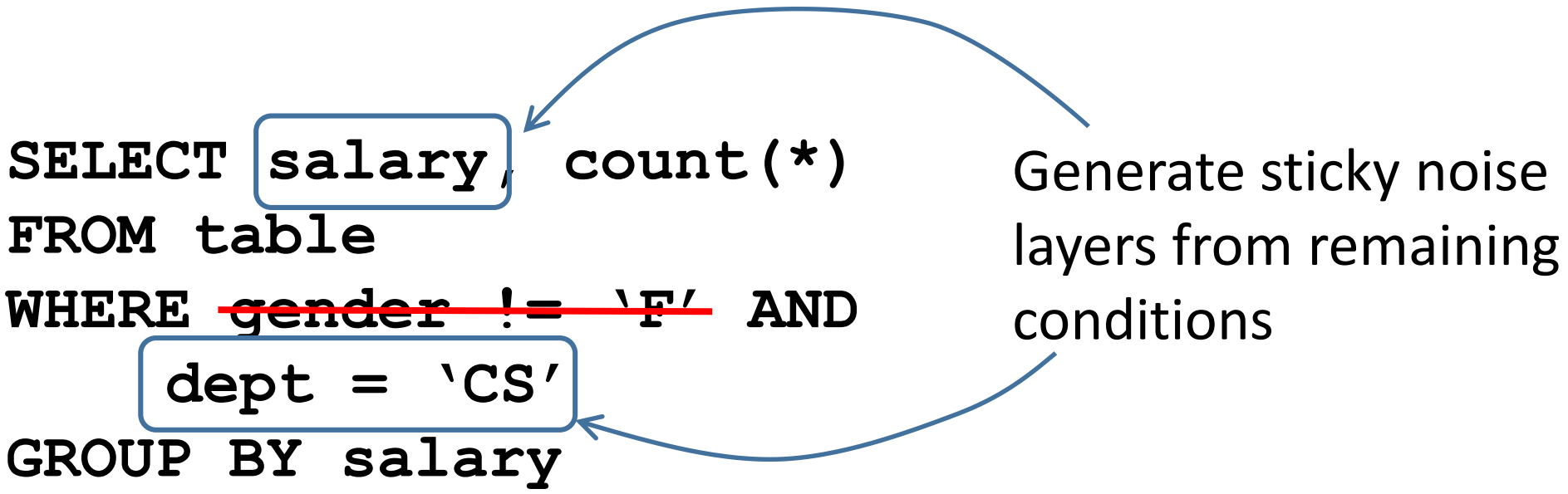
Diffix anonymization mechanisms

- Adds noise to answers
- Uses “sticky noise”
  - Prevents averaging attacks
  - No “budget” as with differential privacy
- Adds sticky noise for each data filter
- Filters answers with too few users
  - Noisy threshold
- Removes “low-effect” data filters

```
SELECT salary, count(*)
FROM table
WHERE gender != ‘F’ AND
     dept = ‘CS’
GROUP BY salary
```

```
SELECT salary, count(*)
FROM table
WHERE gender != 'F' AND
      dept = 'CS'
GROUP BY salary
```

Remove condition with minimal effect


SELECT salary, count(*)
FROM table
WHERE gender != 'F' AND
dept = 'CS'
GROUP BY salary
Generate sticky noise layers from remaining conditions

Building confidence

- Diffix is complex, not formally proven
- Positive evaluation for anonymity from CNIL
  - French national data protection authority
  - But this cannot be considered definitive
- Plan “bug bounty” type evaluation
  - Cash prizes for breaking anonymity
  - First anonymity bug bounty
  - Release in September

Thanks!

francis@mpi-sws.org

solutions@aircloak.com

MAX PLANCK INSTITUTE
FOR SOFTWARE SYSTEMS