Learning Objectives

- Understand an important research case in cloud computing
  - Database as a Service
- Identify research methods used in the research case
Research Case: The DAS Project

- Database as a Service Project @ UC Irvine
  - http://www-db.ics.uci.edu/pages/research/das/
- Pioneering work of SaaS type research in data management
  - Recipient of SIGMOD 10-year best paper
- Two representative works:
  - “Providing Database as a Service,” ICDE 2002
  - “Executing SQL over Encrypted Data in Database Service Provider Model,” SIGMOD 2002
Objective

• Want to store the data on “a server”

• But the **problem** is we **do not trust** “the server” for sensitive information!
  - encrypt the data and store it
  - but still be able to **run queries over the encrypted data**
  - do **most of the work** at the server

• If the server is trusted, ICDE 2002, else SIGMOD 2002
Database as a Service

- **SaaS model for Database**
  - DB management transferred to service provider for
    - backup, administration, restoration, space management, upgrades etc.
  - use the database “as a service” provided by Cloud computing
    - use SW, HW, human resources of CC, instead of your own
Service Provider Architecture

Client Site

- Query Executor
- Query Translator
- Metadata
- Client Side Query
- Original Query
- Temporary Results
- Actual Results

Server Site

- Encrypted User Database
- User
- Server Side Query
- Server Site
- Encrypted Results
- Service Provider

Actual Results

Original Query

Client Side Query
### Relational Encryption

#### NAME | SALARY | PID
--- | --- | ---
John | 50000 | 2
Marry | 110000 | 2
James | 95000 | 3
Lisa | 105000 | 4

#### etuple | N_ID | S_ID | P_ID
--- | --- | --- | ---
fErf!$Q!!vddf>></| | 50 | 1 | 10
F%]%3w&%gfErf!$ | 65 | 2 | 10
&%gfsdf$%343v<l | 50 | 2 | 20
%%33w&%gfs# #! | 65 | 2 | 20

---

- Store an encrypted string – *etuple* – for each tuple in the original table
  - This is called “row level encryption”
  - Any kind of encryption technique can be used
- Create an index for each (or selected) attribute(s) in the original table

---

Server Site
Building the Index

- **Partition function** divides domain values into partitions (= buckets)
  \[ Partition (R.A) = \{ [0,200], (200,400], (400,600], (600,800], (800,1000] \} \]
  - partitioning function has an impact on performance as well as privacy

- **Identification function** assigns a partition id to each partition of attribute \( A \)
  - e.g. \( ident_{R.A}( (200,400] ) = 7 \)
  - Any function can be use as identification function, e.g., hash functions
Mapping Functions

- **Mapping function** maps a value \( v \) in the domain of attribute \( A \) to the id of the partition which value \( v \) belongs to.

  \[
  \begin{align*}
  \text{Map}_{R.A}(250) &= 7, \quad \text{Map}_{R.A}(620) = 1
  \end{align*}
  \]
Storing Encrypted Data

\[ R = \langle A, B, C \rangle \Rightarrow R^S = \langle \text{etuple}, A\_id, B\_id, C\_id \rangle \]

etuple = encrypt \((A \mid B \mid C)\)

\[ A\_id = \text{Map}_{R.A}(A), \; B\_id = \text{Map}_{R.B}(B), \; C\_id = \text{Map}_{R.C}(C) \]

<table>
<thead>
<tr>
<th>NAME</th>
<th>SALARY</th>
<th>PID</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>50000</td>
<td>2</td>
</tr>
<tr>
<td>Marry</td>
<td>110000</td>
<td>2</td>
</tr>
<tr>
<td>James</td>
<td>95000</td>
<td>3</td>
</tr>
<tr>
<td>Lisa</td>
<td>105000</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NAME</th>
<th>SALARY</th>
<th>PID</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>50000</td>
<td>2</td>
</tr>
<tr>
<td>Marry</td>
<td>110000</td>
<td>2</td>
</tr>
<tr>
<td>James</td>
<td>95000</td>
<td>3</td>
</tr>
<tr>
<td>Lisa</td>
<td>105000</td>
<td>4</td>
</tr>
</tbody>
</table>

Table: EMPLOYEE

<table>
<thead>
<tr>
<th>NAME</th>
<th>SALARY</th>
<th>PID</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>50000</td>
<td>2</td>
</tr>
<tr>
<td>Marry</td>
<td>110000</td>
<td>2</td>
</tr>
<tr>
<td>James</td>
<td>95000</td>
<td>3</td>
</tr>
<tr>
<td>Lisa</td>
<td>105000</td>
<td>4</td>
</tr>
</tbody>
</table>

Table: EMPLOYEE\(^S\)

<table>
<thead>
<tr>
<th>Etuple</th>
<th>N_ID</th>
<th>S_ID</th>
<th>P_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>fErf!$Q!!vddf&gt;&gt;/</td>
<td>50</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>F%%3w&amp;%gfErf!$</td>
<td>65</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>&amp;%gfsdf$%343v&lt;l</td>
<td>50</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>%%33w&amp;%gfs##!</td>
<td>65</td>
<td>2</td>
<td>20</td>
</tr>
</tbody>
</table>
Mapping Conditions

Q: SELECT name, pname FROM emp, proj
   WHERE emp.pid=proj.pid AND salary > 100k

- Server stores attribute indices determined by mapping functions
- Client stores metadata and utilizes that to translate the query

Conditions:
- Condition ← Attribute \( op \) Value
- Condition ← Attribute \( op \) Attribute
- Condition ← (Condition or Condition) | (Condition and Condition) | (not Condition)
Mapping Conditions (2)

Example:

- **Attribute = Value**
  - $Map_{\text{cond}}( A = v ) \Rightarrow A^S = Map_A( v )$
  - $Map_{\text{cond}}( A = 250 ) \Rightarrow A^S = 7$
Mapping Conditions (3)

- Attribute1 = Attribute2
- \( Map_{\text{cond}}( A = B ) \Rightarrow \bigvee_{N} (A^s = \text{ident}_A(p_k) \land B^s = \text{ident}_B(p_l)) \)

where \( N \) is \( p_k \in \text{partition}(A), \ p_l \in \text{partition}(B), \ p_k \cap p_l \neq \emptyset \)

<table>
<thead>
<tr>
<th>Partitions</th>
<th>A_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0,100]</td>
<td>2</td>
</tr>
<tr>
<td>(100,200]</td>
<td>4</td>
</tr>
<tr>
<td>(200,300]</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Partitions</th>
<th>B_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0,200]</td>
<td>9</td>
</tr>
<tr>
<td>(200,400]</td>
<td>8</td>
</tr>
</tbody>
</table>

C : A = B \Rightarrow C' : (A^s = 2 \text{ and } B^s = 9) \text{ or } (A^s = 4 \text{ and } B^s = 9) \text{ or } (A^s = 3 \text{ and } B^s = 8)
Relational Operators over Encrypted Relations

- Partition the computation of the operators across client and server
- Compute (possibly) superset of answers at the server
- Filter the answers at the client

Objective: minimize the work at the client and process the answers as soon as they arrive without requiring storage at the client

Operators studied:
- **Selection**
- **Join**
- Grouping and Aggregation
- Sorting
- Duplicate Elimination
- Set Difference
- Union
- Projection
Selection Operator

$$\sigma_c( R ) = \sigma_c( D (\sigma^{S}_{\text{Mapcond}(c)}( R^S ))$$

Example:

$$\sigma_{A=250}$$

Client Query

Server Query

$$\sigma_{A_id = 7}$$

E_TABLE

TABLE

Client Table

2 | 7 | 5 | 1 | 4

0 | 200 | 400 | 600 | 800 | 1000
Join Operator

\[ R \bowtie_c T = \sigma_c( D ( R^S \bowtie_{\text{Mapcond}(c)} T^S ) ) \]

**Example:**

\[ \bowtie_c C \]

\[ \text{EMP} \quad \text{PROJ} \]

\[ \sigma_{A=B} \]

\[ D \]

\[ \bowtie_{C'} \]

\[ \text{E_EMP} \quad \text{E_PROJ} \]

\[ \sigma_{A=B} \]

\[ \text{C'} \]

\[ \text{C : A = B} \quad \Rightarrow \quad \text{C'} : \]

\[ (A_{id} = 2 \text{ and } B_{id} = 9) \]

\[ \text{Or (A_{id} = 4 and B_{id} = 9)} \]

\[ \text{Or (A_{id} = 3 and B_{id} = 8)} \]
Query Decomposition

Q: SELECT name, pname FROM emp, proj
WHERE emp.pid=proj.pid AND salary > 100k
Query Decomposition (2)

Client Query

- \( \pi_{\text{name}, \text{pname}} \)
  - \( e.pid = p.pid \)
  - \( \sigma_{\text{salary} > 100k} \)
  - \( \pi_{\text{name}, \text{pname}} \)
  - \( \sigma_{\text{salary} > 100k} \)
  - \( \pi_{\text{name}, \text{pname}} \)
  - \( \sigma_{s_id = 1 \lor s_id = 2} \)

Server Query

- \( \sigma_{\text{salary} > 100k} \)
  - \( \pi_{\text{name}, \text{pname}} \)
  - \( e.pid = p.pid \)
  - \( \sigma_{\text{salary} > 100k} \)
  - \( \pi_{\text{name}, \text{pname}} \)
  - \( e.pid = p.pid \)
  - \( \sigma_{s_id = 1 \lor s_id = 2} \)
Query Decomposition (3)

Client Query

\[ \pi_{\text{name}, \text{pname}} \]
\[ \sigma_{\text{salary} > 100k} \]
\[ \sigma_{\text{e.pid} = \text{p.pid}} \]
\[ D \]

Server Query

\[ \sigma_{\text{s_id} = 1 \lor \text{s_id} = 2} \]
\[ E_{\text{PROJ}} \]

Client Query

\[ \pi_{\text{name}, \text{pname}} \]
\[ \sigma_{\text{salary} > 100k \land \text{e.pid} = \text{p.pid}} \]
\[ D \]

Server Query

\[ \sigma_{\text{s_id} = 1 \lor \text{s_id} = 2} \]
\[ E_{\text{PROJ}} \]
\[ E_{\text{EMP}} \]
Query Decomposition (4)

Client Query

\[ \Pi_{\text{name, pname}} \]
\[ \sigma_{\text{salary } > 100k \text{ and } \text{e.pid } = \text{p.pid}} \]

Server Query

\[ \sigma_{\text{s_id } = 1 \text{ OR } \text{s_id } = 2} \]
\[ \text{E_EMP} \]
\[ \text{E_PROJ} \]

Q: SELECT name, pname
FROM emp, proj
WHERE emp.pid=proj.pid AND salary > 100k

Q_s: SELECT e_emp.etuple, e_proj.etuple FROM e_emp, e_proj
WHERE e.p_id=p.p_id AND s_id = 1 OR s_id = 2

Q_c: SELECT name, pname
FROM temp
WHERE emp.pid=proj.pid AND salary > 100k
Experimental Evaluation

- Data
  - TPC-H database

- Queries
  - TPC-H Queries, versions of Q#6 and Q#3

- Partitioning Strategy
  - Equi-depth histograms for the first set of experiments
  - Equi-width histograms for the second set of experiments
Effect of # of Buckets in Non-Join Query

- Client and communications costs decreases with increasing number of buckets due to better filtering at the server.
Effect of # of Buckets in Non-Join Query

- Single Server: Server is trusted and performs all operations including decryption on site
- Shows that proposed query execution protocol doesn’t introduce significant overhead
Effect of # of Buckets in Join Query

- **Single Server**: Server is trusted and performs all operations including decryption on site.
- **Consistent with the previous results showing proposed communication protocol doesn’t introduce significant overhead.**
Findings

- Database-as-a-Service model is a promising solution
- Proposed algebraic solution to provide data privacy when cloud computing host is not trusted
  - encrypts data, creates “coarse indexes” and stores the data at the server
  - allows only data owner to decrypt the data
- With query decomposition
  - most of query execution performed at the server
  - client only performs filtering
## CI Research Methods Used

<table>
<thead>
<tr>
<th>Method</th>
<th>SIGMOD 02</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lit-Survey</td>
<td></td>
</tr>
<tr>
<td>Formal</td>
<td></td>
</tr>
<tr>
<td>Solution</td>
<td></td>
</tr>
<tr>
<td>Experimental</td>
<td></td>
</tr>
<tr>
<td>Build</td>
<td></td>
</tr>
<tr>
<td>Process</td>
<td></td>
</tr>
<tr>
<td>Model</td>
<td></td>
</tr>
</tbody>
</table>
Reference

- “Executing SQL over Encrypted Data in Database Service Provider Model,” SIGMOD 2002