Variational Databases

Parisa Ataei, Arash Termehchy, Eric Walkingshaw

Oregon State University

Each software has many variations

```
typedef struct T_node {
  int item;
  struct T_node *next;
#if DLINKED
  struct T_node *prev;
#endif
} node;

node *first = NULL;
#if DLINKED
node *last = NULL;
#endif

void insert(node *elem) {
#if SORTALGO == BUBBLESORT || SORTALGO == INSERTIONSORT
  node *a = NULL;
  node *b = NULL;
#endif
#if SORTALGO == BUBBLESORT
  node *c = NULL;
  node *e = NULL;
  node *tmp = NULL;
#endif
  if (NULL == first) first = elem;
  else {
#if SORTALGO == INSERTIONSORT
    a = first;
    b = first->next;
    if (first->item
#if SORTORDER == 0
      >
#else
      <
#endif
...
#if SORTALGO == BUBBLESORT
...
#endif
}
```

- Reasons:
  - The hardware environment
  - The client requirements
  - The clients' geographical place
  - ....

```
#if LINUX || MAC
  newline = "\n";
#elif WINDOWS
  newline = "\r\n";
#else
  error("Unknown OS!");
#endif
```

Software Product Line (SPL)

- Producing many variations of a software system
- Active research area in the programming languages and software engineering communities.
- It provides a structured approach to produce many variations of a software in similar contexts.
- It saves time and resources.

Software Product Line

BOEING

Bold Stroke Avionics

TELVENT

Industrial supervisory control and business process management systems

NASA

Interferometer product line

GM

Software for engines, transmissions and controllers

E-COM Technology Ltd.

Medical imaging workstations

Variations are configured by a set of *features*

```
typedef struct T_node {
  int item;
  struct T_node *next;
#if DLINKED
  struct T_node *prev;
#endif
} node;

node *first = NULL;
#if DLINKED
node *last = NULL;
#endif

void insert(node *elem) {
#if SORTALGO == BUBBLESORT || SORTALGO == INSERTIONSORT
  node *a = NULL;
  node *b = NULL;
#endif
#if SORTALGO == BUBBLESORT
  node *c = NULL;
  node *e = NULL;
  node *tmp = NULL;
#endif
  if (NULL == first) first = elem;
  else {
#if SORTALGO == INSERTIONSORT
    a = first;
    b = first->next;
    if (first->item
#if SORTORDER == 0
      >
#else
      <
#endif
...
#if SORTALGO == BUBBLESORT
...
#endif
}
```

```
#if LINUX || MAC
  newline = "\n";
#elif WINDOWS
  newline = "\r\n";
#else
  error("Unknown OS!");
#endif
```

- Each set of features creates a variation of software for a setting, group of users, …
- **Example:** installing Linux.

Oregon State University

Managing software variations is challenging

- Typically many features in each SPL

Software	Number of features
Linux	9,102
python	5,127
sqlite	292
opensolaries	10,901

- There are generally exponentially many software variations.

- **Challenge**: *how can one test a functionality over all these variations*?

  - **Solution:** Remove or reduce the degree of variation

  - **Example:** factoring out shared pieces of code among several variations.

Oregon State University

Our focus: variation in database-backed software

- Many software systems needs to collect, store, and manipulate data in one form or another.
- To the best of our knowledge, not much work on the data variability arose in the context of SPL and software production.
- **Example:** consider an SPL that produces banking softwares for clients around the globe.
  - Needs a simple table to store the names of members (customers)
  - One feature: **country**

Impact of feature variations on schema design

member

ID	FirstName	MiddleName	LastName
1	Hank	Joe	Eason
2	Caitlin	Mary	Newport
3	Sean	John	Patrik

```
#if COUNTRY == US
    memberCreateQuery = "CREATE TABLE member(ID INT,
      FirstName VARCHAR(20), MiddleName VARCHAR(20),
       LastName VARCHAR(20));"
```

```
#elif COUNTRY == Iceland
    memberCreateQuery = "CREATE TABLE member(ID INT,
      FirstNameVARCHAR(20), Father'sName VARCHAR(20),
      Gender CHAR(1));"
#elif ...
```

member

ID	FirstName	Father'sName	Gender
1	Sara	Sigmund	F
2	Erla	Helga	F
3	karl	Gudmund	M

member

ID	FirstName	FamilyName
1	Leila	Ranjbar
2	Hamid	Adami
3	Mani	Hamidi

```
#elif COUNTRY == Iran
    memberCreateQuery = "CREATE TABLE member(ID INT,
      FirstName VARCHAR(20), FamilyName VARCHAR(20));"
```

Impact of feature variations on database querying

- We want to find members with a same name.
- Query: return members with the same name.

$$\boxed{A(x,y,z) :- member(a,x,y,z), member(b,x,y,z)}$$

Schema variation 1

$$\boxed{\begin{array}{l} A''(x, concat(z,'sdottir')) :- member(a,x,z,'F'), member(b,x,z,'F') \\ A''(x, concat(z,'sson')) :- member(a,x,z,'M'), member(b,x,z,'M') \end{array}}$$

Schema variation 2

$$\boxed{A'(x,z) :- member(a,x,z), member(b,x,z)}$$

Schema variation 3

Oregon State University

Challenges

- Developing, testing, and managing various schemas and queries.
    - A different query for each variation.
- Remember that we’re only showing one feature and its impact.
    - There are generally many such features.

Current approach

- Design a schema that contains all relations with all possible attributes and possibly using views.
- Shortcomings:
  - Having a lot of null values
  - View-updating problem
  - The developer has to deal with a large number of heterogeneous attributes in a single table.
  - The large schema may increase the running times of queries.

Current approach example

member relation in US

ID	FirstName	MiddleName	LastName	FamilyName	Father'sName	Gender
1	Hank	Joe	Eason	null	null	null
2	Caitlin	Mary	Newport	null	null	null
3	Sean	John	Patrik	null	null	null

member relation in Iceland

ID	FirstName	MiddleName	LastName	FamilyName	Father'sName	Gender
1	Sara	null	null	null	Sigmund	F
2	Erla	null	null	null	Helga	F
3	karl	null	null	null	Gudmund	M

member relation in Iran

ID	FirstName	MiddleName	LastName	FamilyName	Father'sName	Gender
1	Leila	null	null	Ranjbar	null	null
2	Hamid	null	null	Adami	null	null
3	Mani	null	null	Hamidi	null	null

Our proposal: *Variational Databases*
Variational Schema

- An abstract schema, **variational schema**, that compactly represents a set of different schemas.
  - Configuring the features for each use-case generates a plain schema.
- It helps by:
  - preventing null values, dirty data
  - having a desired schema without the need to deal with views

Oregon State University

Variational query

- A variational query represents different ways of expressing an information need over a variational schema.
  - Configuring the features for each use case generates a plain query over the correspondent schema .
- It will make it easier to detect and remove unnecessary variations:
  - Check whether two variational queries are equivalent.
- It will make it easier to factor out commonalities across variations.
  - Check whether two variational queries are superset/ subset.
- It may reduce the amount of developer's effort by lifting a plain query to a variational query.

Oregon State University

Variational relation

- Variational set: $\vec{S} = \left\{ e_1^{c_1}, \ldots, e_n^{c_n} \right\}$
- One may define a variational map using a variational set.

$$\overset{\mapsto}{M}(k_i) = \vec{V}_i$$

- Each variational relation is a variational map from a relation to a set of attributes.
  - Configurations are determined by features
- A schema is a variational set of relations.

Variational schema example

ID	FirstName	MiddleName	LastName
1	Hank	Joe	Eason
2	Caitlin	Mary	Newport
3	Sean	John	Patrik

Schema variation 1

ID	FirstName	FamilyName
1	Leila	Ranjbar
2	Hamid	Adami
3	Mani	Hamidi

Schema variation3

ID	FirstName	Father'sName	Gender
1	Sara	Sigmund	F
2	Erla	Helga	F
3	karl	Gudmund	M

Schema variation 2

Variational schema example
The variational schema

$$\left(\vec{N},\vec{S}\right), s.t.$$

$$\vec{N} = \begin{Bmatrix} 1 \mapsto \{ID\}, 2 \mapsto \{FirstName\}, 3 \mapsto \left\{MiddleName^{U}\right\}, \\ 4 \mapsto \left\{LastName^{U}, FamilyName^{I}, Father'sName^{C}\right\}, 5 \mapsto \left\{Gender^{C}\right\} \end{Bmatrix},$$

$$\vec{S} = \left\{member\{1,2,3,4,5\}\right\}$$

Schema variation 2

Variational query

- A compact way of showing all possible queries of a plain query over variations of a variational schema in one query

- A query that can be executed over any variation defined by the variational schema

$$
\begin{aligned}
f' \in \mathit{VFormula} &::= R\,\{i_1 : a_1^{c_1}, \ldots, i_n : a_n^{c_n}\} \mid a_1 \bullet a_2 \\
g' \in \mathit{Goal} &::= f' \mid \text{not } f' \\
r' \in \mathit{Rule} &::= f' \text{ :- } g_1'^{c_1}, \ldots, g_n'^{c_n} \\
q' \in \mathit{Query} &::= f' \texttt{ with } r_1'^{c_1}, \ldots, r_n'^{c_n}
\end{aligned}
$$

Variational query example

- Return members with the same name.

$$\boxed{A(x,y,z):-member(a,x,y,z),member(b,x,y,z)}$$

Schema variation 1

$$\boxed{A'(x,z):-member(a,x,z),member(b,x,z)}$$

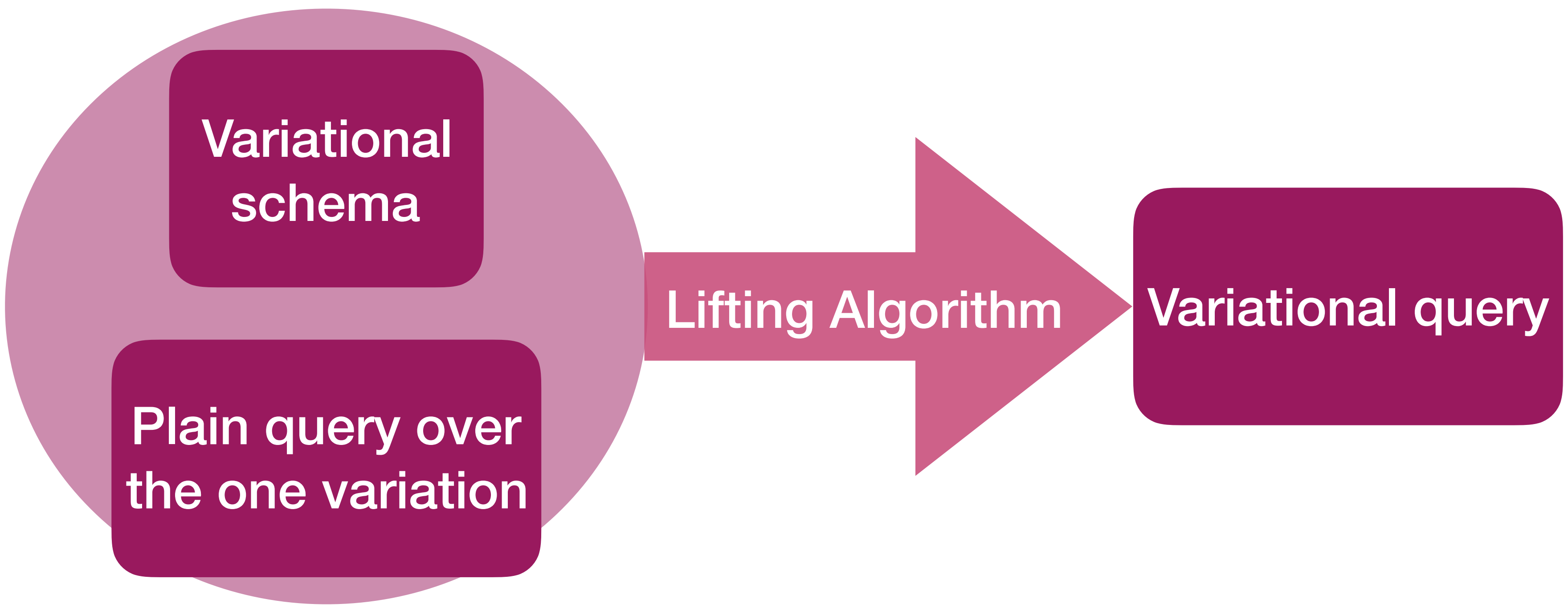
Schema variation 3

$$\boxed{\begin{array}{l} A\{2:x,3:y,4:z\}:-member\{1:a,2:x,3:y,4:z\}, \\ member\{1:b,2:x,3:y,4:z\} \end{array}}$$

Variational query

Oregon State University

Lifting plain queries to variational queries



- Not always possible

Lifting Queries to Variational Queries Example

$$A(x,y,z):-member(a,x,y,z),member(b,x,y,z)$$

Query over variation 1

$$\left(\vec{N},\vec{S}\right),s.t.$$

$$\vec{N}=\left\{\begin{matrix}1\mapsto\{ID\},2\mapsto\{FirstName\},3\mapsto\{MiddleName^{U}\},\\4\mapsto\{LastName^{U},FamilyName^{I}\}\end{matrix}\right\},$$

$$\vec{S}=\{member\{1,2,3,4\}\}$$

Variational schema

Lifting Algorithm

$$A\{2:x,3:y,4:z\}:-member\{1:a,2:x,3:y,4:z\},$$
$$member\{1:b,2:x,3:y,4:z\}$$

Variational query

Oregon State University

Conclusion

- Schema and query variability arise when producing software in SPLs.
- SPL developers have to deal with a great number of variations in database-related operations.
- Variational schema and query provide a systematic approach to reducing the amount of variability.
- We plan to design and implement a general variational algebra for variational query language.
  - Feedback and suggestion welcome! termehca@oregonstate.edu

Oregon State University