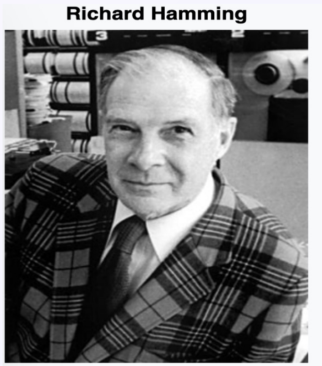
Rethinking Logical Interfaces to Data

Michael Benedikt

includes joint work over the past years with Pierre Bourhis, Louis Jachiet, and Efthymia Tsamoura and joint work with Ehud Hrushovski

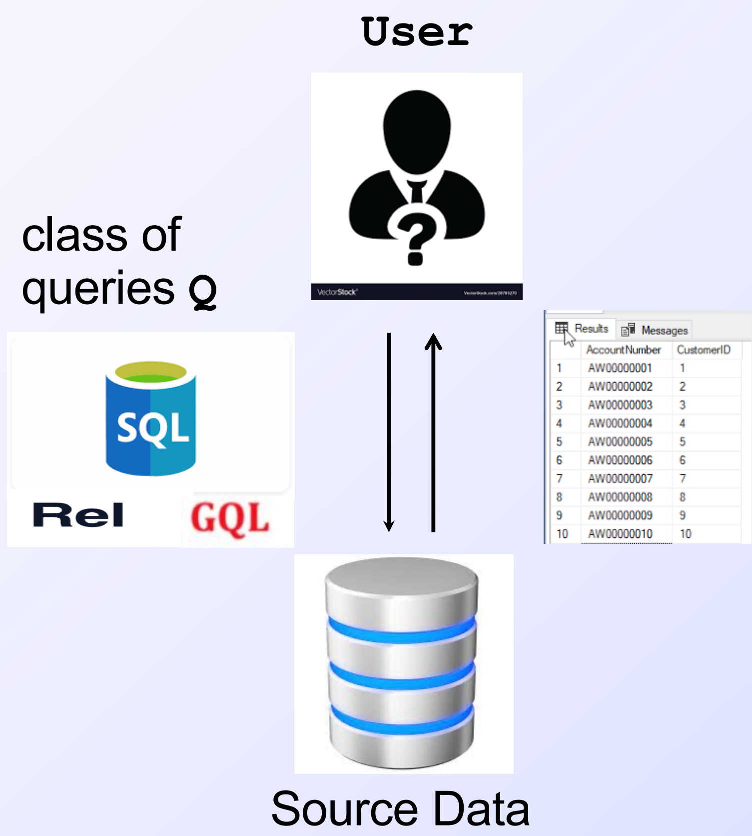
Great Thoughts Fridays.... Thursday warm up



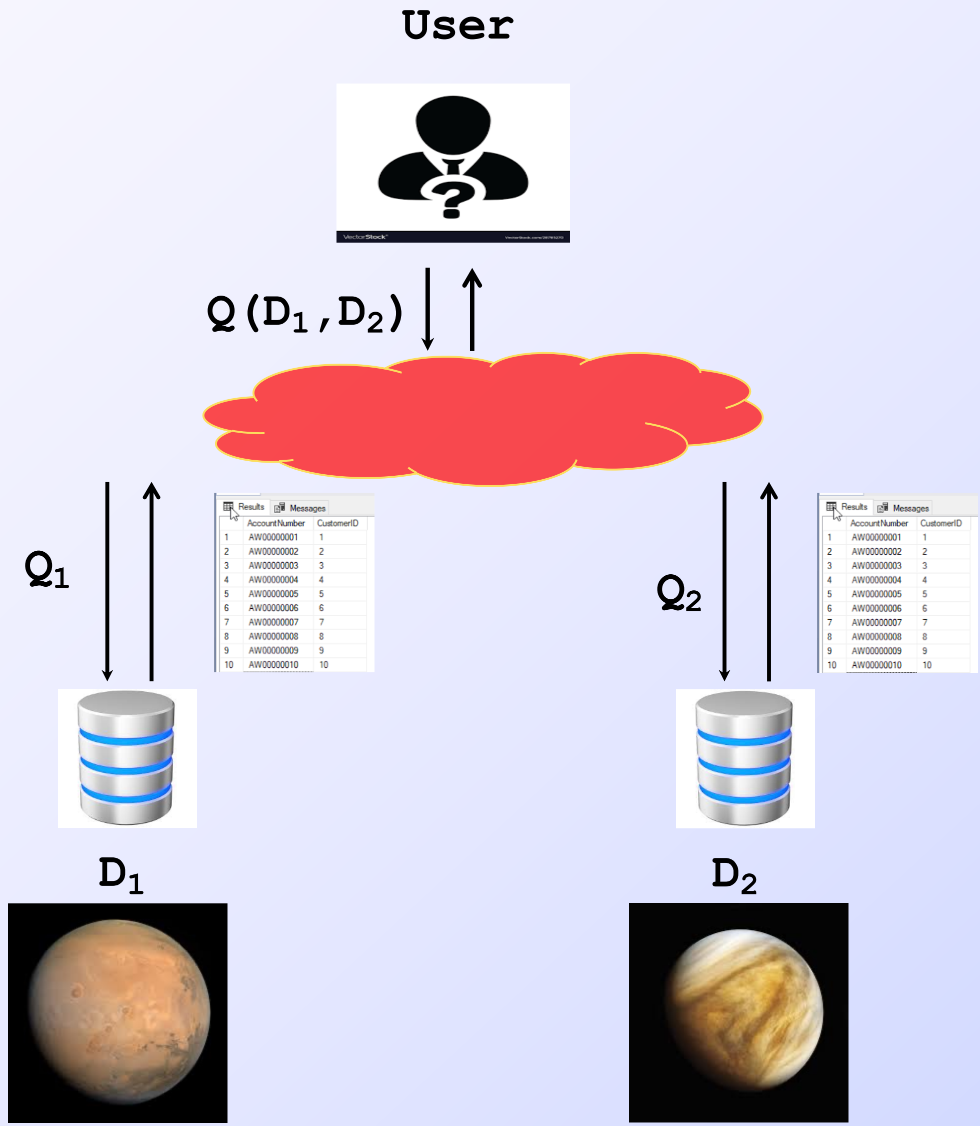
I finally adopted what I called "Great Thoughts Time". When I went to lunch Friday noon, I would only discuss great thoughts after that...

To prepare for great thoughts tomorrow, I'll present some ideas on more flexible/general interfaces to data.

Interface to data: single source



Interface to data: distributed setting



Traditional views

Traditional views are an interface based on making available derived data.

In the distributed setting, for each source $\mathbf{s}$, a view-based interface is a function $\mathbf{F_s}$ that takes as input a database instance for the schema of $\mathbf{s}$ and produces derived data; a **combination function** stitches derived data from different sources to partially/totally answer the user query.

We call the functions on the local sources ***distributed views*** (***d-views***).

Traditional views

Traditional views are an interface based on making available derived data.

In the distributed setting, for each source $\mathbf{s}$, a view-based interface is a function $\mathbf{F_s}$ that takes as input a database instance for the schema of $\mathbf{s}$ and produces derived data; a **combination function** stitches derived data from different sources to partially/totally answer the user query.

We call the functions on the local sources ***distributed views*** (***d-views***).

Views are often **definable** by logical formulas or a given query language.

Traditional views

Traditional views are an interface based on making available derived data.

In the distributed setting, for each source $\mathbf{s}$, a view-based interface is a function $\mathbf{F_s}$ that takes as input a database instance for the schema of $\mathbf{s}$ and produces derived data; a **combination function** stitches derived data from different sources to partially/totally answer the user query.

We call the functions on the local sources ***distributed views*** (***d-views***).

Views are often **definable** by logical formulas or a given query language.
E.g**.** a **conjunctive query view** over source $\mathbf{s}$:

$$\{x_1 \ldots x_m \mid \exists y_1 \ldots y_m\, A_1(x_1 \ldots y_1 \ldots) \wedge \ldots \wedge A_m(\ldots)\}$$

where $\mathbf{A_i}$ are atoms over the relations in the local source $\mathbf{s}$.

Prior interfaces beyond views

Other ways to provide a restricted interface to centralized or integrated data:

- **Access patterns**: allow access to source data, but require certain values to be specified [Chen Li and Edward Chang; Alan Nash et al.; Deutsch, Nash, Ludascher 2007]
- **Views with access patterns** [Deutsch, Nash, Ludascher 2007; Romero, Preda, Amarilli, Suchanek 2020]
- **Data Exchange/Virtual Data Integration** [Halevy; Lenzerini; Fagin, Kolaitis, Miller, Popa 2005]
- **Specification of allowed queries via automata** [Cautis, Deutsch, Onose, TOCS 2010]

Prior interfaces beyond views

Other ways to provide a restricted interface to centralized or integrated data:

- **Access patterns**: allow access to source data, but require certain values to be specified [Chen Li and Edward Chang; Alan Nash et al.; Deutsch, Nash, Ludascher 2007]
- **Views with access patterns** [Deutsch, Nash, Ludascher 2007; Romero, Preda, Amarilli, Suchanek 2020]
- **Data Exchange/Virtual Data Integration** [Halevy; Lenzerini; Fagin, Kolaitis, Miller, Popa 2005]
- **Specification of allowed queries via automata** [Cautis, Deutsch, Onose, TOCS 2010]

- **Minimal information to support a target query** [B., Bourhis, Jachiet, Tsamoura KR 2020/TODS 2022]
- **Generalizing views via indistinguishability** [B. & Hrushovski 2023]

Minimally informative query answering

We specify a set of queries $Q_1 .... Q_k$ ("utility queries") that we want to support, and ask for the **minimally informative views** (within a class) that **support these queries.**

Example

Dagstuhl and the Simons Institute want to support access to their independent datastores

Simons	Dagstuhl
`SimonsParticipant(name,program,year)`	`DagstuhlParticipant(name,progam,year)`

They want the interface to support answering some queries that span sources, like asking if there are researchers attending programs at both venues the same year.

$$Q = \exists \mathtt{program}_1\ \exists \mathtt{program}_2\ \exists \mathtt{name}\ \exists \mathtt{year}\ \mathtt{SimonsParticipant}(\mathtt{name}, \mathtt{program}_1, \mathtt{year}) \land \mathtt{DagstuhlParticipant}(\mathtt{name}, \mathtt{program}_2, \mathtt{year})$$

Example

Dagstuhl and the Simons Institute want to support access to their independent datastores

Simons	Dagstuhl
`SimonsParticipant(name,program,year)`	**`DagstuhlParticipant(name,progam,year)`**

They want the interface to support answering some queries that span sources, like asking if there are researchers attending programs at both venues the same year.

$$\mathtt{Q} = \exists \mathtt{program_1}\ \exists \mathtt{program_2}\ \exists \mathtt{name}\ \exists \mathtt{year}\ \mathtt{SimonsParticipant}(\mathtt{name}, \mathtt{program_1}, \mathtt{year}) \land \mathtt{DagstuhlParticipant}(\mathtt{name}, \mathtt{program_2}, \mathtt{year})$$

The sources should support this query, and give out the **minimal information** among **`d`**-views supporting this join query **`Q`**.

How do we formalize the notion of support and minimal information?

Example

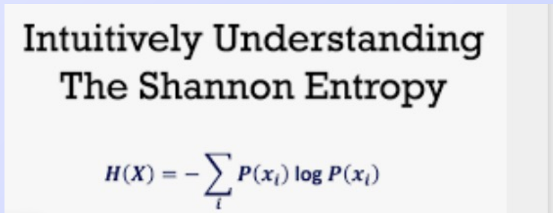
Dagstuhl and the Simons Institute want to support access to their independent datastores

Simons	Dagstuhl
`SimonsParticipant(name,program,year)`	`DagstuhlParticipant(name,progam,year)`

They want the interface to support answering some queries that span sources, like asking if there are researchers attending programs at both venues the same year.

$\mathtt{Q} = \exists \mathtt{program}_1\, \exists \mathtt{program}_2\, \exists \mathtt{name}\, \exists \mathtt{year}\; \mathtt{SimonsParticipant}(\mathtt{name}, \mathtt{program}_1, \mathtt{year}) \wedge \mathtt{DagstuhlParticipant}(\mathtt{name}, \mathtt{program}_2, \mathtt{year})$

The sources should support this query, and give out the **minimal information** among `d`-views supporting this join query `Q`.
How do we formalize the notion of support and minimal information?



Example

Dagstuhl and the Simons Institute want to support access to their independent datastores

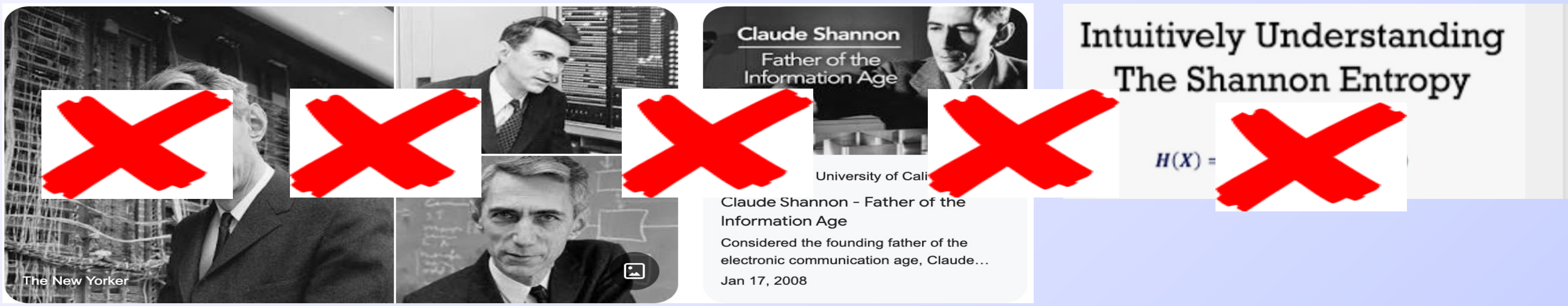
Simons	Dagstuhl
`SimonsParticipant(name,program,year)`	`DagstuhlParticipant(name,progam,year)`

They want the interface to support answering some queries that span sources, like asking if there are researchers attending programs at both venues the same year.

$\mathtt{Q} = \exists \mathtt{program}_1\ \exists \mathtt{program}_2\ \exists \mathtt{name}\ \exists \mathtt{year}\ \mathtt{SimonsParticipant}(\mathtt{name}, \mathtt{program}_1, \mathtt{year}) \wedge \mathtt{DagstuhlParticipant}(\mathtt{name}, \mathtt{program}_2, \mathtt{year})$

The sources should support this query, and give out the **minimal information** among `d`-views supporting this join query `Q`.

How do we formalize the notion of support and minimal information?



Formalization: supporting a query

What formalize the notion that the views support queries $\mathtt{Q}_1 \ldots \mathtt{Q}_u$ using Segoufin and Vianu's notion of determinacy.

Formalization: supporting a query

What formalize the notion that the views support queries Q_1 ... Q_u
using Segoufin and Vianu's notion of determinacy.

Given a query Q and views V_1 ... V_t we say Q is **determined** by V_1 ... V_t if:

for all input D, D' with $V_1(D)=V_1(D')$, ... $V_t(D)=V_t(D')$ we have $Q(D)=Q(D')$

We say that d-view V_1 ... V_t **supports** Q if Q is determined by V_1 ... V_t.

Read as "V_1 ... V_t contains all the information needed for Q"

Formalization: minimal information

We formalize the notion that the views are minimally informative using ....

Formalization: minimal information

We formalize the notion that the views are minimally informative using ....
Segoufin and Vianu's notion of determinacy.

Formalization: minimal information

We formalize the notion that the views are minimally informative using ....
Segoufin and Vianu's notion of determinacy.

We say a `d`-view `V` is a ***minimally informative supportive d-view*** for query `Q` **within a class of queries** `C` if**:**

- `V` supports `Q`
- `V` is based on queries in `C` and for every other `d`-view `V′` using queries from `C` that supports `Q`, we have `V′` determines each view in `V`

Example

Dagstuhl and Simons want to support access to their independent datastores

Simons	Dagstuhl
`SimonsParticipant(name, program, year)`	`DagstuhlParticipant(name, progam, year)`

They want the interface to support answering some queries that span sources, like asking if there are researchers who attended programs at both venues in the same year.

$$Q = \exists \mathbf{program}_1\ \exists \mathbf{program}_2\ \exists \mathbf{name}\ \exists \mathbf{year}\ \mathbf{SimonsParticipant}(\mathbf{name}, \mathbf{program}_1, \mathbf{year}) \wedge \mathbf{DagstuhlParticipant}(\mathbf{name}, \mathbf{program}_2, \mathbf{year})$$

Example

Dagstuhl and Simons want to support access to their independent datastores

Simons	Dagstuhl
SimonsParticipant(name, program, year)	**DagstuhlParticipant(name, progam, year)**

They want the interface to support answering some queries that span sources, like asking if there are researchers who attended programs at both venues in the same year.

$$Q = \exists \mathtt{program}_1\ \exists \mathtt{program}_2\ \exists \mathtt{name}\ \exists \mathtt{year}\ \mathtt{SimonsParticipant}(\mathtt{name}, \mathtt{program}_1, \mathtt{year}) \wedge \mathtt{DagstuhlParticipant}(\mathtt{name}, \mathtt{program}_2, \mathtt{year})$$

The minimal information **d**-views that support this query are the **obvious** ones:

Simons should publish the view:
$\exists \mathtt{program}\ \mathtt{SimonsParticipant}(\mathtt{name}, \mathtt{program}, \mathtt{year})$

While Dagstuhl should publish the view:
$\exists \mathtt{program}\ \mathtt{DagstuhlParticipant}(\mathtt{name}, \mathtt{program}, \mathtt{year})$

Example of our results

Theorem [B., Bourhis, Jachiet, Tsamoura] For **any** utility queries, minimally informative ɗ-views exist, and for **CQ** utility queries they are expressible as traditional views in relational algebra. The same holds in the presence of integrity constraints on each local source.

Example of our results

Theorem [B., Bourhis, Jachiet, Tsamoura] For **any** utility queries, minimally informative $\mathtt{d}$-views exist, and for **CQ** utility queries they are expressible as traditional views in relational algebra. The same holds in the presence of integrity constraints on each local source.

However, there are CQ utility queries, where the minimally informative $\mathtt{d}$-views are not CQs (and in particular, are not the obvious ones).

Example of our results

Theorem [B., Bourhis, Jachiet, Tsamoura] For **any** utility queries, minimally informative $\mathtt{d}$-views exist, and for **CQ** utility queries they are expressible as traditional views in relational algebra. The same holds in the presence of integrity constraints on each local source.

However, there are CQ utility queries, where the minimally informative $\mathtt{d}$-views are not CQs (and in particular, are not the obvious ones).

Theorem [B., Bourhis, Jachiet, Tsamoura] For any CQ utility queries, **minimally informative CQ** $\mathtt{d}$-views exist.
The same holds in the presence of integrity constraints on each local source.

Using logic-based information theory

These tools allow us to analyze trade-offs in view design.
Questions of the form “are there distributed views that support query **`Q`** but which do not reveal any information about query **`P`**”

Using logic-based information theory

These tools allow us to analyze trade-offs in view design.
Questions of the form “are there distributed views that support query **Q** but which do not reveal any information about query **p**”

R | **S,T**

$Q = \exists x\, \exists y\, R(x,y) \land S(x,y) \land T(x,y)$

Clearly, we can design views at each source to answer **Q**:
each source just exports its data.

But suppose we want to keep the following query private:
$p = \exists x\, R(x,x)$

Using logic-based information theory

These tools allow us to analyze trade-offs in view design.
Questions of the form “are there distributed views that support query $\mathbf{Q}$ but which do not reveal any information about query $\mathbf{p}$”

R **S,T**

$\mathbf{Q} = \exists \mathbf{x}\, \exists \mathbf{y}\, \mathbf{R}(\mathbf{x},\mathbf{y}) \wedge \mathbf{S}(\mathbf{x},\mathbf{y}) \wedge \mathbf{T}(\mathbf{x},\mathbf{y})$

Clearly, we can design views at each source to answer $\mathbf{Q}$:
each source just exports its data.

But suppose we want to keep the following query private:
$\mathbf{p} = \exists \mathbf{x}\, \mathbf{R}(\mathbf{x},\mathbf{x})$

Intuitively, any views (no matter what query language) that allow $\mathbf{Q}$ to be answered must disclose $\mathbf{p}$ on some instance.
Using the prior theorem, we can prove this.

Using logic-based information theory

These tools allow us to answer questions of the form “are there distributed views that support query **`Q`** but which do not reveal any information about query **`p`**”

`R,S` **`S,T`**

There is a partial synchronization between Simons and Dagstuhl:
`S` is replicated between the two sources.

Using logic-based information theory

These tools allow us to answer questions of the form “are there distributed views that support query **Q** but which do not reveal any information about query **p**”

R,S **S,T**

There is a partial synchronization between Simons and Dagstuhl:
S is replicated between the two sources.

$\mathbf{Q} = \exists \mathbf{x}\ \exists \mathbf{y}\ \mathbf{R(x,y)} \wedge \mathbf{S(x,y)} \wedge \mathbf{T(x,y)}$

Clearly, we can design views at each source to answer **Q**:
each source just exports its data.

But suppose we want to keep the following query private:
$\mathbf{p} = \exists \mathbf{x}\ \mathbf{R(x,x)}$

Using logic-based information theory

These tools allow us to answer questions of the form “are there distributed views that support query Q but which do not reveal any information about query p”

R,S

S,T

There is a partial synchronization between Simons and Dagstuhl: S is replicated between the two sources.

$Q = \exists x\, \exists y\, R(x,y) \wedge S(x,y) \wedge T(x,y)$

Clearly, we can design views at each source to answer Q: each source just exports its data.

But suppose we want to keep the following query private:

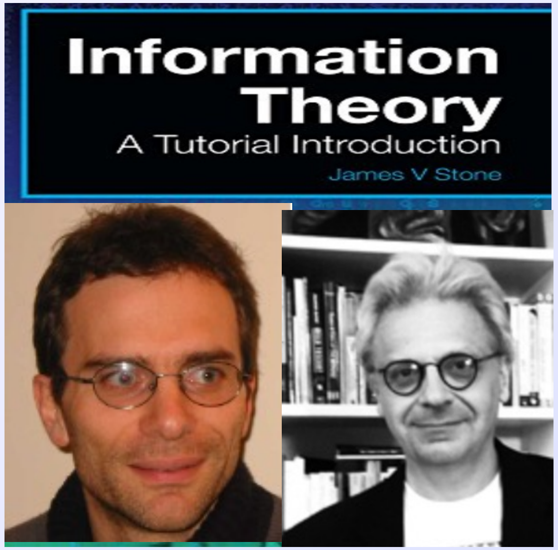
$p = \exists x\, R(x,x)$

It **is** possible to support Q without revealing p.

But we will need an interface mechanism beyond relational algebra views.

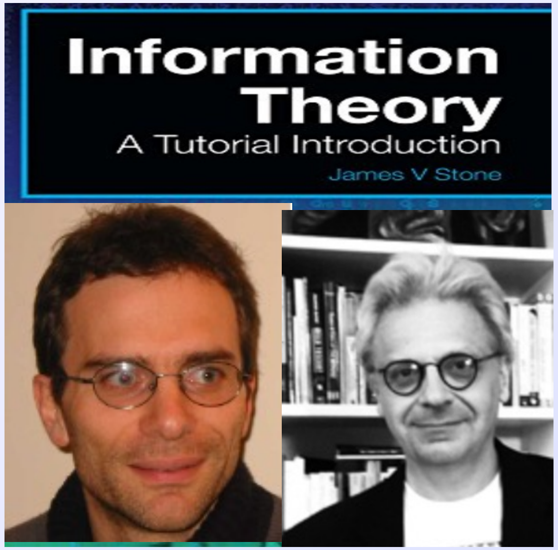
Tentative moral on minimal information querying

- Compare the expressiveness of different interface mechanisms.
- Develop the notion of determinacy from Segoufin and Vianu as a metric to perform this comparison.



Tentative moral on minimal information querying

- Compare the expressiveness of different interface mechanisms.
- Develop the notion of determinacy from Segoufin and Vianu as a metric to perform this comparison.



Also used in query pricing [Koutris, Upadhyaya, Howe, Balazinska, Suciu JACM 2015] and in other work on information disclosure [B., Bourhis, ten Cate, Puppis, Vanden Boom TOCL 2021; B., Cuenca Grau, Kostylev JAIR 2018]

Interfaces beyond views

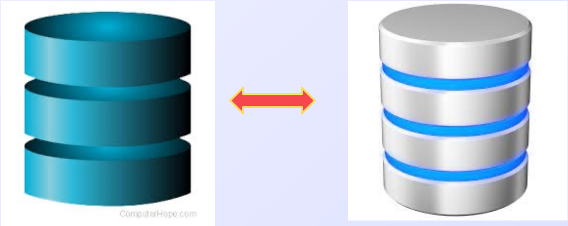
Other ways to provide restricted access to centralized or integrated data:

- **Access patterns**: allow access to source data, but require certain values to be specified [Chen Li and Edward Chang; Alan Nash et al.' Deutsch, Nash, Ludascher 2007]
- **Views with access patterns** [Deutsch, Nash, Ludascher 2007; Romero, Preda, Amarilli, Suchanek 2020]
- **Data Exchange/Virtual Data Integration** [Halevy; Lenzerini; Fagin, Kolaitis, Miller, Popa 2005]
- **Specification of allowed queries via automata** [Cautis, Deutsch, Onose, TOCS 2010]

- **Minimal information views that support a query** [B., Bourhis, Jachiet, Tsamoura KR 2020/TODS 2022]
- **Generalizing views via indistinguishability [B. & Hrushovski 2023]**

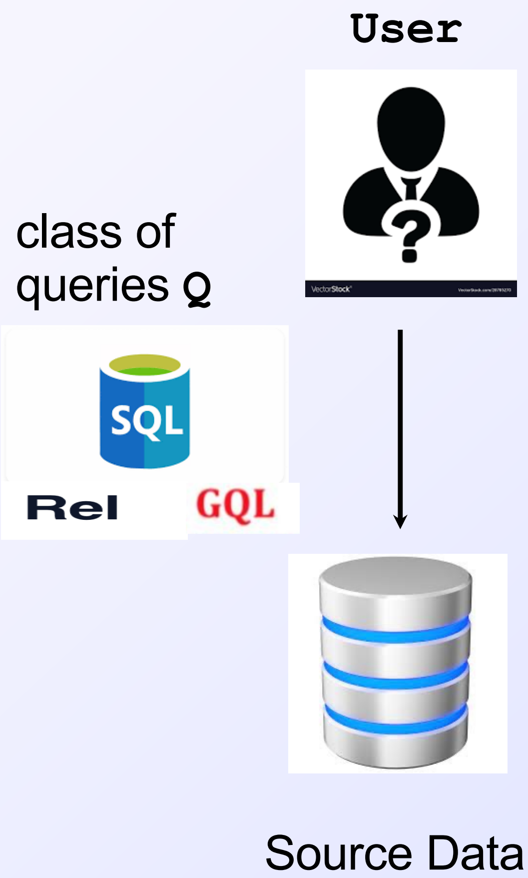
Generalizing views via database indistinguishability

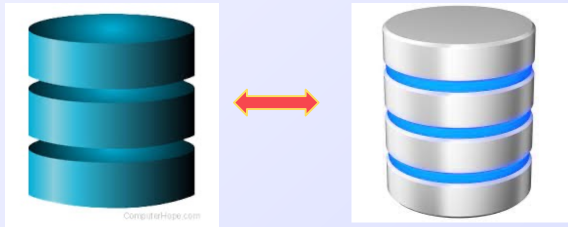
An **indistinguishability relation** is an equivalence relation on databases.
This defines an interface.



Generalizing views via database indistinguishability

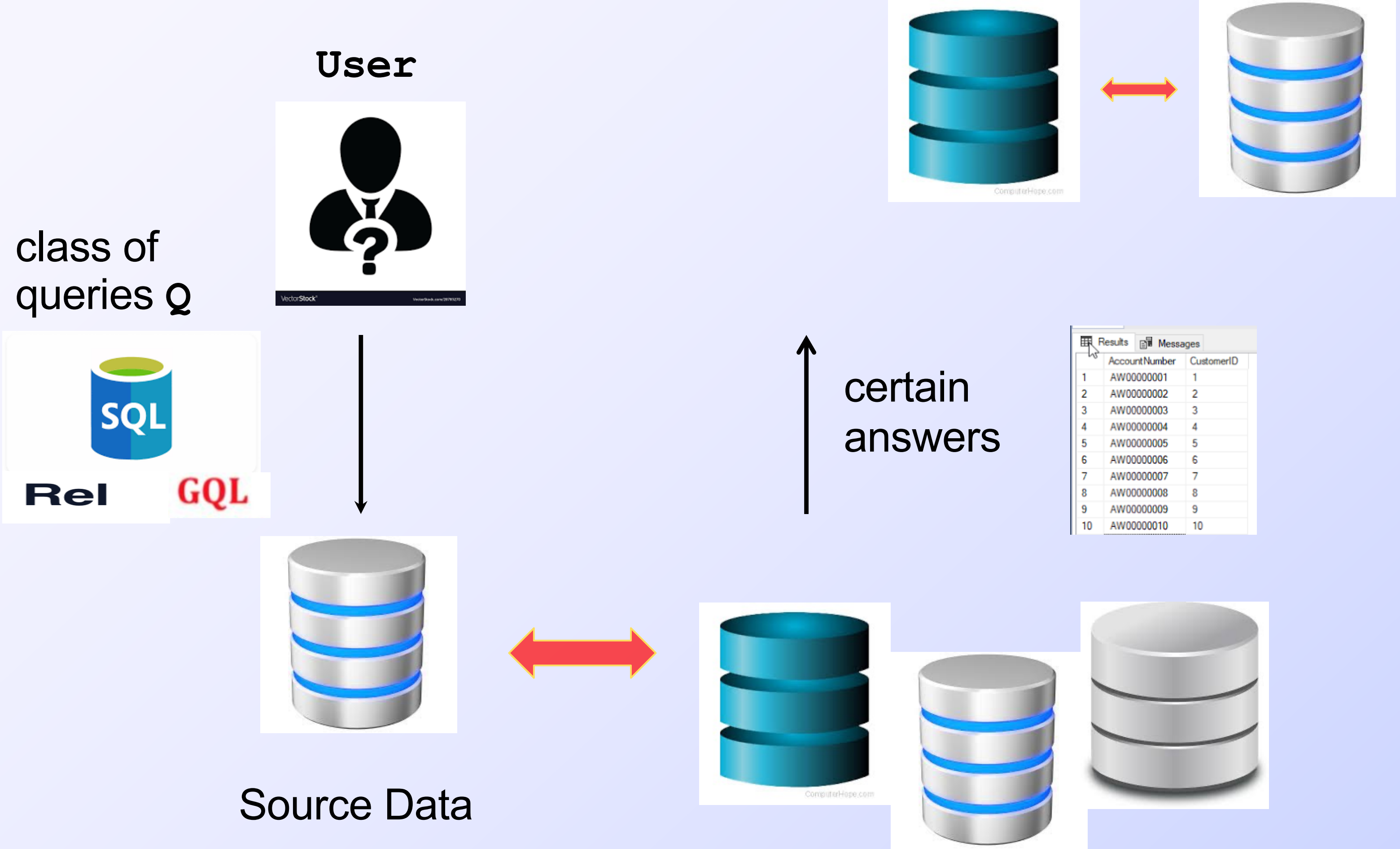
An **indistinguishability relation** is an equivalence relation on databases. This defines an interface.





Generalizing views via database indistinguishability

An **indistinguishability relation** is an equivalence relation on databases. This defines an interface.



Results
Messages

	AccountNumber	CustomerID
1	AW00000001	1
2	AW00000002	2
3	AW00000003	3
4	AW00000004	4
5	AW00000005	5
6	AW00000006	6
7	AW00000007	7
8	AW00000008	8
9	AW00000009	9
10	AW00000010	10

Defining indistinguishability with logic

An **indistinguishability relation** is an equivalence relation on databases.
It can be thought of as an “abstract view”: we are exporting the equivalence class of a database.

Defining indistinguishability with logic

An **indistinguishability relation** is an equivalence relation on databases. It can be thought of as an "abstract view": we are exporting the equivalence class of a database.

Example:

Declare graph databases $\mathbf{G}$ and $\mathbf{G'}$ indistinguishable if they have the same triangles:

$$\forall \mathbf{x}_1\, \mathbf{x}_2\, \mathbf{x}_3$$
$$[\ (G(\mathbf{x}_1, \mathbf{x}_2) \wedge G(\mathbf{x}_2, \mathbf{x}_3) \wedge G(\mathbf{x}_2, \mathbf{x}_3)) \leftrightarrow (G'(\mathbf{x}_1, \mathbf{x}_2) \wedge G'(\mathbf{x}_2, \mathbf{x}_3) \wedge G'(\mathbf{x}_2, \mathbf{x}_3))\ ]$$

Defining indistinguishability with logic

An **indistinguishability relation** is an equivalence relation on databases.
It can be thought of as an “abstract view”: we are exporting the equivalence class of a database.

Example:
Declare graph databases $\mathbf{G}$ and $\mathbf{G'}$ indistinguishable if they have the same triangles:

$$\forall \mathbf{x}_1\, \mathbf{x}_2\, \mathbf{x}_3$$
$$[\ (G(\mathbf{x}_1, \mathbf{x}_2) \wedge G(\mathbf{x}_2, \mathbf{x}_3) \wedge G(\mathbf{x}_2, \mathbf{x}_3)) \leftrightarrow (G'(\mathbf{x}_1, \mathbf{x}_2) \wedge G'(\mathbf{x}_2, \mathbf{x}_3) \wedge G'(\mathbf{x}_2, \mathbf{x}_3))\ ]$$

A **first order definable indistinguishability relation** is given by φ a first order sentence in the language of **two copies of the schema**.
Thus φ defines a collection of pairs of databases. If φ defines an equivalence relation, then φ provides an indistinguishability relation.

Note: a typical first order φ will **not** define an equivalence relation on databases. For example, transitivity will fail.

Defining indistinguishability with logic

An **indistinguishability relation** is an equivalence relation on databases.

A **first order definable indistinguishability relation** is given by φ a first order sentence in the language of **two copies of the schema** which happens to define an equivalence relation.

Class of examples of FO indistinguishability:
Traditional relational algebra views $\mathbf{V}_1 \ldots \mathbf{V}_K$ give a first order indistinguishability relation:

$$\forall \mathbf{x}_1 \ldots \mathbf{x}_j \; [\bigwedge_{i \le k} \mathbf{V}_i(\mathbf{x}_1 \ldots \mathbf{x}_j) \leftrightarrow \mathbf{V'}_i(\mathbf{x}_1 \ldots \mathbf{x}_j)]$$

where $\mathbf{V'}_i$ is a copy of $\mathbf{V}_i$ on the primed signature.

Recall: building interfaces beyond traditional views

Theorem [B., Bourhis, Jachiet, Tsamoura] For any utility queries, minimally informative $\mathfrak{d}$-views exist **.... as an indistinguishability relation**.

For CQ utility queries they are expressible as traditional views in relational algebra. The same holds in the presence of integrity constraints on each local source.

Recall: building interfaces beyond views

These tools allow us to analyze questions of the form “are there distributed views that support query `Q` but which do not reveal any information about query `p`”

`R,S` **`S,T`**

There is a partial synchronization between Simons and Dagstuhl:
`S` is replicated between the two sources.

$Q = \exists x\, y\; R(x,y) \wedge S(x,y) \wedge T(x,y)$

Clearly, we can design views at each source to answer `Q`:
each source just exports its data.

But suppose we want to keep the following query private:

$p = \exists x\; R(x,x)$

It **is** possible to support `Q` without revealing `p`.
But we will need an interface mechanism beyond relational algebra views

Recall: building interfaces beyond views

These tools allow us to analyze questions of the form "are there distributed views that support query **Q** but which do not reveal any information about query **p**"

R,S   **S,T**

SIMONS INSTITUTE for the Theory of Computing   Schloss Dagstuhl Where Computer Scientists Meet

There is a partial synchronization between Simons and Dagstuhl:
S is replicated between the two sources.

$Q = \exists x\, y\, R(x,y) \wedge S(x,y) \wedge T(x,y)$

Clearly, we can design views at each source to answer **Q**:
each source just exports its data.

But suppose we want to keep the following query private:
$p = \exists x\, R(x,x)$

It **is** possible to support **Q** without revealing **p**.
But we will need an interface mechanism beyond relational algebra views
- namely, an indistinguishability relation.

Super-generalizing views via database indistinguishability

An **indistinguishability relation** is an equivalence relation on databases.
It can be thought of as an "abstract view": we are exporting the equivalence class of a database.

This is a super-general notion.

In current work with Hrushovski we study it primarily in the setting of **classical model theory**: indistinguishability relations over **infinite structures**, focusing on relations definable in **first order and infinitary logic**.
Motivated by classification theory, descriptive set theory, model theory for topology and analysis.

But there are some results for first order indistinguishability relations on databases/finite models.

Defining indistinguishability with logic

A **first order definable indistinguishability relation** is given by φ a first order sentence in the language of **two copies of the schema** which happens to define an equivalence relation.

Recall:
Traditional relational algebra views $\mathbf{V_1} \dots \mathbf{V_K}$ give a first order indistinguishability relation:

$$\forall \mathbf{x_1} \dots \mathbf{x_j}\ [\bigwedge_{i \le k} \mathbf{V_i}(\mathbf{x_1} \dots \mathbf{x_j}) \leftrightarrow \mathbf{V'_i}(\mathbf{x_1} \dots \mathbf{x_j})]$$

Indistinguishability versus query-based views

A **first order definable indistinguishability relation** is given by φ a first order sentence in the language of **two copies of the schema** which happens to define an equivalence relation.

Traditional **nested relational calculus views** $\mathbf{V_1} \ldots \mathbf{V_K}$ give a first order indistinguishability relation.

Example: Given binary $\mathbf{R(x,y)}$, consider the view corresponding to the nested query
$\{ \{ \mathbf{y} \mid (\mathbf{x},\mathbf{y}) \in \mathbf{R} \} \mid \mathbf{x} \in \pi_1(\mathbf{R}) \}$
That is, $\mathbf{R}$ and $\mathbf{R'}$ are indistinguishable if they have the same adjacency sets of nodes.

$$\forall \mathbf{x}\ \exists \mathbf{x'} [\forall \mathbf{y}\ \mathbf{R(x,y)} \leftrightarrow \mathbf{R'(x',y)}] \wedge$$
$$\forall \mathbf{x'}\ \exists \mathbf{x} [\forall \mathbf{y}\ \mathbf{R(x,y)} \leftrightarrow \mathbf{R'(x',y)}]$$

Indistinguishibility versus query-based views

Let **E** be an "indistiguishability relation": an equivalence relation on databases. **E** can be thought of as an "abstract view".

A **first order definable indistinguishability relation** is given by φ a first order sentence in the language of **two copies of the schema**. Thus φ defines a collection of pairs of databases, and we require φ to define an equivalence relation.

Traditional **nested relational calculus views** V_1 **...** V_K give a first order indistinguishability relation.

Example: Given ternary `R(x,y,z)`, consider the view corresponding to the nested query

{ {**z** | (**x**,**y**,**z**) ∈ **R** } `\\adjacency set of x,y`

y ∈ π_2(R) } `\\set of adjacency sets for x`

| **x** ∈ π_1(R) } `\\set of sets of adjacency sets`

That is, `R` and `R′` are indistinguishable if they have the same sets of sets of adjacency sets of pairs.

Separation

Hierarchy Theorem [B., Hrushovski]
For every **`n`**, there are depth **`n`** nested relational views whose indistinguishability relation is not given by depth **`n-1`** nested relational views.

Collapse of Nested Relational Views to Relational Views

Sparse Collapse Theorem [B., Hrushovski]
Suppose **`E`** is given by a set of nested relational views on a graph database.
Suppose **`C`** is a collection of graphs that exclude a minor.
Then over **`C`**, **`E`** is given by a set of relational algebra views.

Prefix Classes of FO Indistinguishability Relations

Can classify FO Indistinguishability relations by the quantifier alternation, focusing only on the quantified variables that vary over both models.

E.g. Triangle-based equivalence example is Π_1 : $\forall \mathbf{x}_1\ \mathbf{x}_2\ \mathbf{x}_3\ \ldots$
The first nested relational calculus example (adjacency sets) is Π_3 :
$\forall \mathbf{x}_1 \ldots \exists\ \mathbf{y}_1 \ldots \forall \mathbf{z}_1 \ldots$

Prefix Classes of FO Indistinguishability Relations

Can classify FO indistinguishability relations by the quantifier alternation, focusing only on the quantified variables that vary over both models.

E.g. Triangle-based equivalence example is Π_1 : $\forall \mathbf{x}_1\ \mathbf{x}_2\ \mathbf{x}_3 \ldots$
The first nested relational calculus example (adjacency sets) is Π_3 :
$\forall \mathbf{x}_1 \ldots \exists\ \mathbf{y}_1 \ldots \forall \mathbf{z}_1 \ldots$

Π_2 Theorem [B., Hrushovski]
Suppose $\mathbf{E}$ is a Π_2 indistinguishability relation: given by a $\forall \mathbf{x}_1 \ldots \exists\ \mathbf{y}_1 \ldots$ sentence in two copies of the signature, that happens to define an equivalence relation, showing here only the quantifiers of variables that span both models.
Then $\mathbf{E}$ is a Π_1 indistinguishability relation: given by a universal sentence.

First Order Indistinguishability and Nested Relations

Question: Is every first order indistinguishability relation is given by nested relational calculus views?

Tentative Moral on Indistinguishability Relations

- Indistinguishability relations make the world of traditional view interfaces look very small

- Issue of converting between interface specifications of different natures.

In this case, from a compactedly-represented equivalence class to a canonical representative.

Many analogies in descriptive complexity theory and descriptive set theory.

Lead-In To Great Thoughts Friday

1970

1980

1990

2000

2010

2020

Relational databases have been around for over 50 years.

And in the first 40 years, the notion of logical interface today and notions of comparing interfaces were frequently revisited, often radically so.

Lead-In To Great Thoughts Friday

1970

1980

1990

2000

2010

2020






A Relational Model of Data for Large Shared Data Banks

E. F. Codd
IBM Research Laboratory, San Jose, California

The format model: A theory of database organization

Richard Hull, Chee K. Yap

The Information Manifold

Thomas Kirk	Alon Y. Levy	Yehoshua Sagiv	Divesh Srivastava
AT&T Bell Laboratories	AT&T Bell Laboratories	Hebrew University	AT&T Bell Laboratories
tk@research.att.com	levy@research.att.com	sagiv@cs.huji.ac.il	divesh@research.att.com

Data exchange: semantics and query answering ☆

Ronald Fagin[a], Phokion G. Kolaitis[b 1], Renée J. Miller[c 1], Lucian Popa[a]

Sharing Distributed Knowledge on the Web

Serge Abiteboul

Collège de France, INRIA Saclay & ENS Cachan, France

Lead-In To Great Thoughts Friday

1970

1980

1990

2000

2010

2020

A Relational Model of Data for Large Shared Data Banks

E. F. Codd
IBM Research Laboratory, San Jose, California

The format model: A theory of database organization

Richard Hull, Chee K. Yap

The Information Manifold

Thomas Kirk
AT&T Bell Laboratories
tk@research.att.com

Alon Y. Levy
AT&T Bell Laboratories
levy@research.att.com

Yehoshua Sagiv
Hebrew University
sagiv@cs.huji.ac.il

Divesh Srivastava
AT&T Bell Laboratories
divesh@research.att.com

Data exchange: semantics and query answering ☆

Ronald Fagin [a], Phokion G. Kolaitis [b 1], Renée J. Miller [c 1], Lucian Popa [a]

Sharing Distributed Knowledge on the Web

Serge Abiteboul
Collège de France, INRIA Saclay & ENS Cachan, France

?

When you need more complicated views

R **S**

Query to support specified as

$$Q = \exists\, x\, y\ R(x,y) \wedge S(x,y) \wedge S(y,x)$$

When you need more complicated views

R | **S**

Query to support specified as

$Q = \exists\, x\, y\; R(x,y) \wedge S(x,y) \wedge S(y,x)$

Minimal information supporting view at the Simons source:

$S(x,y) \vee S(y,x)$