On the Expressive Power of the Relational Algebra with Partially Ordered Domains

Wilfred Ng * Mark Levene † Trevor I. Fenner ‡

Abstract

Assuming data domains are partially ordered, we define the partially ordered relational algebra (PORA) by allowing the ordering predicate $\sqsubseteq$ to be used in formulae of the selection operator σ. We apply Paredaens and Bancilhon's Theorem to examine the expressiveness of the PORA, and show that the PORA expresses exactly the set of all possible relations which are invariant under order-preserving automorphisms of databases. The extension is consistent with the two important extreme cases of unordered and linearly ordered domains. We also investigate the three hierarchies of: (1) computable queries, (2) query languages and (3) partially ordered domains, and show that there is a one-to-one correspondence between them.

Key Words: partially ordered domains, relational algebra, relational databases.

C.R. Categories: F.4.3, H.2.1

1 Introduction

Many naturally arising data types have an associated structure, of which domain ordering is the most important one [4, 7]. Moreover, all existing relational database systems support linearly ordered domains, e.g., the integers with the usual ordering. Given a fixed database instance, the impact of order on the expressive power is important since it widens the set of allowable queries with respect to the instance. Therefore, with the assumption of partially ordered domains, we examine two main issues: (1) the extra expressiveness of the relational algebra that we can gain with respect to a database instance, and (2) the relationship between ordered domains and classes of queries.

We extend the relational algebra by allowing the predicate $\sqsubseteq$ in selection formulae, thus obtaining the Partially Ordered Relational Algebra (PORA). Our first result, which is a generalisation of Paredaens and Bancilhon's Theorem [2, 8], shows that the PORA expresses exactly the set of relations which are invariant under order-preserving automorphisms. In the special case of unordered domains (i.e., where any two distinct data elements are incomparable under the ordering), the PORA reduces to the standard relational algebra and gives the same result as Paredaens and Bancilhon's Theorem. In the other special case of linearly ordered domains, the PORA expresses exactly the countably infinite set of all possible relations on the active domain of a given database. Our second result, which involves the notion of a meaningful computable query with respect to a given ordered domain, shows that there is a one-to-one correspondence between hierarchies of (1) meaningful computable queries, (2) partially ordered relational algebras and (3) partially ordered domains.

*Department of Computing, Hong Kong Polytechnic University, email: csshng@comp.polyu.edu.hk

†Department of Computer Science, University College London, University of London, email: m.levene@cs.ucl.ac.uk

‡Department of Computer Science, Birkbeck College, University of London, email: trevor@dcs.bbk.ac.uk

2 Ordered Domains and Ordered Databases

For any set S, $\mathcal{P}(S)$ denotes the *finite powerset* of S and S^k denotes the k term Cartesian product $S \times S \times \cdots \times S$. When no ambiguity arises, $\{A\}$ may be written as A, and id denotes the identity mapping on any set.

As usual a *partial ordering* $\sqsubseteq$ on a set S is a binary relation on S which is *reflexive*, *anti-symmetric* and *transitive* [5]. When $\sqsubseteq$ is a *linear ordering*, we denote it by $\leq$. When each element is only comparable with itself, S is *unordered*, i.e., $\sqsubseteq$ is just the equality predicate $=$. The structure $\langle S, \sqsubseteq \rangle$ is called a *partially ordered set*. (We use the term *ordered* to mean partially ordered unless stated to the contrary). Furthermore, if $T \subseteq S$ then we will consider $\langle T, \sqsubseteq \rangle$ to be an ordered set, interpreting $\sqsubseteq$ as its restriction to $T \times T$.

Let D be a countably infinite set of constants and $\sqsubseteq$ an ordering on D. We assume a countably infinite set U of attribute names which all share the same domain $\langle D, \sqsubseteq \rangle$.

A *(relation) schema* $R = \{A_1, \ldots, A_m\}$ is a non-empty finite subset of U. A *database schema* $\mathbf{R}$ is a non-empty finite set of relation schemas. A *tuple* t over R is a member of D^m, and $t[A_i]$ denotes the ith coordinate of t, i.e., the *projection* of t onto the attribute A_i. A *relation* r defined over R is a finite set of tuples over R. A *database* over $\mathbf{R} = \{R_1, \ldots, R_n\}$ is a finite set of relations $d = \{r_1, \ldots, r_n\}$ where (by convention) r_i is a relation over R_i for each i. We sometimes call r and d an *ordered* relation and an *ordered* database, respectively, to indicate that the underlying domain D is ordered; similarly for an *unordered* or *linearly ordered* relation and database. *We assume throughout* that $\mathbf{R}$ and d are defined as above, and we let $DB(\mathbf{R})$ denote the countably infinite set of all databases over $\mathbf{R}$.

Without loss of generality, we restrict our discussion to the *active domain*, $adom(d)$, which contains only those values that appear in the database instance d; obviously $adom(d)$ is ordered by $\sqsubseteq$. For a relation r, $adom(r)$ is defined similarly. $Poss(d)$, the *possible information* of d, is the countably infinite set of all relations that can be derived from $adom(d)$, viz. $\bigcup_{i=0}^{\infty} \mathcal{P}(adom(d)^i)$.

We now generalise the notion of automorphism [1] to ordered databases. Informally, an *ordering automorphism* of an ordered set is a permutation of its elements such that the ordering of the set is preserved.

Let $\langle S, \sqsubseteq \rangle$ be an ordered set. The function $f : S \to S$ is an *ordering automorphism* of $\langle S, \sqsubseteq \rangle$ if f is bijective and $a_1 \sqsubseteq a_2$ if and only if $f(a_1) \sqsubseteq f(a_2)$. If the set $\{a \in S \mid f(a) \neq a\}$ is finite, then f is a *finite* ordering automorphism. We denote the set of all finite ordering automorphisms of $\langle S, \sqsubseteq \rangle$ by $Aut(S, \sqsubseteq)$.

If $h \in Aut(adom(d), \sqsubseteq)$, we extend h to tuples t, relations r and databases d in the obvious way. Then h is an *order-preserving automorphism* of d if its extension to d satisfies $h(d) = d$, viz. $h(r_i) = r_i$ for $1 \leq i \leq n$; i.e., h is a permutation of the active domain that preserves the ordering and does not alter the database. We denote the set of all order-preserving automorphisms of d by $Aut(\sqsubseteq, d)$. It follows from the definitions that, for all partial orderings $\sqsubseteq$,

$$Aut(\sqsubseteq, d) = Aut(adom(d), \sqsubseteq) \cap Aut(=, d).$$

Also, for all linear orderings $\leq$, $Aut(\leq, d) = \{id\}$.

Example 1 Let d contain just a single relation $r = \{xz, yz, xw, yw\}$, and let $\langle adom(d), \sqsubseteq \rangle = \langle \{w, x, y, z\}, \{x \sqsubseteq y, x \sqsubseteq z, x \sqsubseteq w\} \rangle$. We define functions: h_1 by $h_1(x) = y$, $h_1(y) = x$, $h_1(z) = z$ and $h_1(w) = w$; h_2 by $h_2(x) = x$, $h_2(y) = z$, $h_2(z) = y$ and $h_2(w) = w$; and h_3 by $h_3(x) = x$, $h_3(y) = y$, $h_3(z) = w$ and $h_3(w) = z$. Then $h_1 \notin Aut(\sqsubseteq, d)$ because $h_1 \notin Aut(adom(d), \sqsubseteq)$, although $h_1(d) = d$; and $h_2 \notin Aut(\sqsubseteq, d)$ because $h_2(d) \neq d$, although $h_2 \in Aut(adom(d), \sqsubseteq)$; however, $h_3 \in Aut(\sqsubseteq, d)$ because $h_3 \in Aut(adom(d), \sqsubseteq)$ and $h_3(d) = d$.

3 The PORA and its Expressive Power

In this section we extend Paredaens and Bancilhon's Theorem to ordered databases. We first extend the classical relational algebra to the partially ordered relational algebra (PORA) – a collection of six operators each of which maps a set of relations to a relation.

Definition 3.1 The *PORA* is a collection of the following six operators: *union* ($\cup$), *Cartesian product* ($\times$), *difference* ($-$), *projection* (π_X), where $X \subseteq U$ is finite, *renaming* ($\rho_{X \to Y}$), where $X \to Y$ is a bijective function from a finite set $X \subseteq U$ to a finite set $Y \subseteq U$, and lastly *extended selection* (σ_F), where the selection formula F is restricted to be one of the forms $A = B$, $A \neq B$, $A \sqsubseteq B$ or $A \not\sqsubseteq B$, where A, $B \in U$.

These are the standard six operators (see [1, 6] for formal definitions and semantics); the meaning of $\sigma_{A \sqsubseteq B}$ is as expected: $\sigma_{A \sqsubseteq B}(r) = \{t \in r \mid t[A] \sqsubseteq t[B]\}$.

A *PORA expression* is a well-formed expression using PORA operators whose operands are relation schemas. E_{PORA} is the set of all PORA expressions.

The *answer* $e(d)$ to an expression $e \in E_{PORA}$ with respect to a database d over $\mathbf{R}$ is obtained by substituting the relation r_i for every occurrence of R_i in e, for each i, and computing the result by invoking the operators present in e. The answer is undefined if some operand R of an expression is not in $\mathbf{R}$. Two sets of expressions E_1 and E_2 are *equivalent*, $E_1 \equiv E_2$, if for every $e_1 \in E_1$ there is some $e_2 \in E_2$ such that $e_1(d) = e_2(d)$ for all $d \in DB(\mathbf{R})$, and vice versa.

We note that $\sigma_{\not\sqsubseteq}$, $\sigma_=$ and $\sigma_{\neq}$ are not primitive since, for any relation r, $\sigma_{A \not\sqsubseteq B}(r) \equiv r - \sigma_{A \sqsubseteq B}(r)$, $\sigma_{A=B}(r) \equiv \sigma_{A \sqsubseteq B}(\sigma_{B \sqsubseteq A}(r))$ and $\sigma_{A \neq B}(r) \equiv r - \sigma_{A=B}(r)$. For an unordered domain $\sigma_{\sqsubseteq}$ becomes $\sigma_=$, so the PORA reduces to the standard (unordered) relational algebra [8], the *UORA*. For a given linear ordering $\leq$ of D, the *LORA* is the corresponding *linearly ordered* relational algebra.

In this paper we are only considering expressions that do *not* involve constants; so relations generated by E_{PORA} can only involve relations in d and contain values in $adom(d)$. We can, however, include constants in PORA expressions without affecting our results if we require that all our automorphisms map such constants to themselves (cf. Section 2.3 in [1]) and we also include these constants in $adom(d)$.

In what follows, our approach will be to view an ordered database as an unordered database together with a binary relation representing $\langle adom(d), \sqsubseteq \rangle$. We define ω to be the relation over a schema Ω given by $\omega = \{\langle a, b \rangle \mid a \sqsubseteq b \text{ and } a, b \in adom(d)\}$. An ordered relation r derived from d is regarded as an unordered relation $r \times \omega$. We assume from now on that $adom(r) \subseteq adom(d)$, i.e., $r \in Poss(d)$.) For any relation r, we define $Aut(\sqsubseteq, r)$ to be $Aut(\sqsubseteq, \{r, adom(d)\})$, where $adom(d)$ is regarded as a unary relation, and d is implied from the context.

We need the following technical lemmas to establish our main theorem.

Lemma 3.1 Let d be a database over $\mathbf{R}$ and let $d' = d \cup \{\omega\}$, considered as an unordered database over $\{R_1, \ldots, R_n, \Omega\}$. Then $Aut(=, d') = Aut(\sqsubseteq, d)$. $\square$

Lemma 3.2 Let r be a relation over R and let $r' = r \times \omega$, considered as an unordered relation over $R\Omega$. Then $Aut(=, r') = Aut(\sqsubseteq, r)$. $\square$

Defining d' and r' as above, the following result may be proved using induction on the number of relational operators and some algebraic manipulation.

Lemma 3.3 Let d be a database over $\mathbf{R}$ and r a relation over R. Then $e'(d') = r'$ for some $e' \in E_{UORA}$ if and only if $e(d) = r$ for some $e \in E_{PORA}$. $\square$

We note that r' can be replaced by r in Lemma 3.3, since r is a projection of r'.

We can now state Paredaens and Bancilhon's Theorem [2, 8] as:

Lemma 3.4 Let d be a database over $\mathbf{R}$ and r a relation over R. Then $e(d) = r$ for some $e \in E_{UORA}$ if and only if $Aut(=, d) \subseteq Aut(=, r)$. $\square$

We now extend this result to ordered databases:

Theorem 3.5 Let d be a database over $\mathbf{R}$ and r a relation over R. Then $e(d) = r$ for some $e \in E_{PORA}$ if and only if $Aut(\sqsubseteq, d) \subseteq Aut(\sqsubseteq, r)$.

Proof.
Now $Aut(\sqsubseteq, d) = Aut(=, d')$ by Lemma 3.1 and $Aut(\sqsubseteq, r) = Aut(=, r')$ by Lemma 3.2. So $Aut(\sqsubseteq, d) \subseteq Aut(\sqsubseteq, r)$ if and only if $Aut(=, d') \subseteq Aut(=, r')$. By Lemma 3.4, $Aut(=, d') \subseteq Aut(=, r')$ if and only if $e'(d') = r'$ for some $e' \in E_{UORA}$. It thus follows, by Lemma 3.3, that $Aut(\sqsubseteq, d) \subseteq Aut(\sqsubseteq, r)$ if and only if $e(d) = r$ for some $e \in E_{PORA}$. $\square$

We note that Theorem 3.5 can be generalised from ordered domains to data domains having any specified binary relation (but in this case $\sigma_=$ may be primitive). The following corollary of Theorem 3.5, which will be used in Theorem 4.2, follows using Lemma 3.2.

Corollary 3.6 Let $d = \{r_1, \ldots, r_n\}$ be a database over $\mathbf{R} = \{R_1, \ldots, R_n\}$ and let $r = r_1 \times \cdots \times r_n \times \omega$, considered as an unordered relation over $R_1 \cdots R_n \Omega$. Then $Aut(=, r) = Aut(\sqsubseteq, d)$. $\square$

We close this section with an interesting consequence of Theorem 3.5. Informally, in the case of linearly ordered domains, the LORA expresses exactly the countably infinite set of all possible relations generated by the active domain of a given database. This follows immediately from the fact that, for any linear ordering $\leq$, $Aut(\leq, d) = \{id\}$.

Corollary 3.7 Let d be a linearly ordered database. Then, for all $r \in Poss(d)$, $e(d) = r$ for some $e \in E_{LORA}$. $\square$

4 A Hierarchy of Computable Queries

We now investigate the relationship between computable queries, ordered domains and partially ordered relational algebras. We define hierarchies for each of these and show that there exists a one-to-one correspondence between them.

We will use subscripts to denote different orderings on D, i.e., $\mathcal{D}_i = \langle D, \sqsubseteq_i \rangle$. Similarly, we will write $PORA_i$ and ω_i. We first define the semantics of "more ordered" domains in terms of ordering automorphisms.

Definition 4.1 Domain $\mathcal{D}_2$ is said to be *more ordered* than domain $\mathcal{D}_1$, denoted by $\mathcal{D}_1 \preceq \mathcal{D}_2$, if $Aut(S, \sqsubseteq_2) \subseteq Aut(S, \sqsubseteq_1)$ for all finite $S \subseteq D$.

Note that, for $\mathcal{D}_1 \preceq \mathcal{D}_2$, it is neither necessary nor sufficient that $a \sqsubseteq_1 b$ implies $a \sqsubseteq_2 b$ (or vice versa). We observe that this definition would be equivalent even if we did not require S to be finite. This is a consequence of the fact that we are only considering *finite* ordering automorphisms; it would *not* be true if we allowed non-finite ordering automorphisms. The reason for including *all* finite $S \subseteq D$ in the above definition is that the active domain of a database can be any finite subset of D.

Example 2 In Figure 1, we use Hasse diagrams to represent ordered domains. It is easy to see that, for all $S \subseteq D = \{a, b, c\}$, $Aut(S, \sqsubseteq_3) \subseteq Aut(S, \sqsubseteq_2) \subseteq Aut(S, \sqsubseteq_1)$, and thus $\mathcal{D}_1 \preceq \mathcal{D}_2 \preceq \mathcal{D}_3$. (For an example with D infinite, simply add an infinite number of incomparable elements all greater than a, b and c.)

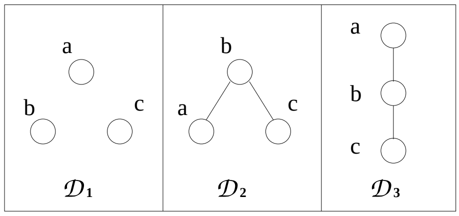

Figure 1: Hasse diagrams of ordered domains

Next we consider the expressiveness of the relational algebra for different orderings. The set of relations generated from the information contained in a database d, denoted by $Gen(\sqsubseteq_i, d)$, is defined by

$$Gen(\sqsubseteq_i, d) = \{r \mid r = e(d) \text{ for some } e \in E_{PORA_i}\}.$$

Definition 4.2 The relational algebra $PORA_2$ is *more powerful* than the algebra $PORA_1$, denoted by $PORA_1 \preceq PORA_2$, if $Gen(\sqsubseteq_1, d) \subseteq Gen(\sqsubseteq_2, d)$ for all databases d.

We now generalise the notion of computable query to ordered databases by appropriately modifying the approach in [3]. Our motivation is to include just those queries which are meaningful with respect to the ordered domain concerned, viz. those which are invariant under all order-preserving database automorphisms.

Let $\Delta = \bigcup_{i=0}^{\infty} \mathcal{P}(D^i)$ – recall that D is assumed to be a common domain.

Definition 4.3 A *meaningful computable query* with respect to a given domain $\mathcal{D}_i$ is a partial recursive function δ from $DB(\mathbf{R})$ to Δ, for some database schema $\mathbf{R}$, such that, for all $d \in DB(\mathbf{R})$,

(a) if $\delta(d)$ is defined, then $\delta(d) \in Poss(d)$, and

(b) for all $h \in Aut(\sqsubseteq_i, d)$, $h(\delta(d)) = \delta(d)$.

We denote the set of all meaningful computable queries by Q_i. Note that, for unordered domains, our definition of a meaningful computable query is just the conventional one [3].

The following lemma allows us to relate ordering automorphisms of domains to order-preserving automorphisms of databases.

Lemma 4.1 $\mathcal{D}_1 \preceq \mathcal{D}_2$ if and only if, for every database schema $\mathbf{R}$, $Aut(\sqsubseteq_2, d) \subseteq Aut(\sqsubseteq_1, d)$ for all $d \in DB(\mathbf{R})$.

Proof.
(If) Consider any finite $S \subseteq D$, say $S = \{a_1, \ldots, a_k\}$, and let $h \in Aut(S, \sqsubseteq_2)$. Choose $d \in DB(\mathbf{R})$ as follows: for each $r \in d$, r consists of exactly k tuples $\{t_1, \ldots, t_k\}$, where $t_i = \langle a_i, \ldots, a_i \rangle$ for $1 \leq i \leq k$. Obviously, $h \in Aut(\sqsubseteq_2, d)$ so, by hypothesis, $h \in Aut(\sqsubseteq_1, d)$, and thus $h \in Aut(S, \sqsubseteq_1)$.
(Only if) This follows immediately, since $Aut(\sqsubseteq_i, d) = Aut(adom(d), \sqsubseteq_i) \cap Aut(=, d)$ for any database d. □

Our main result establishes the correspondence between the hierarchies for partially ordered domains, computable queries and query languages.

Theorem 4.2

(a) $\mathcal{D}_1 \preceq \mathcal{D}_2$ if and only if $Q_1 \subseteq Q_2$,

(b) $\mathcal{D}_1 \preceq \mathcal{D}_2$ if and only if $PORA_1 \preceq PORA_2$.

Proof.

(a) *(If)* Assume $\mathcal{D}_1 \not\preceq \mathcal{D}_2$. By Lemma 4.1, this implies that there exists a database $d' = \{r'_1, \ldots, r'_n\}$ such that $h_2 \notin Aut(\sqsubseteq_1, d')$ for some $h_2 \in Aut(\sqsubseteq_2, d')$. We will construct a query δ that is in Q_1 but not in Q_2. Given d', let $r' = r'_1 \times \cdots \times r'_n \times \omega_1$. Then, by Corollary 3.6, $h(r') = r'$ for all $h \in Aut(\sqsubseteq_1, d')$, but $h_2(r') \neq r'$ since $h_2 \notin Aut(\sqsubseteq_1, d')$. We now define $\delta(d)$ to be r' when $d = d'$ and the empty set otherwise. Thus, by Definition 4.3, $\delta \in Q_1$ but $\delta \notin Q_2$.

(Only if) Assume $\mathcal{D}_1 \preceq \mathcal{D}_2$. Let $\delta \in Q_1$ be a query from $DB(\mathbf{R})$ to Δ and let $d \in DB(\mathbf{R})$. From Definition 4.3, $\delta(d) \in Poss(d)$ and $h(\delta(d)) = \delta(d)$ for all $h \in Aut(\sqsubseteq_1, d)$. By Lemma 4.1, $Aut(\sqsubseteq_2, d) \subseteq Aut(\sqsubseteq_1, d)$, so $h(\delta(d)) = \delta(d)$ for all $h \in Aut(\sqsubseteq_2, d)$. Therefore $\delta \in Q_2$.

(b) *(If)* Assume $PORA_1 \preceq PORA_2$. Let $d = \{r_1, \ldots, r_n\}$ be any database and let $r = r_1 \times \cdots \times r_n \times \omega_1$. Clearly, ω_1 can be derived from d by some $e \in PORA_1$ and thus $r \in Gen(\sqsubseteq_1, d)$. By the above assumption, this implies $r \in Gen(\sqsubseteq_2, d)$. Then, by Theorem 3.5,

$$Aut(\sqsubseteq_2, d) \subseteq Aut(\sqsubseteq_2, r) \subseteq Aut(=, r).$$

By Corollary 3.6, it follows that $Aut(\sqsubseteq_2, d) \subseteq Aut(\sqsubseteq_1, d)$. So $\mathcal{D}_1 \preceq \mathcal{D}_2$ by Lemma 4.1.

(Only if) Assume $\mathcal{D}_1 \preceq \mathcal{D}_2$. Let d be any database and let $r \in Gen(\sqsubseteq_1, d)$. It suffices to show that $r \in Gen(\sqsubseteq_2, d)$. Now

$$Aut(\sqsubseteq_i, d) = Aut(adom(d), \sqsubseteq_i) \cap Aut(=, d),$$

so it follows that

$$Aut(adom(d), \sqsubseteq_2) \cap Aut(\sqsubseteq_1, d) = Aut(\sqsubseteq_2, d),$$

since $Aut(adom(d), \sqsubseteq_2) \subseteq Aut(adom(d), \sqsubseteq_1)$ by Definition 4.1. Similarly,

$$Aut(adom(d), \sqsubseteq_2) \cap Aut(\sqsubseteq_1, r) = Aut(\sqsubseteq_2, r).$$

By Theorem 3.5, $Aut(\sqsubseteq_1, d) \subseteq Aut(\sqsubseteq_1, r)$, therefore $Aut(\sqsubseteq_2, d) \subseteq Aut(\sqsubseteq_2, r)$. Hence, by Theorem 3.5 again, we have $r \in Gen(\sqsubseteq_2, d)$. □

Corollary 4.3 $Q_1 \subseteq Q_2$ if and only if $PORA_1 \preceq PORA_2$. □

We note that, alternatively, we could have proved this corollary directly from Theorem 3.5.

5 Conclusions

We have established a correspondence between the hierarchies of (1) meaningful computable queries, (2) partially ordered domains, and (3) partially ordered relational algebras. The implication of this result is that when the underlying data domain of a database has more inherent structure, then the scope of possible queries is wider. In other words, the ordered relational model can provide more expressive query languages than those of the conventional one, and in this sense we can say that more meaningful queries are possible with respect to an ordered relational database. Furthermore, it is clear that our results extend in the obvious way to the situation where several partial ordering relations are defined on the data domain D.

For any database d, it follows from the results in Section 3 that the PORA applied to d has no more expressive power than the UORA applied to $d \cup \{\omega\}$, where ω is the partial order relation on $adom(d)$. However, in most cases it is much more economical and efficient to compare domain elements computationally rather than storing and using the relation ω (e.g., for a numerical or lexicographical ordering). Even for a partial order with no apparent computational structure (e.g., an employee hierarchy), it is more economical to store the Hasse diagram rather than its transitive closure.

There remains the problem of trying to find an efficient algorithm to determine whether one domain is more ordered than another. We conjecture that this problem is hard since, even in the case of finite domains, the number of automorphisms may be exponential.

References

[1] P. Atzeni and V. De Antonellis. *Relational Database Theory.* Benjamin/Cummings Publishing Company, Inc., (1993).

[2] F. Bancilhon. On the Completeness of Query Languages for Relational Databases. In *LNCS 64: Mathematical Foundations of Computer Science*, Springer-Verlag, pp. 112-124, (1978).

[3] A.K. Chandra and D. Harel. Computable Queries for Relational Data Bases. *Journal of Computer and System Sciences* **21**, pp. 156-178, (1980).

[4] S. Ginsburg and R. Hull. Sort Sets in the Relational Model. *Journal of the Association for Computing Machinery* **33**(3), pp. 465-488, (1986).

[5] P. Halmos. *Naive Set Theory,* Springer-Verlag, New York, (1974).

[6] M. Levene and G. Loizou, *A Guided Tour of Relational Databases and Beyond*, London, Springer-Verlag, (1999).

[7] D. Maier and B. Vance. A Call to Order. In *Proceedings of the Twelfth ACM Symposium on Principles of Databases Systems.* pp. 1-16, (1993).

[8] J. Paredaens. On the Expressive Power of the Relational Algebra. *Information Processing Letter* **7**(2), pp. 107-111, (1978).