Query language support for incomplete information in the MayBMS system

Lyublena Antova, Christoph Koch, and Dan Olteanu
Saarland University Database Group
Saarbrücken, Germany
{lublena, koch, olteanu}@infosys.uni-sb.de

1. INTRODUCTION

MayBMS [4, 1, 3, 2] is a data management system for incomplete information developed at Saarland University. Its main features are a simple and compact representation system for incomplete information and a language called I-SQL with explicit operations for handling uncertainty. MayBMS is currently an extension of PostgreSQL 8.2.3 and can manage both complete and incomplete data and evaluate I-SQL queries.

The focus of the demonstration is I-SQL. I-SQL is a natural extension of SQL to the context of incomplete information. Like SQL, I-SQL is a generic language in that it preserves the independence of the data from its representation. Thus the answers to I-SQL queries do not depend on details of how the data is stored. I-SQL proves expressive enough for a variety of application scenarios, where incompleteness is ubiquitous: planning, design, business decision-making, data cleaning [2], and, as exemplified in Section 3, monitoring moving objects when only partial information is available, like satellite tracking of whales. Differently from SQL, I-SQL has explicit operations for dealing with uncertainty. Extensions of SQL with limited operations, such as *certain* or *top-k*, that close the possible worlds semantics are not expressive enough, as they do not allow for the convenient construction of new worlds or for the use of data correlations across worlds.

2. I-SQL BY EXAMPLES

We next exemplify the I-SQL operations using the complete database of Figure 1 and the world-set of Figure 2.

The evaluation of an I-SQL query follows the possible worlds semantics by which the query is evaluated in each world independently, and the world is extended with the result of the query in it. This also applies to SQL queries and updates. For example, a tuple insertion statement will insert the tuple in each world of the world-set. In case the tuple insertion violates a constraint in some worlds, then the update is discarded in all worlds.

VLDB '07, September 23-28, 2007, Vienna, Austria.

EXAMPLE 2.1. Given the world-set of Figure 2, we want to delete all tuples of I with an A-value of 3:

```
delete from I where A = a3;
```

The answer is a copy of the input world-set, where each world also contains the query answer representing I except its last tuple. This world-set copy is not stored. □

The operation create table is used to materialize the set of worlds created by I-SQL queries.

EXAMPLE 2.2. To store the result of the deletion operation, we use:

```
create table D as delete from I where A = a3;
```

This statement modifies the input world-set by adding the result of deletion to each of the four worlds. □

The repair-by-key operation is motivated by data cleaning scenarios. When applied to a relation that violates a uniqueness constraint, a repair-by-key query generates a world-set representing all possible repairs of that relation.

EXAMPLE 2.3. Figure 2 shows a relation I, whose four instances represent the four possible repairs of R on the key attribute A:

```
create table I as
select A, B, C from R repair by key A;
```

Although not shown here, each world also contains all relations of the world from which it originated. In our case, the relations R and S are contained in each created world.□

The repair operation has an optional *weight* construct that can be used to assign probabilities to the created worlds.

EXAMPLE 2.4. We rephrase the query of Example 2.3 so as to compute the probabilities of each repair based on the weight of the positive numbers representing the D-values occurring in that repair:

```
create table I as
select A, B, C from R repair by key A weight D;
```

The probability of a world, say $\mathcal{A}$, is computed based on the choice of its tuples. For each different A-value a_i, we weight the D-value, which occurs in the same tuple with a_i, with respect to the sum of all D-values for the same A-value in R (this makes sense, of course, if all D-values are numbers

R	A	B	C	D
	a_1	10	c_1	2
	a_1	15	c_2	6
	a_2	14	c_3	4
	a_2	20	c_4	5
	a_3	20	c_5	6

S	C	E
	c_2	e_1
	c_4	e_1
	c_4	e_2

Figure 1: Complete database of two relations R and S.

$I^{\mathcal{A}}$	A	B	C
	a_1	10	c_1
	a_2	14	c_3
	a_3	20	c_5

$P(\mathcal{A})$=0.11

$I^{\mathcal{B}}$	A	B	C
	a_1	15	c_2
	a_2	14	c_3
	a_3	20	c_5

$P(\mathcal{B})$=0.33

$I^{\mathcal{C}}$	A	B	C
	a_1	10	c_1
	a_2	20	c_4
	a_3	20	c_5

$P(\mathcal{C})$=0.14

$I^{\mathcal{D}}$	A	B	C
	a_1	15	c_2
	a_2	20	c_4
	a_3	20	c_5

$P(\mathcal{D})$=0.42

Figure 2: Each world represents one repairing of the key attribute A of R.

greater than zero). The probability of $\mathcal{A}$ is then the product of the weights for its choice of tuples:

$$\frac{2}{2+6} \cdot \frac{4}{4+5} \cdot \frac{6}{6} = 0.11.$$

The probabilities of the other worlds are given in Figure 2.□

The assert operation is used to keep only those input worlds that satisfy the assert condition. The worlds that do not satisfy this condition are dropped.

Example 2.5. Consider the four worlds of Figure 2. To keep only those worlds (repairs) that do not have c_1 as a C-value, we use assert:

```
create table J as select * from I
  assert not exists(select * from I where C = c1);
```

The result of this query is a relation J that equals I in the worlds $\mathcal{B}$ and $\mathcal{D}$, respectively. The remaining two worlds $\mathcal{A}$ and $\mathcal{C}$ are dropped, as they contain instances of I with tuples having c_1 as C-value. □

In the probabilistic case, the probabilities of the remaining worlds are uniformly normalized such that they sum up to one. In the previous example, after normalization we obtain $P(\mathcal{B}) = 0.44$ and $P(\mathcal{D}) = 0.56$.

The operation choice-of creates new worlds based on existing attribute values. Given a set U of attributes of a relation $R^{\mathcal{A}}$ in a world $\mathcal{A}$, choice-of creates one world $\mathcal{A}_i$ for each different U-value u_i. Each world $\mathcal{A}_i$ contains all relations of $\mathcal{A}$ and also the relation consisting of all tuples in $R^{\mathcal{A}}$ with the same U-value u_i.

Example 2.6. For the database of Figure 1, the query

```
select * from S choice of E;
```

creates a set of two worlds corresponding to disjoint partitions of S: a world contains the relation made out of those S-tuples with an E-value of e_1 and e_2, respectively. Both worlds also contain the relations R and S. □

Like in the case of repair-by-key, choice-of can *weight* the probabilities of the created worlds by values of an attribute.

Example 2.7. The following choice-of query creates three worlds, whose probabilities are weighted by the column D:

```
select * from R choice of A weight D;
```

There is a world for each distinct A-value of R. The probability of a world is given by the sum of the D-values of the tuples in that world over the sum of all D-values in R. Thus, the worlds for values a_1, a_2, and a_3 have probabilities 0.35, 0.39, and 0.26 respectively. □

I-SQL has two constructs possible and certain that go across world borders to collect information that appears in other worlds as well. The result of such operations is then added to each world of the input world-set.

Example 2.8. Consider a query that sums up the B-values in each world of Figure 2:

```
select sum(B) from I;
```

The answer is $\{(44)\}$ for world $\mathcal{A}$, $\{(49)\}$ for $\mathcal{B}$, $\{(50)\}$ for $\mathcal{C}$, and $\{(55)\}$ for $\mathcal{D}$. In order to compute the *set of possible* sums of the B-values, we modify the query as follows:

```
select possible sum(B) from I;
```

In contrast to the query variant without possible, the answer to this query is the relation $\{(44), (49), (50), (55)\}$. □

Example 2.9. The following query computes the set of E-values that occur with each different C-value in S:

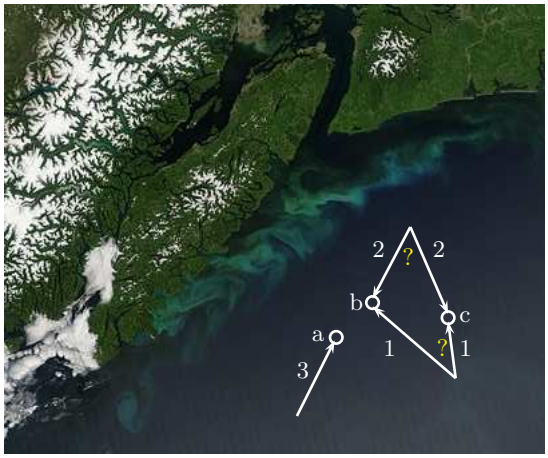
```
select certain E from S choice of C;
```

For each different C-value of S, this query creates a new world. Relation S of Figure 1 has two different C-values (c_2 and c_4), and e_1 is the only E-value that occurs with both. The answer relation is thus $\{(e_1)\}$. □

The operation conf is used to compute the confidence of tuples. The confidence of a tuple t of a relation I is the sum of probabilities of all worlds, in which I contains t. The operations possible and certain can be expressed as straightforward conditions on conf: a tuple is possible if its confidence is greater than zero and certain if its confidence is one.

Example 2.10. We would like to know the confidence that the sum of the B-values is under 50:

```
select conf from I
            where 50 > (select sum(Time) from I);
```



$I^{\mathcal{A}}$

Id	Genus	Gender	Pos
1	sperm	calf	b
2	sperm	cow	c
3	orca	cow	a

$I^{\mathcal{B}}$

Id	Genus	Gender	Pos
1	sperm	calf	b
2	sperm	cow	c
3	orca	bull	a

$I^{\mathcal{C}}$

Id	Genus	Gender	Pos
1	sperm	calf	b
2	sperm	bull	c
3	orca	cow	a

$I^{\mathcal{D}}$

Id	Genus	Gender	Pos
1	sperm	calf	b
2	sperm	bull	c
3	orca	bull	a

$I^{\mathcal{E}}$

Id	Genus	Gender	Pos
1	sperm	calf	c
2	sperm	cow	b
3	orca	cow	a

$I^{\mathcal{F}}$

Id	Genus	Gender	Pos
1	sperm	calf	c
2	sperm	bull	b
3	orca	cow	a

Figure 3: Whale tracking information on top of a satellite photograph of the coast of Vancouver Island.

The answer is $\{(0.53)\}$. The value 0.53 is computed as 0.11+0.42 and represents the sum of probabilities of the worlds $\mathcal{A}$ and $\mathcal{D}$ that satisfy the where-condition. □

The possible and certain operations can be used in combination with a grouping construct. The group-worlds-by operation allows to only look at worlds that are similar to the current world in some predefined sense. For example, a group is formed by those worlds in which a given query has the same answer. Then, possible or certain are computed within each of the created groups.

3. DEMONSTRATION SCENARIOS

We next describe two application scenarios for I-SQL: whale tracking and cleaning of inconsistent data.

3.1 Tracking whales

We consider an application for tracking whales using photographs taken from satellites[1]. Figure 3 shows a photograph from visibleearth.nasa.gov with added whale tracking information. Our photograph reports schematically on the movement of three whales, cataloged by dimensions as two sperm whales (with ids 1 and 2) and an orca whale (with id 3). As depicted, the orca moves towards the two sperm whales (position a), which move towards each other (positions b and c). The information gathered from our observations is represented in Figure 3 by a relation I in six worlds. (Gender encodes whether a whale is a cow, bull, or calf.)

We would like to know if there is an possibility that the adult orca whale attacks the calf sperm whale. This would happen if, for example, the calf moves to position b, which is near position a. This possibility for an attack is encoded as the following query Q:

```
select possible 'yes' from I where Id=1 and Pos=b;
```

The answer is 'yes' because in the worlds $\mathcal{A}$ through $\mathcal{D}$ the calf sperm whale moves to position b.

While inspecting the photograph, we gained additional expert knowledge that sperm cows tend to protect their calves against orca predators by positioning themselves between their calves and the enemy. Also, it would have taken longer for the calf than for the cow to reach position b. We would like to reconsider our question using this additional knowledge. To express our question, we first define, for reasons of simplicity, a view that represents those worlds that do not contradict the additional knowledge:

```
create view Valid as
 select * from I assert exists
         (select * from I
                   where Gender=cow and Pos=b);
```

The view Valid represents the copy of relation I in the world $\mathcal{E}$. The other worlds contradict the additional knowledge and are thus dropped. We are now left with one world.

Clearly, our query Q asking for the possibility that the calf moves to position b returns an empty answer on Valid:

```
select possible 'yes' from Valid
                      where Id=1 and Pos=b;
```

A different approach is to define a new relation that equals I in those worlds that have a cow sperm whale moving to position b and empty otherwise:

```
create view Valid' as
select * from I where exists
         (select * from I
                   where Gender=cow and Pos=b);
```

In contrast to Valid, Valid' defines a relation for all of the input *six* worlds: it equals I for world $\mathcal{E}$ and is empty for the remaining worlds. The two views represent thus different world-sets. This plays no difference, however, for our query Q: its answer is the same on Valid and on Valid'. If we are interested in the certain tuples in the two views

```
select certain * from Valid;
select certain * from Valid';
```

we obtain different answers: the answer is $I^{\mathcal{E}}$ for Valid and empty for Valid'.

A further question we would like to ask is whether the adult sperm whale moves to positions b or c *independently* of his gender of whether the orca is a cow or an aggresive

[1]Ocean Alliance (www.oceanalliance.org), among many other organizations, uses whale tracking to study the feeding and social behavior of orcas and sperm whales.

Groups$^{\mathcal{A}-\mathcal{D}}$	G_2	G_3
	cow	cow
	cow	bull
	bull	cow
	bull	bull

Groups$^{\mathcal{E},\mathcal{F}}$	G_2	G_3
	cow	cow
	bull	cow

Figure 4: Possible combinations of genders of the two adult whales.

R	SSN	TEL
	123	456
	789	123

S	SSN	TEL	SSN'	TEL'
	123	456	123	456
	123	456	456	123
	789	123	789	123
	789	123	123	789

Figure 5: Social security numbers and phone numbers (R) and their possible permutations (S).

bull. (That is, we look for signs that the adult sperm whale may abandon the calf in a dangerous situation in order to save itself.) To answer this question, we first define groups of worlds corresponding to the different cases where the adult sperm whale moves to position b or to position c.

```
create table Groups as
select possible i2.G as G2, i3.G as G3
from   I i2, I i3
where i2.Id = 2 and i3.Id = 3
group worlds by (select Pos from I where Id = 2);
```

The answer to the nested SQL query is $\{(c)\}$ in the worlds $\mathcal{A}$ through $\mathcal{D}$ and $\{(b)\}$ in the remaining worlds $\mathcal{E}$ and $\mathcal{F}$. For each different answer we create a group consisting of the worlds in which the answer is the same. Within each such group, we evaluate the possible-query and obtain the possible combinations of genders of the adult sperm and orca whales. Within each group, the corresponding relation Groups is then added to each of the worlds in that group. The instances of Groups are shown in Figure 4.

We can check that the genders of the two adult sperm whales are indeed independent in both instances of relation Groups. Thus there is no particular correlation between the genders of the adult whales that would point to an abandon of the calf. This check can be expressed in standard SQL (thus in each world) by testing whether

$$\text{Groups} = \pi_{G_2}(\text{Groups}) \times \pi_{G_3}(\text{Groups}).$$

3.2 Data cleaning by constraints and queries

I-SQL offers support for cleaning dirty (complete or incomplete) databases via an interplay of integrity constraint-based and query-based cleaning. The former cleaning is supported by operators like choice-of, repair-by-key, and assert that can be naturally used to enforce constraints on inconsistent databases and create a set of possible consistent (repaired) databases.

Consider the complete relation R of Figure 5 containing information on social security numbers and phone numbers that have possibly been confused (swapped). We would like to clean R by considering all possible pairs of social security numbers and phone numbers that satisfy the uniqueness constraint for the social security number (SSN).

We proceed as follows. We first express our assumption

$T^{\mathcal{A}}$	SSN'	TEL'
	123	456
	789	123

$T^{\mathcal{B}}$	SSN'	TEL'
	123	456
	123	789

$T^{\mathcal{C}}$	SSN'	TEL'
	456	123
	789	123

$T^{\mathcal{D}}$	SSN'	TEL'
	456	123
	123	789

Figure 6: The four possible readings of social security numbers and phone numbers.

$U^{\mathcal{A}}$	SSN'	TEL'
	123	456
	789	123

$U^{\mathcal{C}}$	SSN'	TEL'
	456	123
	789	123

$U^{\mathcal{D}}$	SSN'	TEL'
	456	123
	123	789

Figure 7: Worlds that satisfy the functional dependency SSN' → TEL'.

that any number in R can potentially be a social security number or a phone number:

```
create table S as
select SSN, TEL, SSN as SSN', TEL as TEL' from R
union
select SSN, TEL, TEL as SSN', SSN as TEL' from R;
```

The relation S is given in Figure 5. Now, we consider all possible readings of the two records by repairing the key SSN,TEL of S:

```
create table T as
select SSN', TEL' from S repair by key SSN, TEL;
```

Figure 6 shows the four possible worlds of T. Among the four worlds, the world $\mathcal{B}$ does not satisfy the uniqueness constraint for the social security numbers. This world can be dropped by enforcing the functional dependency SSN' → TEL' in T using an assert condition:

```
create table U as
select * from T assert not exists
  (select 'yes' from T t1, T t2
   where t1.SSN' = t2.SSN' and t1.TEL' <> t2.TEL');
```

Figure 7 shows the remaining three worlds resulted after enforcing the functional dependency.

4. REFERENCES

[1] L. Antova, C. Koch, and D. Olteanu. "10^{10^6} Worlds and Beyond: Efficient Representation and Processing of Incomplete Information". In *Proc. ICDE*, 2007.

[2] L. Antova, C. Koch, and D. Olteanu. "From Complete to Incomplete Information and Back". In *Proc. SIGMOD*, 2007.

[3] L. Antova, C. Koch, and D. Olteanu. "MayBMS: Managing Incomplete Information with Probabilistic World-Set Decompositions". In *Proc. ICDE*, 2007. Demonstration Paper.

[4] L. Antova, C. Koch, and D. Olteanu. "World-set Decompositions: Expressiveness and Efficient Algorithms". In *Proc. ICDT*, 2007.