Adaptive Schema Databases

William Spoth\textsuperscript{b}, Bahareh Sadat Arab\textsuperscript{i}, Eric S. Chan\textsuperscript{o}, Dieter Gawlick\textsuperscript{o}, Adel Ghoneimy\textsuperscript{o}, Boris Glavic\textsuperscript{i}, Beda Hammerschmidt\textsuperscript{o}, Oliver Kennedy\textsuperscript{b}, Seokki Lee\textsuperscript{i}, Zhen Hua Liu\textsuperscript{o}, Xing Niu\textsuperscript{i}, \textbf{Ying Yang}\textsuperscript{b}

\textsuperscript{b}: University at Buffalo \textsuperscript{i}: Illinois Inst. Tech. \textsuperscript{o}: Oracle
Adaptive Schema Databases
Classic relational database

• Navigational and organizational purpose

retain discovery, good performance and space, reusable.
Classic relational database

• But... High upfront cost and inflexible
BigData/NOSQL

• Data can be used immediately.
BigData/NOSQL

• But... Sacrifice navigational and Performance benefit and may end up with duplicate of work
Adaptive Schema Databases

- Bridge the gap between relational database and NoSQL.

Queries and feedback...

eventually
Adaptive Schema Databases

- Bridge the gap between relational database and NoSQL.
Adaptive Schema Databases

Input:
{"grad":{"students":[
   {name:"Alice",deg:"PhD",credits:"10"},
   {name:"Bob",deg:"MS"}, ...]},
"undergrad":{"students":[
   {name:"Carol"},{name:"Dave",deg:"U"}, ...]}}

Queries:
SELECT name FROM Undergrad UNION
SELECT name FROM Grad
SELECT deg FROM Grad
SELECT name FROM Student
...

Outline

• Extraction and discovery
• Adaptive, personalized schemas from queries
• Explanations and feedback
• Adaptive organization
• Conclusions and future work
Extraction

Schema Workspace

Queries + Feedback

Extraction Schema Candidates

Extraction workflow

Unstructured Data

Semi-structured Data (e.g., JSON)

Extraction and discovery
Extraction

- ASD extracts schema candidate set

Given input:

```json
"grad":{"students":[
    {name:"Alice",deg:"PhD",credits:"10"},
    {name:"Bob",deg:"MS"}, ...]},
"undergrad":{"students":[
    {name:"Carol"}, {name:"Dave",deg:"U"}, ...]]}
```

<table>
<thead>
<tr>
<th>Undergrad Name</th>
<th>Grad Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carol</td>
<td>Alice</td>
</tr>
<tr>
<td>Dave</td>
<td>Bob</td>
</tr>
</tbody>
</table>
Extraction

- ASD extracts schema candidate set

Given input:

```json
{"grad":{"students":[
   {name:"Alice", deg:"PhD", credits:"10"},
   {name:"Bob", deg:"MS"}, ...
],
"undergrad":{"students":[
   {name:"Carol"}, {name:"Dave", deg:"U"}, ...
]}
```
Extraction

- ASD extracts schema candidate set

Given input:

```json
{"grad":{"students":[
{name:"Alice",deg:"PhD",credits:"10"},
{name:"Bob",deg:"MS"}, ...],
"undergrad":{"students":[
{name:"Carol"},{name:"Dave",deg:"U"}, ...]}}
```
Extraction

• ASD extracts schema candidate set

Given input:

```json
{"grad":{"students":[
    {"name":"Alice","deg":"PhD","credits":"10"},
    {"name":"Bob","deg":"MS"}, ...]},
"undergrad":{"students":[
    {"name":"Carol"},{"name":"Dave","deg":"U"}, ...]}}
```
Discovery

- ASD extracts schema candidate set

\[ C_{ext} = \{ S_{ext}, P_{ext} \} \]

where \( S_{ext} \) is a set of candidate schemas, and \( P_{ext} \) is a probability distribution over these schemas.

\[ \begin{array}{c|c}
\text{Name} & \text{Degree} \\
\hline
Alice & PhD \\
Bob & MS \\
Carol & \text{(null)} \\
Dave & U \\
\end{array} \]

\[ \begin{array}{c|c}
\text{Name} & \text{Degree} \\
\hline
Carol & \text{(null)} \\
Dave & U \\
\end{array} \]

\[ \begin{array}{c|c|c}
\text{Name} & \text{Degree} & \text{Credits} \\
\hline
Alice & PhD & 10 \\
Bob & MS & \text{(null)} \\
\end{array} \]

\[ \begin{array}{c|c}
\text{Name} & \text{Degree} \\
\hline
Carol & \text{(null)} \\
Dave & U \\
\end{array} \]
Discovery

- ASD extracts schema candidate set

<table>
<thead>
<tr>
<th>Student Name</th>
<th>Student Degree</th>
<th>Undergrad Name</th>
<th>Grad Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>PhD</td>
<td>Carol</td>
<td>Alice</td>
</tr>
<tr>
<td>Bob</td>
<td>MS</td>
<td>Bob</td>
<td>Bob</td>
</tr>
<tr>
<td>Carol</td>
<td>(null)</td>
<td>Dave</td>
<td>Dave</td>
</tr>
</tbody>
</table>

(a) $P = 0.19$  (b) $P = 0.27$

<table>
<thead>
<tr>
<th>Undergrad Name</th>
<th>Degree</th>
<th>Grad Name</th>
<th>Degree</th>
<th>Credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carol</td>
<td>(null)</td>
<td>Alice</td>
<td>PhD</td>
<td>10</td>
</tr>
<tr>
<td>Dave</td>
<td>U</td>
<td>Bob</td>
<td>MS</td>
<td>(null)</td>
</tr>
</tbody>
</table>

(c) $P = 0.22$

(d) $P = 0.32$

Smax: the best guess schema
Adaptive, personalized schemas from queries

Adaptive, personalized schemas from queries

Queries + Feedback

Schema Workspace

Schema Workspace

Schema Workspace

Schema Workspace

Schema Matching

Extraction Schema Candidates

Extraction workflow

Extraction workflow

Extraction workflow

Unstructured Data

Semi-structured Data (e.g., JSON)
Adaptive, personalized schemas

- ASD maintains a set of schema workspaces $W=\{W_1, \ldots, W_n\}$.

Initially, $W=\{\}$
Finding Schemas from Queries

- ASD maintains a set of schema workspaces $W=\{W_1, \ldots, W_n\}$.

Query 1:  
```
SELECT name FROM Undergrad UNION
SELECT name FROM Grad
```
Finding Schemas from Queries

- ASD maintains a set of schema workspaces $W = \{W_1, \ldots, W_n\}$.

Query 1:  
```
SELECT name FROM Undergrad UNION
SELECT name FROM Grad
```
Finding Schemas from Queries

- ASD maintains a set of schema workspaces $W = \{W_1, \ldots, W_n\}$.

Query 2: SELECT `deg` FROM `Grad`
Synthesizing Tables

- ASD maintains a set of schema workspaces \( W = \{ W_1, \ldots, W_n \} \).

Query 3: SELECT name FROM Student

\[
W_1 = (S_1 = \{ \text{Undergrad(name)} \}, P_1 = 0.27), \\
(S_1 = \{ \text{Grad(name)} \}, P_1 = 0.23), \\
(S_1 = \{ \text{Undergrad(name), Grad(name)} \}, P_1 = 0.5)
\]
Explanations and feedback

Unstructured Data

Semi-structured Data (e.g., JSON)

Extraction workflow

Schema Workspace

Queries + Feedback

Extraction Schema Candidates

Schema Matching

Extraction workflow

Unstructured Data

Semi-structured Data (e.g., JSON)
What might go wrong

Extraction errors appear in three forms:
(1) A query incompatible with $S_{\text{max}}$
(2) An update with data that violates $S_{\text{max}}$
(3) An extraction error presented to user

We provide: (1) explanation of results
   (2) provenance
   (3) Warn the analyst with ambiguity
   (4) Explain the ambiguity
   (5) Evaluate the magnitude of ambiguity
   (6) Assist the analyst to resolve the ambiguity
Types of errors

ASD interacts with the outside world: Schema, Data, and Update.

Schema interactions: When a query incompatible with $S_{\text{max}}$ and the workspace

Data interactions: provenance for attribute and row level ambiguity.

Update interactions:
• represent schema mismatches as missing values.
• resolve data errors with a probabilistic repair.
• upgrade her schema to match the changes.
• checkpoint her workspace and ignore new updates.
Explanations and feedback

Condition 2: Query from **unknown** schema elements:

SELECT name FROM Student

W_1 = (S_1={Undergrad(name)}, P_1=0.27),
     (S_1={Grad(name)}, P_1=0.23),
     (S_1={Undergrad(name), Grad(name)}, P_1=0.5)

Explanations:
We match Student with both Grad and Undergrad
Adaptive organization

- Unstructured Data
- Semi-structured Data (e.g., JSON)

Extraction workflow

Schema Workspace

Schema Matching

Extraction Schema Candidates

Extraction workflow

Unstructured Data

Semi-structured Data (e.g., JSON)
Adaptive organization

Trade-off between storing data in its native format and based on a specific schema.

What is the challenge? Many workspaces, add table to the schema, ....

Challenges and Possible Solutions:
• We want multiple personalized schemas
  1. Relational workspace schema is essentially a view over raw data. Materializing view can be used.
  2. Use existing adaptive physical design and caching techniques.
• Shared materializations
  1. Incremental materialized view maintenance. Leverage techniques from revision control systems.
Conclusions and future work

ASD bridges the gap between relational databases and NoSQL.

- **Discovery**: Help user explore and understand new data by providing an outline of the available information. *Done*
- **Materialization**: Adopt work on adaptive data structures. *Partially done*
- **Data Synthesis**: Synthesis new tables and attributes from existing data. *Done*
- **Conflict Response**:
  - Versioning or branching the schema.
  - Log analysis to help users assess the impact of schema revisions.