Relational Database Encryption

Zheguang Zhao

Computer Science Department, Brown University, Rhode Island
zheguang.zhao@gmail.com

Thesis Proposal, 12th December, 2019

Part I

Introduction

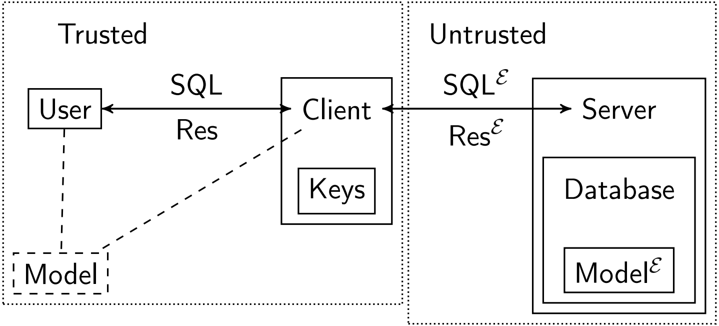
Introduction

1. The Problem
2. Background
3. Conceptual Approach

3 Problems

- Signal Problem: Secure application on untrusted OS
- Equifax Problem: Data within the private cloud
- Amazon Problem: Applications running on the public cloud

Fundamental Problem: Relational Database Encryption



- Security
- Functionality
- Efficiency
- Legacy compliance

Related Works

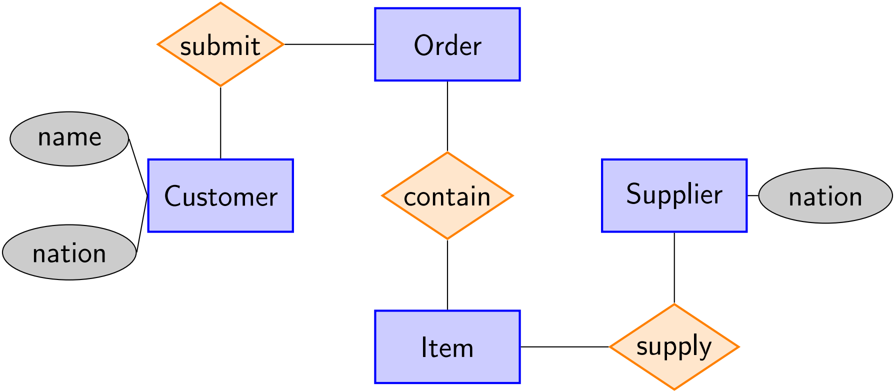
Scheme	Leak.	Eff.	Legacy	Func.
FHE [14]	Less	Low	No	IntComp.
ORAM [11]	Less	Low	No	EqSel,Prefix,Range
PPE [6]	More	High	Y/N	$\subseteq$SQL(?)
STE [7, 10, 12]	Less	High	No	KeyVal,EqSel,EqJn,Prefix,Range

Table: General observation in schemes based on various symmetric searchable encryption primitives.

Thesis Problem

Use STE to construct secure, legacy-compliant, functional and efficient relational database encryption scheme.

Relational Model



- Contains entities and relationships
- First-order logic

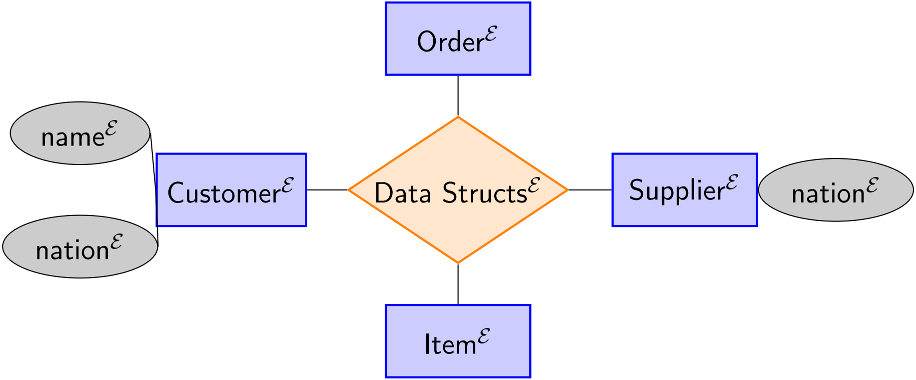
Cryptographic Primitives

- Cryptographic hash function $\mathcal{H}$
- Pseudorandom function $\mathcal{F}$
- Pseudorandom permutation $\mathcal{P}$
- Psuedorandom ciphertext under chosen-plaintext attack (RCPA-secure symmetric encryption $\mathcal{E}$)
- Switching Lemma ($\mathcal{F} \approx \mathcal{P}$)

Introduction

1. The Problem
2. Background
3. Conceptual Approach

Encrypted Data Structures



Security against Leakage

The scheme $\mathbf{dex} = (\mathbf{Setup}, \mathbf{ProcessQuery})$ is $\mathcal{L}$-secure if $\forall$ efficient A $\exists$ efficient S such that

$$\left| \Pr\left(\mathsf{Real}^{A}_{\mathbf{dex}}(\lambda) = 1\right) - \Pr\left(\mathsf{Ideal}^{A}_{\mathbf{dex},\mathcal{L},S}(\lambda) = 1\right) \right| \leq \mathsf{negl}(\lambda)$$

$\mathbf{Real}_{\mathbf{dex}}(\lambda)$	$\mathbf{Ideal}_{\mathbf{dex},\mathcal{L},S}(\lambda)$
$\mathbf{Init}(\mathsf{RDB})$ 1. $(\mathsf{ERDB}, k) \sim \mathbf{Setup}(\mathsf{RDB}, \lambda)$ 2. **return** ERDB $\mathbf{Query}(Q_{\mathsf{RDB}})$ 1. $(\mathsf{Res}, Q_{\mathsf{ERDB}}) \sim \mathbf{ProessQuery}(k, Q_{\mathsf{RDB}})$ on ERDB 2. **return** Q_{ERDB} $\mathbf{Update}(Q_{\mathsf{RDB}})$ 1. $(\mathsf{Res}, Q_{\mathsf{ERDB}}) \sim \mathbf{ProcessQuery}(k, Q_{\mathsf{RDB}})$ on ERDB 2. **return** ERDB $\mathbf{Final}(b)$ 1. **output** b	$\mathbf{Init}(\mathsf{RDB})$ 1. $L \sim \mathcal{L}(\mathsf{RDB})$ 2. $\mathsf{ERDB} \sim S_{\mathbf{Init}}(L)$ 3. **return** ERDB $\mathbf{Query}(Q_{\mathsf{RDB}})$ 1. $L \sim \mathcal{L}(L, Q_{\mathsf{RDB}})$ 2. $(\mathsf{Res}, Q_{\mathsf{ERDB}}) \sim S_{\mathbf{Query}}(L)$ 3. **return** Q_{ERDB} $\mathbf{Update}(Q_{\mathsf{RDB}})$ 1. $L \sim \mathcal{L}(L, Q_{\mathsf{RDB}})$ 2. $(\mathsf{Res}, Q_{\mathsf{ERDB}}) \sim S_{\mathbf{Update}}(L)$ 3. **return** Q_{ERDB} $\mathbf{Final}(b)$ 1. **output** b

Part II

Construction

Construction

Emulation

- Express data structure & computation in domain-specific language

---

[1] Or with transitive closure and windowing function

Emulation

- Express data structure & computation in domain-specific language
- Varying expressibility
  1. SPC algebra: conjunctive queries
  2. Relational algebra: $\subseteq$first-order logic
  3. Datalog: least fixed-point logic
  4. SQL: Turing-complete with procedral extension[1]

[1] Or with transitive closure and windowing function

Emulation

- Express data structure & computation in domain-specific language
- Varying expressibility
  1. SPC algebra: conjunctive queries
  2. Relational algebra: $\subseteq$first-order logic
  3. Datalog: least fixed-point logic
  4. SQL: Turing-complete with procedral extension[1]
- Examples
  - Datalog, SQLite: no procedures.
  - Azure SQL Data Warehouse, SparkSQL: no transitive closure.

[1] Or with transitive closure and windowing function

Emulation

- Express data structure & computation in domain-specific language
- Varying expressibility
  1. SPC algebra: conjunctive queries
  2. Relational algebra: $\subseteq$first-order logic
  3. Datalog: least fixed-point logic
  4. SQL: Turing-complete with procedral extension[1]
- Examples
  - Datalog, SQLite: no procedures.
  - Azure SQL Data Warehouse, SparkSQL: no transitive closure.

Problem

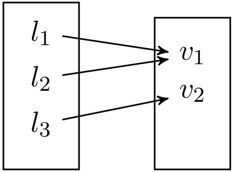
Minimum language for expressing an (encryted) data structure

[1] Or with transitive closure and windowing function

Map

A map is a binary relation of labels and values
M : $\{(l \rightarrow v) \mid l \in L, v \in V\}$ which associates each label to a unique value.
The operation $\mathsf{M}[l] = v$ if $(l \rightarrow v) \in \mathsf{M}$.

Example:



$\mathsf{M}[l_2] = v_1$

Emulation:

label	value
l_1	v_1
l_2	v_1
l_3	v_2

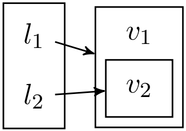
$$\pi_{\texttt{value}}\sigma_{\texttt{label}=l_2}T_{\mathsf{M}} \implies [v_1]$$

Multi-Map

Multi-Map

A multi-map is a map[1] between labels and sets of values $\mathsf{MM} : \{(l \to V_l \mid l \in L, V_l \subseteq V)\}$. The operation $\mathsf{MM}[l] = V_l$ if $(l \to V_l) \in \mathsf{MM}$.

Example:



$\mathsf{MM}[l_1] = \{v_1, v_2\}$

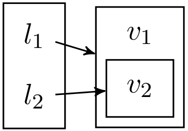
[1] Elsewhere the term "map" will be reserved to a map that is *not* a multi-map.

Multi-Map

Multi-Map

A multi-map is a map[1] between labels and sets of values MM : $\{(l \to V_l \mid l \in L, V_l \subseteq V)\}$. The operation MM$[l] = V_l$ if $(l \to V_l) \in$ MM.

Example:



MM$[l_1] = \{v_1, v_2\}$

Emulation Idea 1: Padding

`label`	`value`
l_1	v_1, v_2
l_2	v_2

Bad: Set of sets is unrelational.

[1] Elsewhere the term "map" will be reserved to a map that is *not* a multi-map.

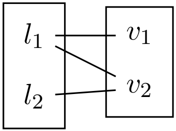
Emulation of Idea 2: Flattening

`label`	`value`
l_1	v_1
l_1	v_2
l_2	v_2

$$\pi_{\texttt{value}}\sigma_{\texttt{label}=l_1}T_{\mathsf{MM}} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

Multi-Map

Idea 2: Flattening



Bad: No longer a map[1].

Emulation of Idea 2: Flattening

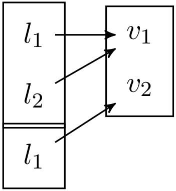
`label`	`value`
l_1	v_1
l_1	v_2
l_2	v_2

$$\pi_{\texttt{value}}\sigma_{\texttt{label}=l_1}T_{\mathsf{MM}} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

[1]However it meets a non-map definition of multi-map.

Multi-Map

Idea 3a: two maps

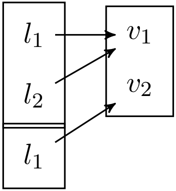


Multi-Map

Label transformation

$f : \mathbb{Z}^+ \times L \to L'$ is a one-to-one function.

Idea 3a: two maps

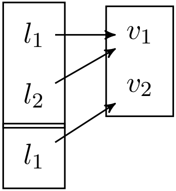


Multi-Map

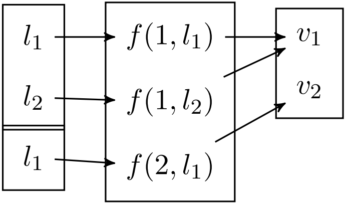
Label transformation

$f : \mathbb{Z}^+ \times L \to L'$ is a one-to-one function.

Idea 3a: two maps



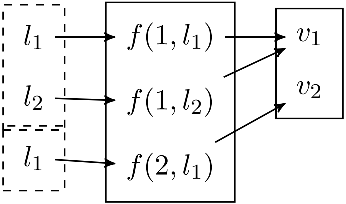
Idea 3b: three maps



The i in $f(i, l)$ means the ith map of $L \to L'$.

Multi-Map

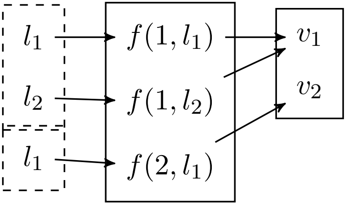
- Knowing f, reduce to one map
- Idea 3c: One map



Multi-Map

- Knowing f, reduce to one map
- Emulate as table T_{MM}

- Idea 3c: One map



- T_{MM} for M : $f \rightarrow V$

`label`	`value`
$f(1, l_1)$	v_1
$f(1, l_2)$	v_2
$f(2, l_1)$	v_2

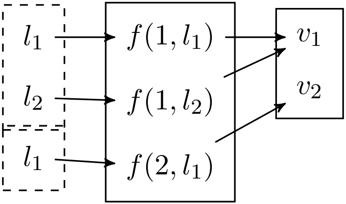
Multi-Map

- Knowing f, reduce to one map
- Emulate as table T_{MM}
- Express $\text{MM}[l]$ as union

$$\text{MM}[l_1] \to \{\text{M}[f(1, l_1)]\} \cup \{\text{M}[f(2, l_1)]\}$$

$$\text{MM}[l_2] \to \{\text{M}[f(1, l_2)]\}$$

- Idea 3c: One map



- T_{MM} for $\text{M} : f \to V$

`label`	`value`
$f(1, l_1)$	v_1
$f(1, l_2)$	v_2
$f(2, l_1)$	v_2

Multi-Map

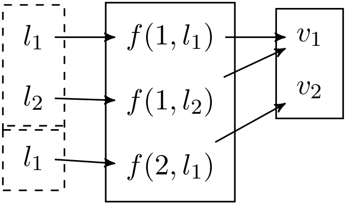
- Knowing f, reduce to one map
- Emulate as table T_{MM}
- Express $\mathsf{MM}[l]$ as union

$$\mathsf{MM}[l_1] \to \{\mathsf{M}[f(1,l_1)]\} \cup \{\mathsf{M}[f(2,l_1)]\}$$

$$\mathsf{MM}[l_2] \to \{\mathsf{M}[f(1,l_2)]\}$$

- Is iteration a viable idea?
- In relational algebra?

- Idea 3c: One map



- T_{MM} for $\mathsf{M} : f \to V$

label	value
$f(1,l_1)$	v_1
$f(1,l_2)$	v_2
$f(2,l_1)$	v_2

Multi-Map

Multi-Map as Recursive Map

A multi-map MM : $L \to 2^V$ is a map M : $f(\mathbb{Z}^+, L) \to V$ where f is a one-to-one function. The operation $\mathsf{MM}[l] = \mathsf{rec}(1, l)$ where

$$\mathsf{rec}(i,l) = \begin{cases} \{\mathsf{M}[f(i,l)]\} \cup \mathsf{rec}(i+1,l) & \text{if} \quad \exists f(i,l) \in \mathrm{Dom}(\mathsf{M}) \\ \emptyset & \text{else} \end{cases}$$

Multi-Map

Multi-Map as Recursive Map

A multi-map $\mathsf{MM} : L \to 2^V$ is a map $\mathsf{M} : f(\mathbb{Z}^+, L) \to V$ where f is a one-to-one function. The operation $\mathsf{MM}[l] = \mathsf{rec}(1, l)$ where

$$\mathsf{rec}(i, l) = \begin{cases} \{\mathsf{M}[f(i, l)]\} \cup \mathsf{rec}(i + 1, l) & \text{if} \quad \exists f(i, l) \in \mathrm{Dom}(\mathsf{M}) \\ \emptyset & \text{else} \end{cases}$$

Negative Result

Relational algebra cannot express the multi-map as the recursive map.

Relational Algebra with Transitive Closure

- Transitive closure is needed
- Undecidablility impacts query optimization (Later).

Syntax of RCTE

Transitive closure can be expressed throuh the recursive common table expression (RCTE) as in SQL-99 [3]

```
With Recursive View As
    BaseSubquery
  Union All
    RecursiveSubquery
```

Relational Algebra with Transitive Closure

- But the semantics of RCTE has only ad-hoc definition in the literature. Here we provide an algebraic one.

Semantics of RCTE

The semantics of the RCTE defines that the recursive view is equivalent to

$$\mathsf{View} = \mathsf{rec}_{\mathsf{rcte}}(1)$$

where the recursive function is defined as

$$\mathsf{rec}_{\mathsf{rcte}}(i) = \begin{cases} \Delta\mathsf{View}_i \cup \mathsf{rec}_{\mathsf{rcte}}(i+1) & \text{if} \quad \Delta\mathsf{View}_i \neq \emptyset \\ \emptyset & \text{else} \end{cases}$$

with the view increment

$$\Delta\mathsf{View}_i = \begin{cases} \mathsf{BaseSubquery} & \text{if} \quad i = 1 \\ \mathsf{RecursiveSubquery} \mid \Delta\mathsf{View}_{i-1} & \text{else} \end{cases}$$

Relational Algebra with Transitive Closure

- The RCTE semantics can express the recursive map. Why?
- Simply compare

$$\mathsf{rec}(i,l) = \begin{cases} \{\mathsf{M}[f(i,l)]\} \cup \mathsf{rec}(i+1,l) & \text{if} \quad \exists f(i,l) \in \mathrm{Dom}(\mathsf{M}) \\ \emptyset & \text{else} \end{cases}$$

and

$$\mathsf{rec}_{\mathsf{rcte}}(i) = \begin{cases} \Delta\mathsf{View}_i \cup \mathsf{rec}_{\mathsf{rcte}}(i+1) & \text{if} \quad \Delta\mathsf{View}_i \neq \emptyset \\ \emptyset & \text{else} \end{cases}$$

Multi-Map Emulation

T_{MM} for rec. map

`label`	`value`
$f(1, l_1)$	v_1
$f(1, l_2)$	v_2
$f(2, l_1)$	v_2

Multi-Map Emulation

- Operation MM$[l]$ is emulated by $\mathbf{RCTE}(l)$ as

With Recursive View As

$T_{\text{MM}} \bowtie_{\texttt{label}=f(i,l)} \{\langle i : 1 \rangle\}$

Union All

$T_{\text{MM}} \bowtie_{\texttt{label}=f(i,l)} \pi_{i+1 \to i}\text{View}$

$\pi_{\texttt{value}}\text{View}$

T_{MM} for rec. map

label	value
$f(1, l_1)$	v_1
$f(1, l_2)$	v_2
$f(2, l_1)$	v_2

Multi-Map Emulation

T_{MM} for rec. map

label	value
$f(1, l_1)$	v_1
$f(1, l_2)$	v_2
$f(2, l_1)$	v_2

- Operation $\mathsf{MM}[l]$ is emulated by $\mathbf{RCTE}(l)$ as

$$
\begin{array}{l}
\texttt{With Recursive}\ \ \mathsf{View}\ \ \texttt{As} \\
\qquad T_{\mathsf{MM}} \bowtie_{\texttt{label}=f(i,l)} \{\langle i : 1\rangle\} \\
\quad \texttt{Union All} \\
\qquad T_{\mathsf{MM}} \bowtie_{\texttt{label}=f(i,l)} \pi_{i+1\to i}\mathsf{View} \\
\pi_{\texttt{value}}\mathsf{View}
\end{array}
$$

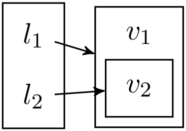
- $\mathbf{RCTE}[l_1]$:

$$\{i = 1, f(1, l_1), v_1\} \cup \{i = 2, f(2, l_1), v_2\} \cup \{i = \cancelto{\emptyset}{3, f(3, l_1)}, \mathsf{nil}\}$$

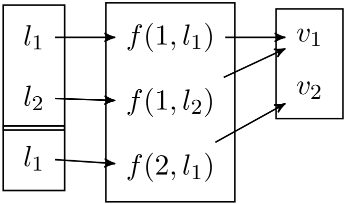
- Projects as $\begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$

Construction

- Multi-Map



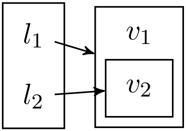
- Recursive map



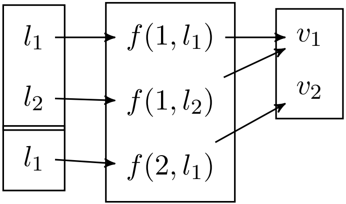
Encrypted Multi-Map

- Correct:
  $\Pr(\mathsf{MM}[l] \equiv \mathsf{EMM}[l]) = 1 - \mathsf{negl}(\lambda)$

- Multi-Map



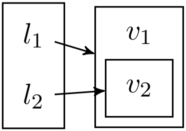
- Recursive map



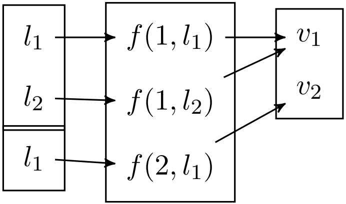
Encrypted Multi-Map

- Correct: $\Pr(\mathsf{MM}[l] \equiv \mathsf{EMM}[l]) = 1 - \mathsf{negl}(\lambda)$
- Leakage: # edges, queries, results

- Multi-Map



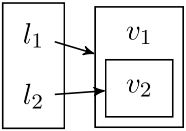
- Recursive map



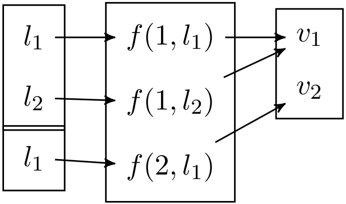
Encrypted Multi-Map

- Correct: $\Pr(\mathsf{MM}[l] \equiv \mathsf{EMM}[l]) = 1 - \mathsf{negl}(\lambda)$
- Leakage: # edges, queries, results
- Structure: recursive map homomorphism
  - perspective on correctness and security
  - for emulation

- Multi-Map



- Recursive map



Encrypted Multi-Map

Π_{bas} Encrypted Multi-Map

The scheme Π_{bas} [8] can be seen as a transformation of a recursive map representation of MM

- Client: $l' = \mathcal{F}(k, l)$, $\text{trpd}_j(l') = \mathcal{F}(k, l'||j)$[1] , $\mathcal{E}(\cdot, \cdot)$
- Server: $\mathcal{F}_s(\text{trpd}_1, i)$, $\mathcal{D}(\text{trpd}_2, \cdot)$[2]

where

- $\mathcal{F}, \mathcal{F}_s$ are PRFs, and $\mathcal{E}, \mathcal{D}$ are part of RCPA-secure symmetric key encryption.

---

[1] Output splitting is used in [8] instead.

[2] Response revealing.

Figure: Multi-Map

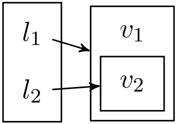


Figure: Recursive Map

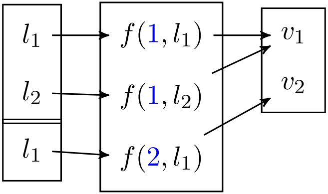
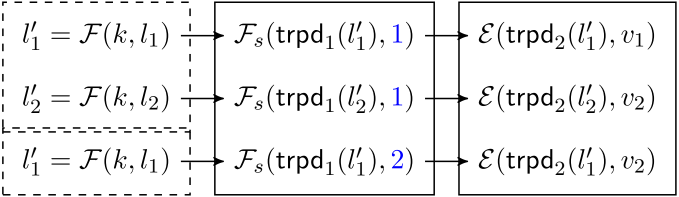


Figure: Encrypted Multi-Map



Encrypted Multi-Map

Π_{bas} Security and Correctness

Provided by Cash *et al.* [7].

Π_{bas} Structure

Π_{bas} preserves the *equivalent* recursive map for the multi-map.

- Homomorphism: by proving comp. ind. 1-1 on the label transformation. What about value transformation?
- Elementary equivalence: by Ehrenfreucht-Fraïssé Game [1, 2], with extension to pseudorandom objects.

Emulation of Encrypted Multi-Map

label	value
$\mathcal{F}_s(\mathsf{trpd}_1(l'_1), 1)$	$\mathcal{E}(\mathsf{trpd}_2(l'_1), v_1)$
$\mathcal{F}_s(\mathsf{trpd}_1(l'_2), 1)$	$\mathcal{E}(\mathsf{trpd}_2(l'_2), v_2)$
$\mathcal{F}_s(\mathsf{trpd}_1(l'_1), 2)$	$\mathcal{E}(\mathsf{trpd}_2(l'_1), v_2)$

Table: T_{EMM} for Π_{bas}. $l' = \mathcal{F}(k, l), \mathsf{trpd}_j(l') = \mathcal{F}(k, l'||j)$

- Operation $\mathsf{EMM}[l]$ is emulated by $\mathbf{RCTE}(l)$ as

$$
\begin{array}{l}
\texttt{With Recursive View As} \\
\quad T_{\mathsf{EMM}} \bowtie_{\texttt{label}=\mathcal{F}_s(\mathsf{trpd}_1(l'))} \{\langle i : 1 \rangle\} \\
\texttt{Union All} \\
\quad T_{\mathsf{EMM}} \bowtie_{\texttt{label}=\mathcal{F}_s(\mathsf{trpd}_2(l'))} \pi_{i+1 \to i} \mathsf{View} \\
\pi_{\texttt{value}} \mathsf{View}
\end{array}
$$

Independence

Table: Customer$^{\mathcal{F}}$

rid	Name$^{\mathcal{F}}$	Country$^{\mathcal{F}}$
c_1	Alice$^{\mathcal{E}}$	US$^{\mathcal{E}}$
c_2	Bob$^{\mathcal{E}}$	US$^{\mathcal{E}}$
c_3	Alice$^{\mathcal{E}}$	China$^{\mathcal{E}}$

Table: Supplier$^{\mathcal{F}}$

rid	Name$^{\mathcal{F}}$	Country$^{\mathcal{F}}$
s_1	Li-Ning$^{\mathcal{E}}$	China$^{\mathcal{E}}$
s_2	Nike$^{\mathcal{E}}$	US$^{\mathcal{E}}$

Independence

Table: Customer$^{\mathcal{F}}$

rid	Name$^{\mathcal{F}}$	Country$^{\mathcal{F}}$
c_1	Alice$^{\mathcal{E}}$	US$^{\mathcal{E}}$
c_2	Bob$^{\mathcal{E}}$	US$^{\mathcal{E}}$
c_3	Alice$^{\mathcal{E}}$	China$^{\mathcal{E}}$

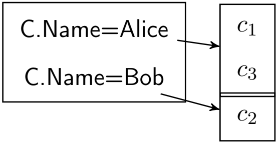
Table: Supplier$^{\mathcal{F}}$

rid	Name$^{\mathcal{F}}$	Country$^{\mathcal{F}}$
s_1	Li-Ning$^{\mathcal{E}}$	China$^{\mathcal{E}}$
s_2	Nike$^{\mathcal{E}}$	US$^{\mathcal{E}}$

Table: EMM$_{\sigma}$

label	value
$\mathcal{F}$(trpd$_1$(C.Name=Alice$'$), 1)	$\mathcal{E}$((trpd$_2$(C.Name=Alice$'$)), c_1)
f(C.Name=Bob, 1)	e(C.Name=Bob, c_2)
f(C.Name=Alice, 2)	e(C.Name=AliceS, c_3)

Figure: MM$_{\sigma}$



Independence

Table: Customer$^{\mathcal{F}}$

rid	Name$^{\mathcal{F}}$	Country$^{\mathcal{F}}$
c_1	Alice$^{\mathcal{E}}$	US$^{\mathcal{E}}$
c_2	Bob$^{\mathcal{E}}$	US$^{\mathcal{E}}$
c_3	Alice$^{\mathcal{E}}$	China$^{\mathcal{E}}$

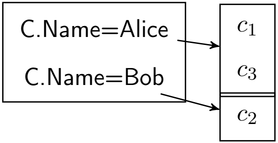
Table: Supplier$^{\mathcal{F}}$

rid	Name$^{\mathcal{F}}$	Country$^{\mathcal{F}}$
s_1	Li-Ning$^{\mathcal{E}}$	China$^{\mathcal{E}}$
s_2	Nike$^{\mathcal{E}}$	US$^{\mathcal{E}}$

Table: EMM$_\sigma$

`label`	`value`
$\mathcal{F}(\text{trpd}_1(\text{C.Name=Alice}'), 1)$	$\mathcal{E}((\text{trpd}_2(\text{C.Name=Alice}')), c_1)$
$f(\text{C.Name=Bob}, 1)$	$e(\text{C.Name=Bob}, c_2)$
$f(\text{C.Name=Alice}, 2)$	$e(\text{C.Name=AliceS}, c_3)$

Figure: MM$_\sigma$



- $\pi_{\text{Country}}\sigma_{\text{Name=Alice}}\text{Customer} \equiv$
  $\pi_{\text{Country}^{\mathcal{F}}} \underbrace{\text{Customer}^{\mathcal{F}} \ltimes_{\text{rid=value}} \mathbf{RCTE}_\sigma(\text{C.Name=Alice})}_{\sigma^{\mathcal{E}}_{\text{C.Name=Alice}}\text{Customer}^{\mathcal{F}}}$

Independence

Table: Customer$^{\mathcal{F}}$

rid	Name$^{\mathcal{F}}$	Country$^{\mathcal{F}}$
c_1	Alice$^{\mathcal{E}}$	US$^{\mathcal{E}}$
c_2	Bob$^{\mathcal{E}}$	US$^{\mathcal{E}}$
c_3	Alice$^{\mathcal{E}}$	China$^{\mathcal{E}}$

Table: Supplier$^{\mathcal{F}}$

rid	Name$^{\mathcal{F}}$	Country$^{\mathcal{F}}$
s_1	Li-Ning$^{\mathcal{E}}$	China$^{\mathcal{E}}$
s_2	Nike$^{\mathcal{E}}$	US$^{\mathcal{E}}$

Figure: $\mathsf{MM}_{\bowtie}$

$\boxed{\text{C.Country} = \text{S.Country}} \rightarrow \boxed{\begin{array}{c}(c_1, s_2)\\(c_2, s_2)\\(c_3, s_1)\end{array}}$

Independence

Table: Customer$^{\mathcal{F}}$

rid	Name$^{\mathcal{F}}$	Country$^{\mathcal{F}}$
c_1	Alice$^{\mathcal{E}}$	US$^{\mathcal{E}}$
c_2	Bob$^{\mathcal{E}}$	US$^{\mathcal{E}}$
c_3	Alice$^{\mathcal{E}}$	China$^{\mathcal{E}}$

Table: Supplier$^{\mathcal{F}}$

rid	Name$^{\mathcal{F}}$	Country$^{\mathcal{F}}$
s_1	Li-Ning$^{\mathcal{E}}$	China$^{\mathcal{E}}$
s_2	Nike$^{\mathcal{E}}$	US$^{\mathcal{E}}$

Figure: MM$_{\bowtie}$

$$\boxed{\text{C.Country} = \text{S.Country}} \longrightarrow \boxed{\begin{array}{c}(c_1, s_2)\\(c_2, s_2)\\(c_3, s_1)\end{array}}$$

Table: EMM$_{\bowtie}$

`label`	`value`$_1$	`value`$_2$
$f(\text{C.Country} = \text{S.Country}, 1)$	$e(\text{C.Country} = \text{S.Country}, c_1)$	$e(\text{C.Country} = \text{S.Country}, s_2)$
$f(\text{C.Country} = \text{S.Country}, 2)$	$e(\text{C.Country} = \text{S.Country}, c_2)$	$e(\text{C.Country} = \text{S.Country}, s_2)$
$f(\text{C.Country} = \text{S.Country}, 3)$	$e(\text{C.Country} = \text{S.Country}, c_3)$	$e(\text{C.Country} = \text{S.Country}, s_1)$

Independence

Table: Customer$^{\mathcal{F}}$

rid	Name$^{\mathcal{F}}$	Country$^{\mathcal{F}}$
c_1	Alice$^{\mathcal{E}}$	US$^{\mathcal{E}}$
c_2	Bob$^{\mathcal{E}}$	US$^{\mathcal{E}}$
c_3	Alice$^{\mathcal{E}}$	China$^{\mathcal{E}}$

Table: Supplier$^{\mathcal{F}}$

rid	Name$^{\mathcal{F}}$	Country$^{\mathcal{F}}$
s_1	Li-Ning$^{\mathcal{E}}$	China$^{\mathcal{E}}$
s_2	Nike$^{\mathcal{E}}$	US$^{\mathcal{E}}$

Figure: MM$_{\bowtie}$

$\boxed{\text{C.Country} = \text{S.Country}} \rightarrow \boxed{(c_1, s_2) \quad (c_2, s_2) \quad (c_3, s_1)}$

Table: EMM$_{\bowtie}$

`label`	`value`$_1$	`value`$_2$
$f(\text{C.Country} = \text{S.Country}, 1)$	$e(\text{C.Country} = \text{S.Country}, c_1)$	$e(\text{C.Country} = \text{S.Country}, s_2)$
$f(\text{C.Country} = \text{S.Country}, 2)$	$e(\text{C.Country} = \text{S.Country}, c_2)$	$e(\text{C.Country} = \text{S.Country}, s_2)$
$f(\text{C.Country} = \text{S.Country}, 3)$	$e(\text{C.Country} = \text{S.Country}, c_3)$	$e(\text{C.Country} = \text{S.Country}, s_1)$

- Customer $\bowtie_{\text{C.Country=S.Country}}$ Supplier $\equiv$ $\underbrace{\mathbf{RCTE}_{\bowtie}(\text{C.Country=S.Country}) \bowtie_{\texttt{value}_1=\text{rid}} \text{C} \bowtie_{\texttt{value}_2=\text{rid}} \text{S}}_{\text{C}^{\mathcal{E}} \bowtie^{\mathcal{E}}_{\text{C.Country=S.Country}} \text{S}^{\mathcal{E}}}$

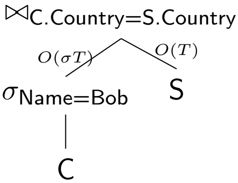
Independence

Table: Customer$^{\mathcal{F}}$

rid	Name$^{\mathcal{F}}$	Country$^{\mathcal{F}}$
c_1	Alice$^{\mathcal{E}}$	US$^{\mathcal{E}}$
c_2	Bob$^{\mathcal{E}}$	US$^{\mathcal{E}}$
c_3	Alice$^{\mathcal{E}}$	China$^{\mathcal{E}}$

Table: Supplier$^{\mathcal{F}}$

rid	Name$^{\mathcal{F}}$	Country$^{\mathcal{F}}$
s_1	Li-Ning$^{\mathcal{E}}$	China$^{\mathcal{E}}$
s_2	Nike$^{\mathcal{E}}$	US$^{\mathcal{E}}$



Hints from thought experiment:

- On efficiency: $O(\sigma T^2)$ where σ is the selectivity
- On optimal leakage: only the encrypted result set

Independence

Figure: MM_σ

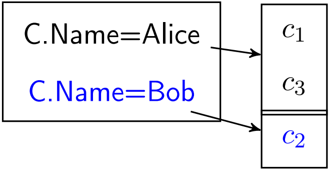
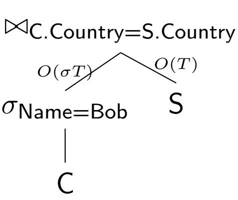
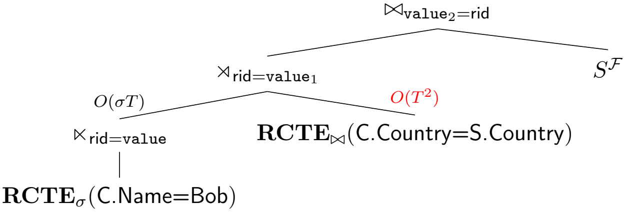


Figure: $\mathrm{MM}_\bowtie$





Independence

Table: Customer$^{\mathcal{F}}$

rid	Name$^{\mathcal{F}}$	Country$^{\mathcal{F}}$
c_1	Alice$^{\mathcal{E}}$	US$^{\mathcal{E}}$
c_2	Bob$^{\mathcal{E}}$	US$^{\mathcal{E}}$
c_3	Alice$^{\mathcal{E}}$	China$^{\mathcal{E}}$

Table: Supplier$^{\mathcal{F}}$

rid	Name$^{\mathcal{F}}$	Country$^{\mathcal{F}}$
s_1	Li-Ning$^{\mathcal{E}}$	China$^{\mathcal{E}}$
s_2	Nike$^{\mathcal{E}}$	US$^{\mathcal{E}}$

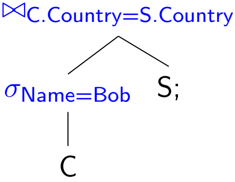
- Also similar issue with compound formula, such as $C.Name = Alice \wedge C.Country = US$ for filters, or $C.Country = S.Country \wedge C.Name = S.Name$.

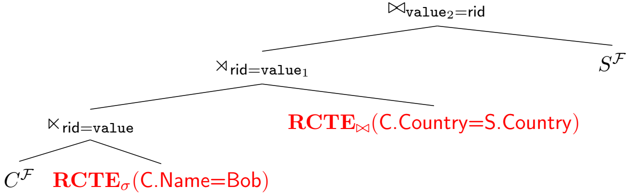
Suboptimality

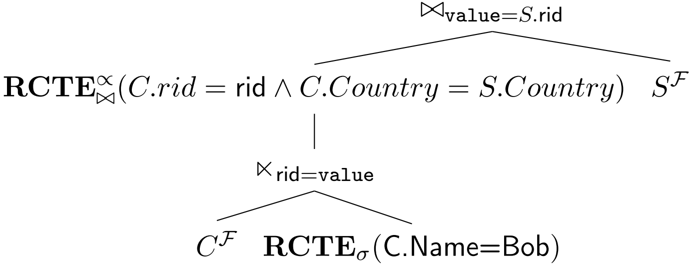
Independent operators are suboptimal in both efficiency and leakage for correlated filter and join predicates.

Dependence

- Introduce dependence between encrypted operators







Dependence

Figure: MM_σ

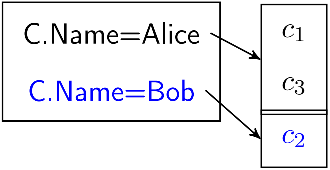


Figure: $\mathrm{MM}_{\bowtie}^{\propto}$

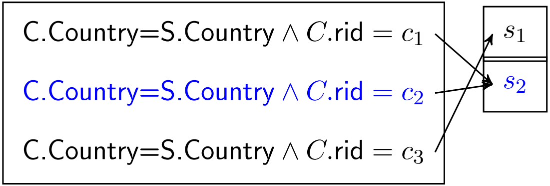


Figure: MM_σ

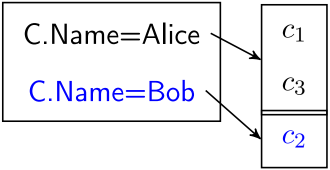
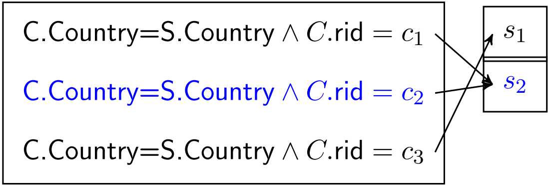


Figure: $\mathsf{MM}_{\bowtie}^{\propto}$



- Problem: Client has to keep $O(T)$ trapdoors for $\mathsf{MM}_{\bowtie}^{\propto}$, and interaction
- Solution:
  - Client: master trapdoor $\mathsf{trpd}_{\bowtie} = \mathcal{F}(k, \text{C.Country=S.Country})$
  - Server: derive $O(T)$ trapdoors for each c_i as $\mathsf{trpd}_j = \mathcal{F}(\mathsf{trpd}_{\bowtie}, c_i||j)$ for $j = 1, 2$

Construction

Normal Form

Figure: General Multi-Map

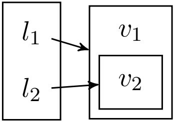


Figure: MM_σ

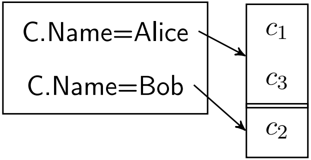


Figure: General Multi-Map

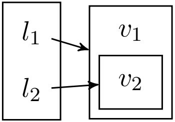
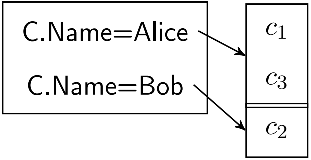


Figure: MM_σ



Multi-Map Range Partition under 1NF

The range of MM_σ *always* forms a partition in the space of $C^{\mathcal{F}}$.rid $= \{c_i\}_{i=1}^{T}$, because of 1st normal form

- $C.Name$ is an elemantary set
- $C^{\mathcal{F}}$.rid $= \{c_i\}_{i=1}^{T}$ is a candidate key $\rightarrow$ uniquely identifies a row

By contrast

- The general multi-map may have overlapping value sets in its range
- EMM suitable for document keyword model, but overkill relational model?

Normal Form

Figure: Many-to-Many Join

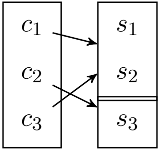


Figure: 1-to-Many Join

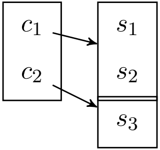
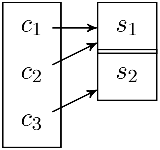


Figure: Many-to-1 Join



Multi-Map Range Partition for Joins

The range partition generalize to $\mathrm{MM}^{\propto}_{\bowtie}$ too.

Normal Form

Figure: Many-to-Many Join

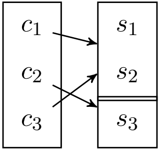


Figure: 1-to-Many Join

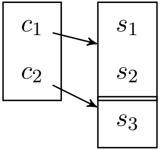
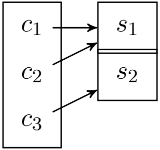


Figure: Many-to-1 Join



Multi-Map Range Partition for Joins

The range partition generalize to $\mathrm{MM}_{\bowtie}^{\propto}$ too.

3NF has only three key joins:

- Foreign-to-foreign key join $\in$ many-to-many join
- Primary-to-foreign key join $\in$ 1-to-many join
- Foreign-to-primary key join $\in$ many-to-one join

Worst-case Optimal Space for Joins

The worst-case optimal space for joins in a 3NF data model is $O(T)$.

Normal Form

Adapt EMM for 1/3NF for efficiency/security?

Normal Form

Adapt EMM for 1/3NF for efficiency/security?

Semi-Encrypted Multi-Map

The semi-encrypted multi-map is Π_{bas} with its values in clear. Formally it is a transformation of MM

- Client: $\mathcal{F}(k, \cdot)$, $\text{trpd}_j(l') = \mathcal{F}(k, l'||j)$
- Server: $\mathcal{F}_s(\cdot, i)$

Figure: Encrypted Multi-Map

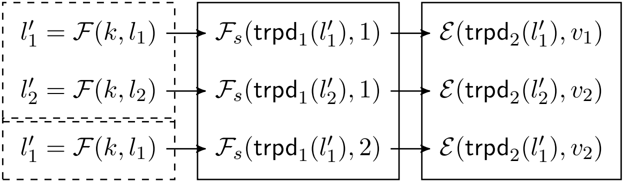
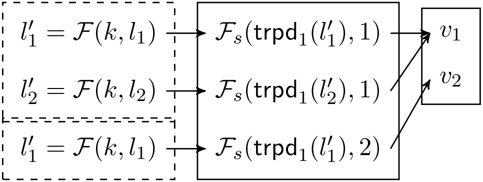


Figure: Semi-Encrypted Multi-Map



Leaks "co-occurence" pattern. Insecure for document keyword model.

Normal Form

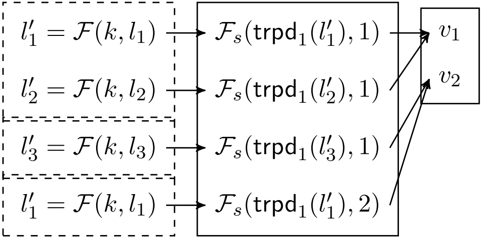
Security of SEMM under 1NF

The SEMM under 1NF leaks only dimensions of table (trivial co-ocurrence). SEMM achieves the same security as Π_{bas} for range as row ids.

Proof sketch:

- Co-occurrence is the same for every row. Each v_i uniquely identifies a row. So the range size is # rows. Number of in-edges for each row id v_i is always equal to # attributes. All rows have the same # attributes.

Figure: SEMM is Secure under 1NF



Normal Form

Implications

1. Can divide the SEMM and collocate with tables such that $SEMM.\texttt{value} = \text{rid}$.
2. Share the cleartext $SEMM.\texttt{value}$ for all SEMMs for the same table to reduce SEMM size.
3. Pre-indexing on $SEMM.\texttt{value}$ to reduce computation time.

Normal Form

Implications

1. Can divide the SEMM and collocate with tables such that $SEMM.\texttt{value} =$ rid.
2. Share the cleartext $SEMM.\texttt{value}$ for all SEMMs for the same table to reduce SEMM size.
3. Pre-indexing on $SEMM.\texttt{value}$ to reduce computation time.

Achieve worst-case optimal space for joins

1. SEMM only index many-to-1 or 1-to-many joins.
2. Worst-case optimal space for joins: avoid storing many-to-many joins in SEMM, but factor them into two many-to-1 or 1-to-many joins.

Other Topics

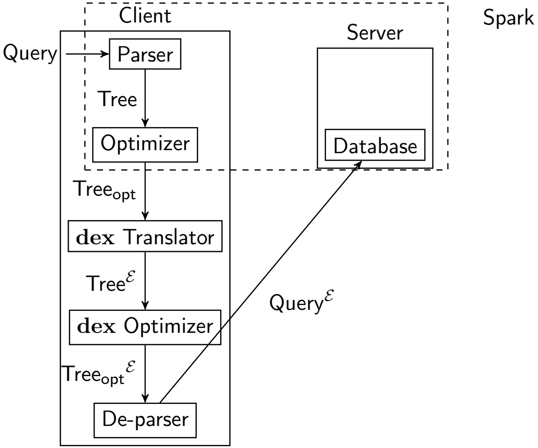
1. Worst-case optimal joins (efficiency). Idea from [7]
2. Range queries [10]
3. Updates with minimum interactions [12]
4. Consolidate the framework/proofs in this thesis
5. Query optimization
6. Security against malicious server

Construction

4. Emulation
   - Map
   - Multi-Map
   - Encrypted Multi-Map
5. Encrypted SQL
   - Independence
   - Dependence
   - Normal Form
6. Other Topics
7. System & Evaluation

- Prototype system is open-source [15]
- Based on the algebraic core of Apache Spark SQL [9], interface with any database endpoints e.g. PostgreSQL [13]
- Parallel database encryption
- Fusion of plaintext and encrypted operators.

Figure: The **dex** encrypted relational database.



TPC-H Benchmark [5] on the dependence scheme (without normal form).

dex-cor	mean(ms)(m44xlarge)	rel.err.	slowdown vs. Postgres	slowdown vs. Postgres(t22xlarge)
q1	2149.5	1.38%	1.4	1.0
q10	217837.6	0.16%	115.0	32.7
q11	993.3	2.15%	13.8	5.4
q12	38976.8	0.25%	32.7	26.9
q13	61460.2	0.27%	84.0	64.9
q14	110169.4	0.15%	40.5	29.7
q15	100299.6	0.29%	41.0	33.3
q16	478.2	1.99%	3.0	2.6
q17	437.2	2.29%	6.0	4.0
q18	280982.8	0.28%	66.6	50.0
q19	31976.2	0.36%	373.1	324.3
q2	2445.4	0.52%	15.8	12.3
q20	27546.1	0.51%	284.6	35.7
q21	447845.6	0.22%	441.9	354.6
q22	55829.3	0.36%	134.1	108.7
q3	40152.1	0.38%	22.8	17.2
q4	119831.3	0.20%	30.6	23.1
q5	4645337.0	0.62%	2231.8	1875.5
q7	47910.1	0.30%	131.8	81.6
q8	117817.6	0.44%	57.4	38.8
q9a	464302.6	0.20%	10.8	12.3
q9b	356243.1	0.32%	8.3	9.4

table name	row est.	attrs	page est.	total(bytes)	index(bytes)	table(bytes)
region	5	4(3)	1	32 k (64 k)	16 k(48 k)	8192
orders	$1.5e+06$	10(9)	71484(26405)	604 m(603 m)	45 m(396 m)	559 m(206 m)
supplier	10000	8(7)	447(226)	3936 k(4680 k)	328 k(2840 k)	3600 k(1832 k)
customer	150000	9(8)	7620(3758)	64 m(75 m)	4640 k(46 m)	60 m (29m)
partsupp	800000	7(5)	35580(18242)	302 m(337 m)	24 m(194 m)	278 m (143 m)
nation	25	5(4)	1	32 k(80 k)	16 k(64 k)	8192
lineitem	$6.00139e+06$	19(16)	500251(117594)	4090 m(3338 m)	181 m(2419 m)	3909 m(919 m)
part	200000	10(9)	9656(3832)	82 m(85 m)	6184 k(55 m)	75 m(30 m)
$T_{\sigma}^{\perp}$	$8.74646e+07$	2	1079906	13 G	4926 m	8439 m
$T_{\bowtie}^{\propto}$	$5.45298e+07$	2	673291	8332 m	3071 m	5261 m

Table: TPCH benchmark **dex-cor** versus plaintext Postgres storage size. The plaintext Postgres storage size is shown in parenthesis. TPC-H scale factor is 1.

References I

1. Fraïssé, R. *Sur quelques classifications des systemes de relations.* PhD thesis (1955).
2. Ehrenfeucht, A. An application of games to the completeness problem for formalized theories. *Fund. Math* **49,** 13 (1961).
3. Eisenberg, A. & Melton, J. SQL: 1999, formerly known as SQL3. *ACM Sigmod record* **28,** 131–138 (1999).
4. Bellare, M. & Rogaway, P. *The security of triple encryption and a framework for code-based game-playing proofs.* in *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (2006), 409–426.
5. Council, T. P. P. TPC-H benchmark specification. *Published at http://www. tcp. org/hspec. html* **21,** 592–603 (2008).

References II

6. Popa, R. A., Redfield, C., Zeldovich, N. & Balakrishnan, H. *CryptDB: protecting confidentiality with encrypted query processing.* in *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles* (2011), 85–100.
7. Cash, D. *et al. Highly-scalable searchable symmetric encryption with support for boolean queries.* in *Annual Cryptology Conference* (2013), 353–373.
8. Cash, D. *et al. Dynamic searchable encryption in very-large databases: data structures and implementation..* in *NDSS* **14** (2014), 23–26.
9. Armbrust, M. *et al. Spark sql: Relational data processing in spark.* in *Proceedings of the 2015 ACM SIGMOD international conference on management of data* (2015), 1383–1394.

References III

10. Faber, S. *et al.* *Rich queries on encrypted data: Beyond exact matches.* in *European Symposium on Research in Computer Security* (2015), 123–145.
11. Degitz, A., Köhler, J. & Hartenstein, H. *Access Pattern Confidentiality-Preserving Relational Databases: Deployment Concept and Efficiency Evaluation..* in *EDBT/ICDT Workshops* (2016).
12. Kamara, S. & Moataz, T. *SQL on structurally-encrypted databases.* in *International Conference on the Theory and Application of Cryptology and Information Security* (2018), 149–180.
13. Group, T. P. G. D. *PostgreSQL.* `https://www.postgresql.org/`. 2019.

References IV

14. Tan, B. H. M., Lee, H. T., Wang, H., Ren, S. Q. & Aung, K. M. M. Efficient Private Comparison Queries over Encrypted Databases using Fully Homomorphic Encryption with Finite Fields. *IACR Cryptology ePrint Archive* **2019**, 332 (2019).

15. Zhao, Z. *encrypted-spark.* https://github.com/zheguang/encrypted-spark/tree/dex. 2019.