Pergamon

Information Systems Vol. 24, No. 7, pp. 535–554, 1999

Printed in Great Britain
0306-4379/99 $20.00 + 0.00

ORDERED FUNCTIONAL DEPENDENCIES IN RELATIONAL DATABASES†

WILFRED NG

Department of Computing, Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong

(Received 17 November 1998; in final revised form 19 August 1999)

Abstract — We extend the relational data model to incorporate linear orderings into data domains, which we call the ordered relational model. The conventional Functional Dependencies (FDs) are examined in the context of ordered relational databases by using the notion of System Ordering Independence (SOI), which refers to the desirable scenario that the ordering of tuples in a relation is independent of the implementation of the underlying DBMS. We also extend Armstrong's axiom system for FDs to object relations, which are a subclass of ordered relations that allow us to view tuples as objects. We formally define Ordered Functional Dependencies (OFDs) for the extended model by means of two possible extensions of domains, pointwise-orderings and lexicographical orderings. We first present a sound and complete axiom system for OFDs in the case of pointwise-orderings and then establish a sound and complete set of chase rules for OFDs in the case of lexicographical orderings. Our main result shows that the implication problems for both cases of OFDs are decidable, and that it is linear time for the case of pointwise-orderings. ©1999 Elsevier Science Ltd. All rights reserved

Key words: Functional Dependencies, Relational Databases, Linear Ordering, Linearly Ordered Domains, Chase Procedure

1. INTRODUCTION

Functional dependencies (FDs) are commonly recognised as the most fundamental integrity constraint arising in practice in conventional relational databases [38, 3]. The implication problem for functional dependencies is also well-known [21, 17]. We assume here that the data domains of the relational data model are linearly ordered and call the extended model the ordered relational model. Our assumption is well-justified, since linear ordering is a fundamental property of almost all primitive data types. Existing database theory usually makes an implicit assumption that domains are linearly ordered, allowing the linear ordering predicate, $\leq$, to be used in selection formulae [38, 3]. In fact, all relational database systems also support the following three kinds of domain orderings considered to be essential in practical use: (1) the *alphabetical ordering* over the domain of strings, (2) the *numerical ordering* over the domain of numbers, and (3) the *chronological ordering* over the domain of dates [10, 11]. Let us call these linear orderings the *standard domain orderings*.

There is strong evidence that ordering is inherent to the underlying structure of data in many database applications [7, 26, 25, 34, 32], in which linear ordering is particularly important to those advanced applications involving temporal or scientific information [26, 28]. We restrict the scope of our investigation to the case of linear orderings. In addition, we call the ordering semantics in the context of a specific application *semantic orderings*, which are a central notion that we have used in developing ordered SQL [31, 32]. Semantic orderings are also used here in defining the ordered relational model in the following way, given a data domain of the extended model, apart from the standard domain orderings at the logical level such as numerical and alphabetical orderings provided by DBMSs, we can also declare new semantic orderings at the external level above the logical level, overriding the standard domain orderings.

We formalise the notion of FDs being satisfied in an ordered database and call them *Ordered Functional Dependencies* (OFDs). Informally speaking, OFDs can capture a monotonicity property between two sets of values projected onto some attributes in a relation. The semantics of OFDs are defined by means of two possible extensions of the domain orderings: *pointwise-orderings* and *lexicographical orderings*. Pointwise-orderings require each component of a data value to be greater

†Recommended by Maurizio Lenzerini

EMP	POST_TITLE	YEARS	SALARY
Mark	Senior Programmer	15	35K
Nadav	Junior Programmer	7	25K
Ethan	Junior Programmer	6	22K

Fig. 1: An Employee Relation EMP_RECORD

than its predecessors and lexicographical orderings resemble the way in which words are arranged in a dictionary. For example, the tuple $\langle x_1, \ldots, x_n \rangle$ is less than another tuple $\langle y_1, \ldots, y_n \rangle$ according to a pointwise-ordering, if, for all $1 \leq i \leq n$, $x_i \leq y_i$. The tuple $\langle x_1, \ldots, x_n \rangle$ is less than another tuple $\langle y_1, \ldots, y_n \rangle$ according to a lexicographical ordering, if there is an index $j \geq 1$ such that $x_j < y_j$ and, for each $i < j$, $x_i = y_i$. We classify OFDs according to whether we use pointwise-orderings or lexicographical orderings in their definitions, whose short forms are written as POFDs ($X \hookrightarrow Y$) and LOFDs ($X \rightsquigarrow Y$).

As a motivating example, consider in Figure 1 a relation called EMP_RECORD over the set of attributes {EMP, POST_TITLE, YEARS, SALARY}. The semantics of EMP_RECORD is as follows, an EMPloyee with a given POST TITLE, who has been working in a company for some YEARS, has the present SALARY.

We assume that there is a semantic ordering in POST_TITLE as represented by the following domain {'Junior Programmer' < 'Senior Programmer'}. The relation EMP_RECORD in Figure 1 then satisfies the POFD, {POST_TITLE, YEARS} $\hookrightarrow$ SALARY, which states the fact that the SALARY of an employee is greater than other employees who have junior post titles and less experience in the company, and the LOFD, {POST_TITLE, YEARS} $\rightsquigarrow$ SALARY, which states the fact the SALARY of an employee is greater than other employees who have junior post titles, or the same post title but less experience in the company. Note that the semantics of the POFD and the LOFD mentioned above are different. For instance, in the first case, an employee has a higher salary only if he or she has both a senior post title and more experience than another, whereas in the second, it requires only that he or she only has to have a more senior post title. If Mark leaves his post, Ethan replaces him and his record is updated to $\langle Ethan, SeniorProgrammer, 6, 26K \rangle$ (i.e., updating the third tuple), then this updating violates neither the POFD nor the LOFD. However, if his record is updated to $\langle Ethan, SeniorProgrammer, 6, 24K \rangle$, then it violates the LOFD, since Ethan now has a more senior title but a lower salary than Nadav. But the POFD still holds in this updating, since Nadav still has more experience than Ethan. The appropriateness of the choice between the POFD or the LOFD in this case depends entirely on the semantics of the promotion policy adopted by the company.

In the relational database literature, the *implication problem* is an important issue arising from investigating data dependencies, which we now state as follows: given a relation r which satisfies a set of data dependencies F, is it also true that r satisfies a data dependency f? If the answer to the above question is positive, then we say F *logically implies* f and denote this fact by F $\models f$. There are two approaches to this problem.

One approach is to establish a set of *inference rules* which constitutes the *axiom system* $\mathcal{A}$. We can use the rules of $\mathcal{A}$ to *derive* f from F and denote this process by F $\vdash f$. We call $\mathcal{A}$ *sound and complete*, if we can prove that F $\vdash f$ if and only if F $\models f$. A sound and complete axiom system for F is desirable, since it guarantees that the implication problem for F is *recursively enumerable* [13], which also implies in principle that we can exhaustively apply the rules of $\mathcal{A}$ to generate all data dependencies logically implied by F. The axiom system $\mathcal{A}$ also provides us with a basis for finding a more efficient algorithm to solve the implication problem. We adopt this approach to show that the axiom system comprising the inference rules for POFDs, a superset of Armstrong's axiom system for FDs [2], is sound and complete.

Another approach is to develop a *chase* procedure, which consists of a set of *chase rules* as a theorem-proving tool. We choose an appropriate chase rule to apply to a relation r until a fixpoint is attained in order to test whether r satisfies F [27, 3]. The chase procedure operates on a relation containing variables as data values, known as a *tableau* [27, 3], which is basically the template for those relations that could possibly violate f. Suppose we can prove that using the chase procedure we can transform a tableau that satisfies F into a tableau that also satisfies f, and this holds if and only if F $\models f$. Then we are able to use the chase procedure to confirm or refute that F logically implies f. We call the chase procedure possessing this property *sound and complete*. We adopt this approach to extend the chase rules for LOFDs. We investigate the properties of a relation r being chased with respect to a set of LOFDs F (which we denote as $CHASE(r,\mathrm{F})$) and then show that the procedure $CHASE(r,\mathrm{F})$ terminates and satisfies F. Using an extended notion of *tableaux* for LOFDs, we show that the chase is sound and complete for LOFDs.

Our investigation also relates to the issues of data dependencies in those extensions of the relational data model to incorporate lists or sequences as data types [19, 15, 37]. A list can arrange objects in some pre-defined linear order. It can therefore be defined as a mapping between a collection of similarly structured real world objects and a linearly ordered domain. From this point of view, a linearly ordered set can be regarded as a non-repeating list. However, a linearly ordered set is not allowed to contain duplicates, which is different from a list in general. Ginsburg and Hull [14] have introduced the term *order dependencies* and examined the issue of the extension of functional dependencies to incorporate information involving partial order. They exhibit a sound and complete set of inference rules for order dependencies, whose implication problem is shown to be co-NP complete [13]. The central notion of order dependencies is similar to that of our definition of ordered functional dependencies arising from pointwise-orderings (POFDs), except that the involved domain orderings in order dependencies are classified into total order, empty order and general partial-order. This finer classification of a partial ordering requires relatively complex mathematical tools to explore the axiom systems for order dependencies, whereas we intend to further clarify the issues arising from the semantics of FDs, POFDs and LOFDs in the context of ordered databases.

The rest of the paper is organised as follows. In Section 2 we clarify the notion of linear order and its extensions to Cartesian product of linearly ordered sets. We formally define the ordered relational model. In Section 3 we examine FDs in the context of ordered databases. In Section 4 we present the axiom system comprising the inference rules for POFDs, which is sound and complete. Also, we define the chase rules for LOFDs and, using an extended notion of tableaux for LOFDs, show that the chase is sound and complete for LOFDs. In Section 5 we give our concluding remarks.

Throughout this paper we make use of the following notation.

Definition 1 Let X and Y be sets, then $X \subseteq Y$ denotes *set inclusion* and $X \subset Y$ denotes *proper set inclusion*. We denote the k term Cartesian product $X \times X \cdots \times X$ by X^k, and the singleton $\{A\}$ simply by A when no ambiguity arises. We refer to a sequence of attributes as a short hand for a sequence of distinct attributes. (In other words, we assume that sequences of attributes do not contain any repeated attributes.) We use the common notation for both sequences and sets, i.e., X and Y are used to denote sequences of attributes, whereas A and B are used to denote single attributes. When no ambiguity arises we refer to a sequence of attributes as a set of attributes. However, we remark that the sequences AB and BA are different, whereas the sets AB and BA are the same. We take $A \in \langle A_1, \ldots, A_n \rangle$ to mean $A \in \{A_1, \ldots, A_n\}$ and $\langle A_1, \ldots, A_n \rangle \subseteq \langle B_1, \ldots, B_m \rangle$ where $n \leq m$, to mean $\{A_1, \ldots, A_n\} \subseteq \{B_1, \ldots, B_m\}$. We may also write $A_1 \cdots A_n$ instead of $\langle A_1, \ldots, A_n \rangle$. Let $X = \langle A_1, \ldots, A_m \rangle$ and $Y = \langle B_1, \ldots, B_n \rangle$. We denote the fact that two sequences have the same elements, i.e., $\{A_1, \ldots, A_m\} = \{B_1, \ldots, B_n\}$, by $X \sim Y$. The difference between two sequences of attributes, denoted as $X - Y$, is defined by the sequence resulting from removing all the common attributes in X and Y from X while maintaining the original order of the remaining attributes in X. We also denote by XY the *concatenation* of two sequences X and Y, where X and Y are *disjoint*, i.e., they have no common attributes. If X and Y are not disjoint, then XY is defined as $X(Y - X)$.

2. THE ORDERED RELATIONAL MODEL

We assume the usual definition of a *linear ordering* $\leq$ on the set S: a binary relation on S satisfying the conditions of *reflexivity*, *anti-symmetry*, *transitivity* and *linearity*, which becomes a *partial ordering* if without the linearity condition [18].

Definition 2 A *linear ordering of the set* S is a binary relation on S, denoted by $\leq$, satisfying the following conditions.
For all $x, y, z \in S$,

1. *Reflexivity*: $x \leq x$.
2. *Anti-symmetry*: if $x \leq y$ and $y \leq x$, then $x = y$.
3. *Transitivity*: if $x \leq y$ and $y \leq z$, then $x \leq z$.
4. *Linearity*: $x \leq y$ or $y \leq x$.

A *linearly ordered set*, denoted as $\mathcal{S}$, is a structure $\langle S, \leq \rangle$. It consists of a set S which is linearly ordered by the relation $\leq$. From now on, the term *ordered* will mean linearly ordered, unless explicitly stated otherwise. We assume that the equality predicate, $=$, still applies to ordered sets and use the notation $<$ to represent the usual meaning of $\leq$ but $\neq$. We now define the extension of the orderings of data domains on the Cartesian product of ordered sets so as to capture the semantics of data.

Let $D_1, \ldots, D_n$ be n ordered sets, t be an element in the Cartesian product $S = D_1 \times \cdots \times D_n$ and $t[i]$ be the ith coordinate of t. We now define *pointwise-orderings* on the Cartesian product of ordered sets.

Definition 3 Let $t_1, t_2 \in S$. A *pointwise-ordering* on S, denoted by $\leq_S^p$, is defined as follows: $t_1 \leq_S^p t_2$, if, for all $1 \leq i \leq n$, $t_1[i] \leq_{D_i} t_2[i]$.

It is easy to check from Definition 3 that $\leq_S^p$ is a partial ordering on S. The idea of the pointwise-ordering extension on data domains has been commonly used to study the issues concerning incomplete information [40, 25, 24], but in this case we need to define partial ordering over data domains, in order to capture the semantics that a known data value is equally informative to itself but more informative than a null value. This approach can be further explained by the following example: assume that a domain of constants, denoted as Dom, contains a distinguished symbol UNK, which means that the data value exists but is UNKnown. A partial ordering on Dom is defined by, for all $x, y \in Dom$, $x \leq y$ if $x = y$ or $x =$ UNK. Then we can extend $\leq$ to be a pointwise-ordering in a relation r over $\{A, B\}$ as follows, for all $t_1, t_2 \in r$, $t_1 \leq^p t_2$ if $t_1[A] \leq t_2[A]$ and $t_1[B] \leq t_2[B]$. This extension naturally captures the meaning of t_1 being *less informative* than t_2, or alternatively t_2 being *more informative* than t_1.

We next define another kind of ordering, *lexicographical ordering*, on the Cartesian product of ordered sets.

Definition 4 Let $t_1, t_2 \in S$. A *lexicographical ordering* on S, denoted by $\leq_S^l$, is defined as follows: $t_1 \leq_S^l t_2$, if either

1. there exists k with $1 \leq k \leq n$ such that $t_1[k] <_{D_k} t_2[k]$, and for all $1 \leq i < k$, $t_1[i] = t_2[i]$, or
2. for all $1 \leq i \leq n$, $t_1[i] = t_2[i]$.

In contrast to pointwise-ordering, it follows from Definition 4 that $\leq_S^l$ is a linear ordering on S. Lexicographical ordering is a common and fundamental property of many data structures. For example, let N be the set of natural numbers, then we can construct the lexicographical ordering on

N^n, which is an infinite lexicographical ordering. Another important example is the lexicographical ordering on alphabets. Let A be an ordered set over a finite alphabet. Then we can easily construct a finite lexicographical ordering on A^n in the same way as N^n, which we call a *dictionary ordering* or an *alphabetical ordering*, since it resembles the ordering of words in a dictionary. We observe that the ordering of the domain $DATE$, called *chronological orderings*, follows the lexicographical ordering of the domains $YEAR$, $MONTH$ and DAY, if (1) the domain $MONTH$ has the ordering as $\{JAN \leq FEB \leq \cdots \leq DEC\}$ and (2) the Cartesian product of the domains are taken in the following order: $YEAR \times MONTH \times DAY$. However, we remark that the domain $DATE$ is not equivalent to $YEAR \times MONTH \times DAY$ due to the fact that months have different number of days [26].

Let D be a countably infinite set of constant values and $\leq_D$ be an ordering on D. Without loss of generality, we assume that all attributes share the same domain D. We now give the definition of an ordered database.

Definition 5 We assume a countably infinite ordered set of attribute names, $\langle U, \leq_U \rangle$. For all attributes $A \in U$, the domain of A is $\langle D, \leq_D \rangle$. We call $\leq_D$ the *domain ordering* of D.

Definition 6 A relation schema (or simply a schema) R, is a subset of U consisting of a finite set of attributes $\{A_1, \ldots, A_m\}$ for some $m \geq 0$. A *database schema* is a finite set $\mathbf{R} = \{R_1, \ldots, R_n\}$ of relation schemas, for some $n \geq 1$.

Definition 7 Let $X = \{A_1, \ldots, A_m\}$ be a finite subset of U where $A_i \neq A_j$ for $i \neq j$ and $A_1 \leq_U \cdots \leq_U A_m$. A *tuple* t over X is a member of D^m. We let $t[A_i]$ denote the ith coordinate of t. The *projection* of a tuple t onto a set of attributes $Y = \{A_{i_1}, \ldots, A_{i_k}\}$, where $1 \leq i_1 < \cdots < i_k \leq m$, is the tuple $t[Y] = \langle t[A_{i_1}], \ldots, t[A_{i_k}] \rangle$.

Definition 8 An *ordered relation* (or simply a relation) r defined over a schema R is a finite set of tuples over R. An *ordered database* (or simply a database) over $\mathbf{R} = \{R_1, \ldots, R_n\}$ is a finite set $d = \{r_1, \ldots, r_n\}$ such that each r_i is a relation over R_i.

We make two assumptions in our model. First, the orderings of domains can be extended to tuples so that tuples in an ordered relation are ordered according to the lexicographical ordering of the domains associated with the attributes present in the underlying relation schema. Any change in the order of attributes in a relation schema may affect the order of tuples in an ordered relation. Second, given a data domain, apart from the system ordering assumption, we can declare one or more semantic orderings which override the default system ordering.

In order to further discuss the relationship between various notions of orderings in a DBMS, we let a *system ordering*, denoted by $\leq_{sys}$, on a relation r be a linear ordering on r that is generated by a DBMS. Note that the concepts of system orderings and domain orderings are different. The ordering $\leq_{sys}$ may or may not follow the extension of domain orderings on tuples, as different DBMSs have their own storage and retrieval strategy. The following example helps clarify this concept further.

Example 1 Let r over A be the relation $\{a, b, c\}$ (having 3 tuples) and the domain of A be alphabetically ordered. A system ordering, which is dependent on a particular DBMS, can choose one of the six ways as shown in Figure 2 to arrange the tuples in r.

A
a
b
c

A
a
c
b

A
b
a
c

A
b
c
a

A
c
a
b

A
c
b
a

Fig. 2: Six Possible System Orderings of Tuples in r □

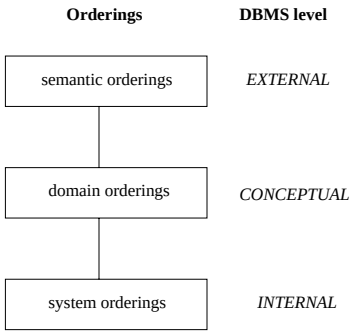

Fig. 3: Orderings at Different DBMS Levels

Although in most cases the choice of the ordering of r in the above example is done according to standard domain orderings (i.e., the first one in Figure 2), the ordering of tuples cannot be guaranteed as alphabetically ordered if r is the answer to a complex query over the DBMS. This is because the choice of ordering of r is dependent on the implementation of a particular DBMS. It is worthwhile to consider how $\leq_{sys}$ affects the use of *cursors* in an embedded SQL statement [11]. For example, the result of selecting the nth tuple of r is dependent on the ordering of tuples in r. In such a case there will be a risk of losing *physical data independence*, because the returned tuples depend on $\leq_{sys}$, which in turn depends on the implementation of the system. This is rather undesirable and thus the current remedy is to use the *ORDER BY* clause to help "position" tuples when declaring a cursor (c.f., see chapter 10 in [11]). In other words, we need domain orderings to achieve physical data independence. We show in Figure 3 the differences between the various notions of orderings introduced so far.

For the sake of simplicity in notation, we use $\leq_X^p$ (or $\leq_X^l$) to mean the pointwise-ordering (or the lexicographical ordering) on the Cartesian products of data domains associated with a sequence of attributes X in our further discussion. We now define an operator called a *domain ordering operator* whose aim is to help present the relationship between domain orderings and data dependencies.

Definition 9 Let r be a relation over R and $\mathcal{L}$ be the set of all linear orderings on r. A *domain ordering operator* over r, denoted by ω_X, where $X \subseteq R$ is a sequence of attributes, is defined by $\omega_X(r) = \{\leq_r \in \mathcal{L} \mid \forall t_1, t_2 \in r$, if $t_1 \leq_r t_2$, then $t_1[X] \leq_X^l t_2[X]\}$.

We now define an important subclass of the results obtained by ω_X over a given relation r in order to investigate the independence of system orderings for r.

Definition 10 Given a relation r over R, we call ω_X *system ordering independent* with respect to r (or simply SOI when r is clear from the context) if $\omega_X(r)$ is a singleton.

Informally, the SOI property of a domain operator ensures that the ordering of the tuples in a relation can be uniquely determined by the domain ordering associated with a given sequence of attributes, and thus the relation avoids the interference arising from the low level system ordering. The following example help clarify this concept further.

Example 2 Let $D_1 = \{1, 2\}$ and $D_2 = \{a, b, c\}$ and a given relation $r = \{\langle 2, a\rangle, \langle 1, c\rangle, \langle 1, b\rangle\}$. Then the ordered relations given in Figure 4 exhibit two different domain orderings in $\omega_{A_1}(r)$, because there are two choices of ordering for the tuples $\langle 1, b\rangle$ and $\langle 1, c\rangle$ by the system.

A_1	A_2
1	b
1	c
2	a

A_1	A_2
1	c
1	b
2	a

Fig. 4: Two Possible Domain Orderings of Tuples on r □

So we can see from the above example that the ordering of r is still partially system dependent when it is ordered according to the domain ordering of A_1 only. It is also clear that if X is equal to the schema of r, then ω_X is SOI. If X is a proper subset of the schema of r, then it is desirable for ω_X to be SOI, since we can save some computation resources of the system to achieve the independence of system orderings. This is because the system does not have to perform the sorting over every attribute in the relation schema in order to maintain ordered relations (recall the assumption that in an ordered relation tuples are ordered). Let the projection of a relation r over R onto Y, denoted as $\pi_Y(r)$, be defined by $\pi_Y(r) = \{t[Y] \mid t \in r\}$. We now introduce the following interesting properties, which will be useful in establishing the axiom system for FDs holding in object relations next section.

Proposition 1 *Let $X, Y, Z \subseteq R$ and r be a relation over R. The following statements are true.*

1. *$\omega_R(r)$ is SOI.*
2. *If $\omega_X \pi_{XYZ}(r)$ is SOI, then $\omega_X \pi_{XY}(r)$ is SOI.*
3. *If $\omega_X \pi_{XY}(r)$ is SOI, then $\omega_{XZ} \pi_{XYZ}(r)$ is SOI.*
4. *If $\omega_X \pi_{XY}(r)$ is SOI, then $\omega_{XZ} \pi_{XY}(r)$ is SOI.*
5. *If $\omega_X \pi_{XY}(r)$ is SOI and $\omega_Y \pi_{YZ}(r)$ is SOI, then $\omega_X \pi_{XZ}(r)$ is SOI.*

Proof. The first statement is obviously true, since $\leq_R^l$ defines a unique linear ordering on r. The second statement can be proved by assuming to the contrary that it is possible to have two distinct linear orderings, say $\leq_1$ and $\leq_2$, in $\omega_X \pi_{XY}(r)$. It thus follows that there should be two distinct tuples t_1 and t_2 in $\pi_{XY}(r)$ such that $t_1 <_1 t_2$ but $t_1 <_2 t_2$. It also follows from Definition 9 that $t_1[X] = t_2[X]$. Let t_1' and t_2' in $\pi_{XYZ}(r)$ such that $t_1'[XY] = t_1$ and $t_2'[XY] = t_2$. So, $t_1' \neq t_2'$ but $t_1'[X] = t_2'[X]$. Thus, we have two distinct linear orderings that can be chosen to arrange tuples over $\pi_{XYZ}(r)$, leading to a contradiction to the assumption of $\omega_X \pi_{XYZ}(r)$ being SOI. The remaining statements can be proved in a similar way. For instance, in order to prove the fifth statement we assume $\leq_1$ and $\leq_2$ being two distinct linear orderings in $\pi_{XZ}(r)$ and let t_1 and t_2 be two distinct tuples in $\pi_{XZ}(r)$ such that $t_1 <_1 t_2$ but $t_1 <_2 t_2$. It follows from Definition 9 that $t_1[X] = t_2[X]$. Thus, $t_1[Z] \neq t_2[Z]$ and $t_1[YZ] \neq t_2[YZ]$. Two cases are then needed to consider. First, if $t_1[Y] = t_2[Y]$, we have two distinct linear orderings that can be chosen to arrange tuples over $\pi_{YZ}(r)$, leading to a contradiction to the assumption of $\omega_Y \pi_{YZ}(r)$ being SOI. Second, if $t_1[Y] \neq t_2[Y]$, then $t_1[XY] \neq t_2[XY]$. Again, we have two distinct linear orderings that can be chosen to arrange tuples over $\pi_{XY}(r)$, leading to a contradiction to another assumption of $\omega_X \pi_{XY}(r)$ being SOI. □

3. FUNCTIONAL DEPENDENCIES IN ORDERED DATABASES

Bearing in mind that the implication problem is an important issue arising in developing the theory of data dependencies, and that FDs are the most natural data dependencies arising in practice, we first formalise the notions of *logical implication* and *an axiom system*, and then review *Armstrong's axiom system* for FDs, which is a classic example of *axiom systems* in the literature of relational database theory [38, 3].

Definition 11 A set of data dependencies F *logically implies* a data dependency f over R, written $F \models f$, whenever for all relations r over R, if, for all $f' \in F$, $r \models f'$ holds, then $r \models f$ also holds. An *axiom system* $\mathcal{A}$ for F is a set of inference rules (or simply rules) that can be used to *derive* data dependencies from F over R. We say that f is *derivable* from F by $\mathcal{A}$, if there is a finite sequence of data dependencies over R, whose last element is f, and where each data dependency in the said sequence is either in F or follows from a finite number of previous data dependencies in the sequence by one of the inference rules. We denote by $F \vdash f$ the fact that f is *derivable* from F by a specified axiom system.

Definition 11 will be repeatedly used in different contexts of data dependencies. For example, in this section the set of data dependencies F is restricted to the scope of FDs, but when discussing POFDs in Section 4.1, we will use $F \models f$ to mean that a set of POFDs F logically implies a POFD f. Similarly, we will also use $F \models f$ to mean that a set of LOFDs F logically implies an LOFD f, when discussing LOFDs in Section 4.2.

Armstrong's axiom system provides a set of inference rules which can infer new FDs from given ones. It is also well-known that Armstrong's axiom system is sound and complete for FDs being satisfied in conventional relations [2, 38]. This result is very significant in database design, since by using this axiom system we can derive some efficient algorithms to confirm whether or not a given FD holds in a relation schema [3]. It also provides us with a basis for developing FDs in the context of other advanced applications which have fuzzy, incomplete or imprecise information [33, 23, 24].

Definition 12 Let X, Y, Z be subsets of R, $A \in R$ and F be a set of FDs. *Armstrong's axiom system* constitutes the following inference rules for FDs.

(FD1) *Reflexivity*: if $Y \subseteq X$, then $F \vdash X \rightarrow Y$.

(FD2) *Augmentation*: if $F \vdash X \rightarrow Y$, then $F \vdash XA \rightarrow YA$.

(FD3) *Transitivity*: if $F \vdash X \rightarrow Y$ and $F \vdash Y \rightarrow Z$, then $F \vdash X \rightarrow Z$.

There are two possible views of FDs in the context of ordered databases. The first view is straightforward, that is, Armstrong's system can be directly carried over to ordered relations, since we assume that the equality predicate still applies to ordered domains. Another view of FDs in the context of ordering is more interesting. We now use ω_X and π_X in the following theorem to give a new interpretation of FDs via the notion of SOI.

Theorem 2 *An ordered relation r over R satisfies a functional dependency $X \rightarrow Y$ if $\omega_X \pi_{XY}(r)$ is SOI.*

Proof. Let $t_1, t_2 \in r$ such that $t_1[X] = t_2[X]$. Assume to the contrary that $t_1[Y] \neq t_2[Y]$ and thus it follows that $t_1[XY] \neq t_2[XY]$. So we have two distinct linear orderings, say $\leq_1$ and $\leq_2$, which can be chosen to arrange tuples over $\pi_{XY}(r)$, such that $t_1[XY] <_1 t_2[XY]$ but $t_2[XY] <_2 t_1[XY]$. This leads to a contradiction, since we violate the assumption of $\omega_X \pi_{XY}(r)$ being SOI. □

The operator ω_X can be further used to define a subclass of relations called *object relations* [6]. We need the following definition to illustrate this concept.

Definition 13 An attribute $M \in R$ is said to be a *meta-attribute* for an ordered relation r over R, if it satisfies $\omega_{MX}(r) = \leq_R^l$ for all $X \subseteq R$, where X can be empty.

We call a relation schema R an *object relational schema* if it contains a distinguished attribute being a meta attribute. We also call a subclass of relations *object relations*, if it consists of relations that are defined over object relational schemas. Meta-attributes in object relational schemas can be maintained by the system only, and can be hidden from users. The definition of meta-attributes formalises the use of *tuple identifiers* in a relation. For example, the relational DBMS Oracle employs an attribute called *ROWID* (ROW IDentifier) to manipulate tuples but this attribute is normally hidden from users [22]. The following proposition states that meta-attributes possess the desirable property of SOI.

M	X^+	Z
1	$1 \cdots 1$	$1 \cdots 1$
0	$1 \cdots 1$	$0 \cdots 0$

Fig. 5: An Object Relation r Showing that $r \not\models X \rightarrow Y$

Proposition 3 *$\omega_M(r)$ is SOI.*

Proof. The result immediately follows from Definition 13. □

We now extend Armstrong's axiom system for FDs to the class of object relations by adding the following inference rule.

(FD4) *Meta-attribute:* $\mathrm{F} \vdash M \rightarrow R$.

We need the following inference rule, which can be derivable from FD1 to FD3, to prove next theorem.

(FD5) *Union:* if $\mathrm{F} \vdash X \rightarrow Y$ and $\mathrm{F} \vdash X \rightarrow Z$, then $\mathrm{F} \vdash X \rightarrow YZ$.

The *closure* of a set of attributes, $X \subseteq R$, with respect to a given set of FDs F, denoted as X^+, is given by $X^+ = \{A \mid \mathrm{F} \vdash X \rightarrow A\}$. We now show that the axiom system comprising inference rules from FD1 to FD4 is also sound and complete for FDs, holding in the class of object relations. The method that we use is standard (c.f., see Chapter 7.3 in [38]), whose idea is first to assume that $X \rightarrow Y$ cannot be inferred from the axiom system, and then to present a relation as a counter-example in which all the dependencies of F hold except $X \rightarrow Y$. In other words, our result is that F does not logically imply $X \rightarrow Y$.

Theorem 4 *The axiom system comprising inference rules from FD1 to FD4 is sound and complete for a set of FDs F, holding in the class of object relations.*

Proof. By Proposition 1 and Theorem 2, it follows that the inference rules from FD1 to FD3 are sound. FD4 is also sound by Definition 13 and Proposition 3. We prove completeness by showing that if $\mathrm{F} \not\vdash X \rightarrow Y$, then $\mathrm{F} \not\models X \rightarrow Y$. Equivalently for the latter, it is sufficient to exhibit a relation r such that $r \models \mathrm{F}$ but $r \not\models X \rightarrow Y$. Let r be the relation shown in Figure 5, where M, X^+ and Z denote pairwise disjoint sets of attributes such that $Z = R - MX^+$. Note that $M \notin X^+$, otherwise, it is trivial that $X \rightarrow Y$ by FD4.

We first show that $r \models \mathrm{F}$. Suppose to the contrary that $r \not\models \mathrm{F}$ and thus there exists an FD, $V \rightarrow W \in \mathrm{F}$ such that $r \not\models V \rightarrow W$. It follows by the construction of r that $V \subseteq X^+$ and there exists $A \in (W \cap ZM)$ such that $A \notin X^+$. Suppose $A \in Z$. By FD1, it follows that $V \rightarrow A$ and by FD3 again, it follows that $X \rightarrow A$. This leads to a contradiction, since it follows that $A \in X^+$. Suppose $A = M$. By FD4, it follows that $M \rightarrow R$, by FD1, it follows that $M \rightarrow Y$, by FD3, it follows that $X \rightarrow M$, and finally by FD3 again, it follows that $X \rightarrow Y$. This leads to a contradiction, since we have derived $\mathrm{F} \vdash X \rightarrow Y$.

We conclude the proof by showing that $r \not\models X \rightarrow Y$. Suppose to the contrary that $r \models X \rightarrow Y$; by the construction of r, $Y \subseteq X^+$ since $M \notin X^+$. It implies that for all $A \in Y$, $\mathrm{F} \models X \rightarrow A$. Therefore, for all $A \in Y$, $\mathrm{F} \vdash X \rightarrow A$. By FD5, it follows that $\mathrm{F} \vdash X \rightarrow Y$. This leads to a contradiction, since we have derived $\mathrm{F} \vdash X \rightarrow Y$. □

4. ORDERED FUNCTIONAL DEPENDENCIES

An OFD in the ordered relational data model involves comparing the orderings between two sets of data items. We find that OFDs arise naturally in many applications, especially in those that consist of temporal data [26]. A typical example is that an OFD can capture the constraint that the salary of an employee increases every year. Another example (c.f., see [14]) is the constraint that in a bank account the chronological ordering of date increases, as does the numerical ordering

of check numbers. OFDs can also be applied to maintain the "sum of data values" relative to a set of attributes. For instance, the total production for a manufacturing plant should increase every month, or the commission earned by insurance people should increase as the total number of policies they can make from their customers.

The semantics of an OFD with two or more attributes on either the left or right hand side is defined according to lexicographical orderings and pointwise-orderings on the Cartesian product of the underlying domains of the attributes in the OFD, which gives rise to POFDs and LOFDs, respectively. From now on, OFDs mean either POFDs or LOFDs. We remark also that they are exactly the same data dependencies in the special case of unary attributes, which means that only one attribute is allowed on both the left and right hand sides of an OFD.

4.1. OFDs Arising from Pointwise-Orderings

We give the definition of a POFD as follows.

Definition 14 An *ordered functional dependency arising from pointwise-orderings* (or simply a POFD) over a relation schema R, is a statement of the form $R : X \hookrightarrow Y$ (or simply $X \hookrightarrow Y$ whenever R is understood from the context), where $X, Y \subseteq R$ are sequences of attributes. The POFD $X \hookrightarrow Y$ is said to be *standard* if $X \neq \emptyset$.

From now on, we will assume that all POFDs are standard. For non-standard POFDs, for example $\emptyset \hookrightarrow Y$, it follows from Definition 15 that r over R satisfies such a POFD if and only if $\pi_Y(r)$ is a singleton. However, we do not consider such a cardinality constraint and therefore assume that all POFDs are standard. It also implies that the special case of $\mid R \mid = 0$ is irrelevant in the context of standard POFDs. The similar comments can also be applied to LOFDs next section. We now give the definition of the semantics of a POFD.

Definition 15 A POFD, $R : X \hookrightarrow Y$, is *satisfied* in a relation r over R, denoted by $r \models X \hookrightarrow Y$, if, for all $t_1, t_2 \in r$, $t_1[X] \leq^p_X t_2[X]$ implies that $t_1[Y] \leq^p_Y t_2[Y]$.

We next give a set of inference rules for POFDs and show that Armstrong's axiom system carries over to ordered relations with respect to POFDs.

Definition 16 Let X, Y, Z, W be subsets of R and F be a set of POFDs over R. The *inference rules for* POFDs are defined as follows:

(POFD1) *Reflexivity*: if $Y \subseteq X$, then $\mathrm{F} \vdash X \hookrightarrow Y$.

(POFD2) *Augmentation*: if $\mathrm{F} \vdash X \hookrightarrow Y$, then $\mathrm{F} \vdash XZ \hookrightarrow YZ$.

(POFD3) *Transitivity*: if $\mathrm{F} \vdash X \hookrightarrow Y$ and $\mathrm{F} \vdash Y \hookrightarrow Z$, then $\mathrm{F} \vdash X \hookrightarrow Z$.

(POFD4) *Permutation*: if $\mathrm{F} \vdash X \hookrightarrow Y$, $W \sim X$ and $Z \sim Y$, then $\mathrm{F} \vdash W \hookrightarrow Z$.

We remark that POFD4 is needed because we are dealing with sequences of attributes rather than the usual sets of attributes in FDs.

Lemma 1 *Let F be a set of POFDs, $f = X \hookrightarrow Y$ be a POFD and $f^* = X \rightarrow Y$ be an FD corresponding to f. We define $F^* = \{f^* \mid f \in F\}$. Then f^* is derivable from F^* using Armstrong's axiom if and only if $F \vdash f$.*

Proof. The "if part" can be readily proved by induction on the number of steps in the inference of $X \hookrightarrow Y$ from a set of POFDs and the similar technique can be applied to the "only if" part in the inference of $X \rightarrow Y$ from a set of FDs. □

The above lemma is useful because it suggests that we can apply existing algorithms for FDs to determine whether a POFD f can be inferred from a given set of POFDs using the inference rules from POFD1 to POFD4. For example, Beeri and Bernstein's algorithm [5] can be used to compute the closure of a set of attributes with respect to a set of POFDs. We need the following rules derivable from Definition 16 to establish the soundness and completeness of the axiom system for POFDs.

Lemma 2 *The following inference rules can be derived from the inference rules in Definition 16.*

(POFD5) Decomposition*: if* $F \vdash X \hookrightarrow Y$*, then* $F \vdash X \hookrightarrow Z$*, where* $Z \subseteq Y$*.*

(POFD6) Union*: if* $F \vdash X \hookrightarrow Y$ *and* $F \vdash X \hookrightarrow Z$*, then* $F \vdash X \hookrightarrow YZ$*.*

Proof. POFD5 can be derived from POFD1 and POFD3. POFD6 can be derived from POFD2, augmenting the necessary attributes on the antecedents. □

The closure of a set of attributes X^+ in the context of POFDs is given by $X^+ = \{A \mid \mathrm{F} \vdash \mathrm{X} \hookrightarrow \mathrm{A}\}$. We now show in the following theorem that the above axiom system is sound and complete for POFDs, holding in ordered databases. The underlying idea in this proof is standard [38] and similar to Theorem 4. We also need to assume that each domain has at least two distinct elements. We believe that this assumption is reasonable in practice.

Theorem 5 *Let the common domain D contain at least two distinct elements. The axiom system comprising from POFD1 to POFD4 is sound and complete for POFDs.*

Proof. It is easy to show that the inference rules from POFD1 to POFD4 are sound. We prove completeness by showing that if $\mathrm{F} \not\vdash X \hookrightarrow Y$, then $\mathrm{F} \not\models X \hookrightarrow Y$. Equivalently for the latter, it is sufficient to exhibit a relation, say r, such that $r \models \mathrm{F}$ but $r \not\models X \hookrightarrow Y$. Let r be the relation consisting of two tuples t_1 and t_2 shown in Figure 6, where $Z = R - X^+$.

	X^+	Z
t_1	$1 \cdots 1$	$1 \cdots 1$
t_2	$1 \cdots 1$	$0 \cdots 0$

Fig. 6: A Relation r Showing that $r \not\models X \hookrightarrow Y$

Assuming that $0 < 1$, we have $1 \not\leq 0$ (i.e. $t_1[Z] \not\leq t_2[Z]$). We first show that $r \models \mathrm{F}$. Suppose to the contrary that $\mathrm{r} \not\models \mathrm{F}$, and thus there exists a POFD, $V \hookrightarrow W \in \mathrm{F}$ such that $r \not\models V \hookrightarrow W$. It follows by the construction of r and by POFD5 that $V \subseteq X^+$ and that $\exists A \in Z$ such that $A \notin X^+$. By POFD1 and POFD5, it follows that $V \hookrightarrow A$, and by POFD3, it follows that $X \hookrightarrow A$. This leads to a contradiction, since it follows that $A \in X^+$. We conclude the proof by showing that $r \not\models X \hookrightarrow Y$. Suppose to the contrary that $r \models X \hookrightarrow Y$. By the construction of r, $Y \subseteq X^+$. This leads to a contradiction, since by POFD6 we have $X \hookrightarrow Y'$, where $Y' \sim Y$. Then by POFD4 we could derive $\mathrm{F} \vdash X \hookrightarrow Y$. □

4.2. OFDs Arising from Lexicographical Orderings

We give the definition of an LOFD as follows.

Definition 17 An *ordered functional dependency arising from lexicographical orderings* (or simply an LOFD) over a relation schema R, is a statement of the form $R : X \rightsquigarrow Y$ (or simply $X \rightsquigarrow Y$ whenever R is understood from the context), $X, Y \subseteq R$ are sequences of attributes.

Similar to POFDs, we assume that all LOFDs are standard. We now give the definition of the semantics of an LOFD.

Definition 18 An LOFD, $R : X \rightsquigarrow Y$, is *satisfied* in a relation r over R, denoted by $r \models X \rightsquigarrow Y$, if, for all $t_1, t_2 \in r$, $t_1[X] \leq^l_X t_2[X]$ implies that $t_1[Y] \leq^l_Y t_2[Y]$.

We observe that the concept of POFDs and LOFDs are incomparable. A relation satisfying the POFD $X \hookrightarrow Y$ may not necessarily satisfy the LOFD $X \rightsquigarrow Y$ and conversely, a relation satisfies the LOFD $X \rightsquigarrow Y$ may not necessarily satisfy the POFD $X \hookrightarrow Y$. The following example helps to illustrate this point.

$r_1 =$

A	B	C
1	3	6
2	4	5

(a)

$r_2 =$

A	B	C
1	4	6
2	3	5

(b)

Fig. 7: Relations r_1 and r_2 Showing that POFDs and LOFDs are Incomparable

Example 3 Consider the relations r_1 and r_2 over $R = \{A, B, C\}$ shown in Figure 7. It is clear that in (a) $r_1 \models A \rightsquigarrow BC$ but $r_1 \not\models A \hookrightarrow BC$. On the other hand, in (b) $r_2 \models AB \hookrightarrow C$ but $r_2 \not\models AB \rightsquigarrow C$. □

The chase is a fundamental theorem-proving tool in relational database theory. The main uses of the chase have been to test the implications of data dependencies [27] and to test the consistency of a relational database, with respect to a set of data dependencies [17, 23]. We now extend the classical chase defined over conventional relations with respect to FDs [27, 3] to ordered relations with respect to LOFDs. The extended chase will be used as a sound and complete inference tool for LOFDs in Theorem 6. We need two operations, *equate* and *swap*, to manipulate values in ordered domains before presenting our chase rules.

Definition 19 Let $min(a, b)$ and $max(a, b)$ denote the minimum and maximum of the values a and b, respectively. For any two distinct tuples $t_1, t_2 \in r$ over R and some $A \in R$, the *equate* of t_1 and t_2 on A, denoted as $equate(t_1[A], t_2[A])$, is defined by replacing both $t_1[A]$ and $t_2[A]$ by $min(t_1[A], t_2[A])$; the *swap* of t_1 and t_2 on A, denoted as $swap(t_1[A], t_2[A])$, is defined by replacing $t_1[A]$ by $min(t_1[A], t_2[A])$ and $t_2[A]$ by $max(t_1[A], t_2[A])$, respectively.

We demonstrate how to use the equate and swap operations with the following example.

Example 4 Consider a relation r over A shown in Figure 8 (a), which consists of two tuples $t_1 = \langle 2 \rangle$ and $t_2 = \langle 1 \rangle$. We apply the equate operation of t_1 and t_2 on A, resulting in the relation shown in Figure 8 (b). We apply the swap operation of t_1 and t_2 on A, resulting in the relation shown in Figure 8 (c).

	A
t_1	2
t_2	1

(a) $r = \{t_1, t_2\}$

	A
t_1	1
t_2	1

(b) $equate(t_1[A], t_2[A])$

	A
t_1	1
t_2	2

(c) $swap(t_1[A], t_2[A])$

Fig. 8: An Example of Using the Equate and Swap Operations □

We now give the chase rules, which are applied to two tuples in a relation with respect to a set of LOFDs.

Definition 20 Let t_1 and t_2 be two tuples in r such that $t_1[X] \preceq^l_X t_2[X]$ but $t_1[Y] \not\preceq^l_Y t_2[Y]$, A be the first attribute in X such that $t_1[A] \neq t_2[A]$, if such an attribute exists, and B be the first attribute in Y such that $t_1[B] \neq t_2[B]$, then the *chase rules* for the LOFD $X \rightsquigarrow Y$ are the following:

Equate rule: if $t_1[X] = t_2[X]$ but $t_1[B] \neq t_2[B]$, then $equate(t_1[B], t_2[B])$;

Swap rule: if $t_1[A] < t_2[A]$ but $t_2[B] < t_1[B]$, then $swap(t_1[B], t_2[B])$, or if $t_2[A] < t_1[A]$ but $t_1[B] < t_2[B]$, then $swap(t_1[A], t_2[A])$.

The said chase rules cater for all the possible cases when there are two tuples in a relation violating $X \rightsquigarrow Y$. In applying the chase rules we need a fixed ordering on the tuples t_1 and t_2. If we choose different orderings on t_1 and t_2 in different applications of the rules, then the chase procedure may result in a non-terminating process. We can clarify this point by the following example.

Example 5 Let F = $\{A \rightsquigarrow B, C \rightsquigarrow B\}$ and the tuples $t_p = \langle 1, 4, 6\rangle$ and $t_q = \langle 2, 3, 5\rangle$, respectively, as shown in Figure 9 (a). First, we let $t_1 = t_p$ and $t_2 = t_q$, then apply the swap rule with respect to $A \rightsquigarrow B$, obtaining the result shown in Figure 9 (b). Now we let $t_1 = t_q$ and $t_2 = t_p$ (i.e., change the ordering of t_p and t_q), then apply the swap rule with respect to $C \rightsquigarrow B$, obtaining the result as shown in Figure 9 (c), which is the beginning relation that we have shown in Figure 9 (a).

	A	B	C
t_p (as t_1)	1	4	6
t_q (as t_2)	2	3	5

(a) before the chase

	A	B	C
t_p (as t_2)	1	*3*	6
t_q (as t_1)	2	*4*	5

(b) chase for $A \rightsquigarrow B$ on (a)

	A	B	C
t_p	1	*4*	6
t_q	2	*3*	5

(c) chase for $C \rightsquigarrow B$ on (b)

Fig. 9: An Example Showing that the Chase Procedure never Terminates □

Fortunately, this undesirable property can be removed if we impose a fixed linear ordering on r and assign t_1 as the smaller tuple and t_2 as the larger tuple with respect to this ordering. We will show in Lemma 3 that under such a condition the chase procedure always terminates. Therefore, in Example 5 if we assume the ordering of t_p and t_q is fixed as given in Figure 9 (a) throughout the chase procedure, then the process terminates and it can be checked that the final relation is obtained as shown in Figure 10.

	A	B	C
t_p (as t_1)	1	*3*	*5*
t_q (as t_2)	2	*4*	*6*

Fig. 10: The Chase Procedure Terminates in Example 5 with a Fixed Ordering

Let $r = \{t_1, \ldots, t_n\}$ be an ordered relation over R and F be a set of LOFDs with $\mid R \mid = m$. We now give the pseudo-code of an algorithm designated $CHASE(r,\text{F})$, which applies the chase rules given in Definition 20 to R as long as possible and returns the resulting relation r over R, also denoted as $CHASE(r,\text{F})$.

Algorithm 1 (**$CHASE(r,\mathbf{F})$**)

```
1.    begin
2.      Result := r = ⟨t_1, ..., t_n⟩ ;
3.      Tmp:= ∅;
4.      while Tmp ≠ Result do
5.        Tmp := Result;
6.        if ∃X ⇝ Y ∈ F, ∃ t_p, t_q ∈ Result such that
            t_p[X] ≤^l_X t_q[X] but t_p[Y] ⋠^l_Y t_q[Y]
7.        then Apply the appropriate chase rule to Result with
            t_1 = t_min(p,q) and t_2 = t_max(p,q) ;
8.      end while
9.      return Result;
10.   end.
```

Lemma 3 *$CHASE(r,F)$ in Algorithm 1 terminates and satisfies F.*

Proof. Let P_j with $1 \leq j \leq m$ be the sequence $\langle a_{1j}, \ldots, a_{nj} \rangle$, where $a_{ij} = t_i[A_j]$ (i.e., $P_j = \pi_{A_j}(Result)$), a_j^{min} be the minimum value in P_j, and P_j^{min} be the sequence $\langle a_j^{min}, \ldots, a_j^{min} \rangle$ (a sequence of n identical values). Suppose an application of a chase rule changes P_j to $P_j' = \langle a_{1j}', \ldots, a_{nj}' \rangle$. Since the chase rules neither change the value a_j^{min} nor introduce any new values into the variable *Result*, P_j^{min} is unchanged throughout the process of the chase. In order to prove that $CHASE(r,\mathrm{F})$ terminates, it suffices to show that $P_j^{min} \leq^l P_j' <^l P_j$. There are two cases to consider.

In the first case the change to P_j is due to an application of the equate rule. Then by Algorithm 1 we have $a_{pj} \neq a_{qj}$. It follows that $a_{pj}' = a_{qj}' = min(a_{pj}, a_{qj})$, and $a_{ij}' = a_{ij}$ for $i \notin \{p, q\}$. Thus, $P_j' <^l P_j$.

In the second case the change to P_j is due to an application of the swap rule. Without loss of generality we assume $p < q$. Then by Algorithm 1 $a_{qj} < a_{pj}$. It follows that $a_{pj}' = min(a_{pj}, a_{qj})$, $a_{qj}' = max(a_{pj}, a_{qj})$ and $a_{ij}' = a_{ij}$ for $i \notin \{p, q\}$. Thus, $P_j' <^l P_j$.

It is also trivial that in both cases $P_j^{min} \leq^l P_j'$, since the minimum of any two values in P_j is greater than or equal to the minimum of all values in P_j.

Due to the above consideration, it follows that $CHASE(r,\mathrm{F})$ satisfies F, otherwise we can apply one of the chase rules in Definition 20 to $CHASE(r,\mathrm{F})$, thus leading to a contradiction, since $CHASE(r,\mathrm{F})$ has not yet terminated. □

Lemma 4 *$CHASE(r,F)$ in Algorithm 1 can be computed in polynomial-time in the sizes of r and F.*

Proof. By Definition 20, we observe that lines 6 to 7 in Algorithm 1 can be executed at most $O(m)$ times for an LOFD in F, where m is the number of distinct symbols in r. There is at most $O(m)$ application of chase rules to r. So each execution of the while loop beginning in line 4 and ending at line 8 can be computed in polynomial-time in the sizes of r and F. □

Example 6 Let $\mathrm{F} = \{A \rightsquigarrow B, B \rightsquigarrow C\}$ and r be a relation consisting of three tuples $t_1 = \langle 3, 3, 2 \rangle$, $t_2 = \langle 2, 2, 1 \rangle$ and $t_3 = \langle 3, 1, 3 \rangle$, as shown in Figure 11 (a). First, we carry out the chase rules to eliminate the violation of $A \rightsquigarrow B$ as follows, apply the chase rule $swap(t_2[B], t_3[B])$ since $t_2[A] < t_3[A]$ but $t_3[B] < t_2[B]$, and then apply the chase rule $equate(t_1[B], t_3[B])$ since $t_1[A] = t_3[A]$ but $t_1[B] \neq t_3[B]$. We therefore obtain the intermediate result as shown in Figure 11 (b), which satisfies $A \rightsquigarrow B$. Second, we carry out the chase rules to eliminate the violation of $B \rightsquigarrow C$ as follows, apply the chase rule $equate(t_1[C], t_3[C])$ since $t_1[B] = t_3[B]$ but $t_1[C] \neq t_3[C]$. The chase procedure now terminates and the final result, $CHASE(r,\mathrm{F})$ is given in Figure 11 (c), which satisfies F.

	A	B	C
t_1	3	3	2
t_2	2	2	1
t_3	3	1	3

(a) r prior to the chase

	A	B	C
t_1	3	*2*	2
t_2	2	*1*	1
t_3	3	*2*	3

(b) chase for $A \rightsquigarrow B$ on (a)

	A	B	C
t_1	3	2	2
t_2	2	1	1
t_3	3	2	*2*

(c) chase for $B \rightsquigarrow C$ on (b)

Fig. 11: An Example of Obtaining $CHASE(r,\mathrm{F})$ □

We note that the result of the chase is not necessarily unique. For instance, in the above example we can apply $equate(t_1[B], t_3[B])$ first to eliminate the violation of $A \rightsquigarrow B$, then we have at least two '1's under the column of the attribute B, leading to a final result different from that given in Figure 11 (c). Although the final result of the chase may not be unique, we still can apply it in tackling the implication problem of LOFDs. This point is illustrated by the results shown in next theorem and Theorem 9.

Theorem 6 *Let r be a relation over R and F be a set of LOFDs over R. Then $r \models F$ if and only if $r = CHASE(r,F)$.*

Proof. (*IF:*) Assume to the contrary that $r \not\models$ F and thus there exists an LOFD, $X \rightsquigarrow Y \in$ F such that $r \not\models X \rightsquigarrow Y$. It follows that there must be two rows, $t_1, t_2 \in r$, such that $t_1[X] \preceq^l_X t_2[X]$ but $t_1[Y] \not\preceq^l_Y t_2[Y]$; so the chase rule for $X \rightsquigarrow Y$ can be applied to r resulting in a different relation. Hence $r \neq CHASE(r, F)$.
(*ONLY IF:*) It follows from Definition 20 that a chase rule for F can be carried out only if r violates some LOFDs in F. □

Lemma 3 and Theorem 6 are fundamental because they allow the chase procedure to be employed in order to test the satisfaction of r with respect to a set of F in a finite number of steps; many similar results for different kinds of data dependencies such as FDs, INclusion Dependencies (INDs), and Join Dependencies (JDs) can be found in [27, 12, 29]. These results provide us with a theorem-proving tool to test the consistency of a database with respect to a set of LOFDs. The chase can also be used for maintaining consistency by applying the rules in Definition 20 to fix the violation of an LOFD in relations.

In order to provide a proof procedure for LOFDs, we now define the notion of *ordered variables*. Such variables afford us the ability to infer orderings between attribute values and to set up a set of *templates for relations*, which are essentially the same concept as the tableaux used in [27, 3, 24].

Definition 21 The *variable domain* of a relation schema R, denoted by $vdom(R)$, is the finite set $\{l_1, \ldots, l_m, h_1, \ldots, h_m\}$, where $m = \mid R \mid$. The variables l_i and h_i with $i \in \{1, \ldots, m\}$ are called *low ordered variables* and *high ordered variables*, respectively. We call them collectively *ordered variables*, whose ordering is given by $l_i < h_i$.

We now define a set of relations defined over variable domains with respect to a given LOFD, which basically enumerate all the possible cases for two tuples violating the LOFD.

Definition 22 Let f be the LOFD $X \rightsquigarrow Y$ over R with $\mid X \mid = n$ and $\mid R \mid = m$. We use two short hand symbols u_i and v_i to represent one of the following three cases: (1) $u_i = l_i$ and $v_i = l_i$, (2) $u_i = l_i$ and $v_i = h_i$ or (3) $u_i = h_i$ and $v_i = l_i$. A *template relation* (or simply a template) with respect to f, denoted as r_f, is a relation consisting of two tuples, t_1 and t_2, whose underlying domain is $vdom(R)$, such that it is equal to either T_0 or T_k, where $Pre(X) = \langle x_1, \ldots, x_k \rangle$ for $1 \leq k \leq n$.

$T_0 =$

	X	$R - X$
t_1	$l_1 \cdots l_n$	$u_{n+1} \cdots u_m$
t_2	$l_1 \cdots l_n$	$v_{n+1} \cdots v_m$

$T_k =$

	$x_1 \cdots x_{k-1}$	x_k	$R - Pre(X)$
t_1	$l_1 \cdots l_{k-1}$	l_k	$u_{k+1} \cdots u_m$
t_2	$l_1 \cdots l_{k-1}$	h_k	$v_{k+1} \cdots v_m$

Fig. 12: Template Relations for an LOFD

We remark that in Definition 22 the symbols u_i and v_i represent three possibilities of combinations of l_i and h_i. Therefore, it is easy to verify that there are 3^{m-n} templates defined by T_0 and 3^{m-k} templates defined by T_k for each k. Altogether there are $3^{m-n} + (3^{m-n} + \cdots + 3^{m-1})$ $= 3^{m-n} + \frac{3^m - 3^{m-n}}{2} = \frac{3^m + 3^{m-n}}{2}$ templates. Note that there are some redundant templates in both T_0 and T_k, if we take into account the fact that there are two possible orderings for t_1 and t_2, but this does not affect the order of the upper bound of the number of templates, which is shown to be $O(3^m)$.

We apply the chase rules to a template relation using the ordering defined on a variable domain $vdom(R)$. The following proposition gives the result corresponding to Theorem 6.

Proposition 7 *Let r_f be a template relation over R and F be a set of LOFDs over R. Then $r_f \models F$ if and only if $r_f = CHASE(r_f, F)$.*

Proof. The result immediately follows from Theorem 6, where we substitute r_f for r and apply the chase rules on the ordered variables. □

A template relation can be viewed as a relation instance consisting of two tuples, in the sense that we could use an ordering isomorphism mapping values in D to low ordered variables and high ordered variables, respectively. We formalise this idea by the following definition.

Definition 23 Let $R = \{A_1, \ldots, A_m\}$ and $vdom(R) = \{l_1, \ldots, l_m, h_1, \ldots, h_m\}$. A *valuation mapping* ρ is a mapping from $vdom(R)$ to D such that $\rho(l_i) < \rho(h_i)$ for all $1 \leq i \leq m$. We extend ρ to a tuple t by $\rho(t) = \langle \rho(t[A_1]), \ldots, \rho(t[A_m]) \rangle$. We also extend ρ to a template relation r_f by $\rho(r_f) = \{\rho(t_1), \rho(t_2)\}$.

The next proposition states that if there is a valuation mapping relating a template relation to a relation having two tuples, then they satisfy the same set of LOFDs.

Proposition 8 *Let $\rho(r_f) = r$, where r is a relation over R having two tuples. Then $r_f \models X \rightsquigarrow Y$ if and only if $r \models X \rightsquigarrow Y$.*

Proof. The result immediately follows from Definition 23, since r_f is isomorphic to r and the ordering of data values in the ith column of r corresponds to the ordering of the ordered variables l_i and h_i. □

The following example shows how to apply a valuation mapping to a template relation.

Example 7 Consider the template relation r_f over $\{A, B, C\}$ with respect to the LOFD f, $A \rightsquigarrow BC$, which is shown in Figure 13 (a). We define the valuation mapping ρ by $\rho(l_1) = 1$, $\rho(l_2) = 2$, $\rho(h_2) = 3$, $\rho(l_3) = 4$ and $\rho(h_3) = 5$. Then we have $\rho(r_f)$, shown in Figure 13 (b). Note that in this example r_f is one of the templates defined by T_0 in Definition 22.

$r_f =$

A	B	C
l_1	l_2	h_3
l_1	h_2	l_3

(a)

$\rho(r_f) =$

A	B	C
1	2	5
1	3	4

(b)

Fig. 13: An Example Showing the Application of a Valuation Mapping □

We now extend the notion of tableaux for an LOFD f to be a set of templates. The tableaux in our case is different from that for FDs, which just requires a single template for FDs (see Theorem 4.2 in [3]). We define *tableaux*, denoted by T_f, to be the set of all template relations in Definition 22.

Definition 24 The chase of T_f, denoted as $CHASE(T_f, F)$, is defined by $CHASE(T_f, F) = \{CHASE(r_f, F) \mid r_f \in T_f\}$. $CHASE(T_f, F)$ satisfies $X \rightsquigarrow Y$, denoted by $CHASE(T_f, F) \models X \rightsquigarrow Y$, if, for all $r_f \in T_f$, $CHASE(r_f, F) \models X \rightsquigarrow Y$. Furthermore, $CHASE(T_f, F)$ satisfies F, denoted by $CHASE(T_f, F) \models$ F, if, for all $X \rightsquigarrow Y \in$ F, $CHASE(T_f, F) \models X \rightsquigarrow Y$. A *valuation mapping* of T_f is a valuation mapping of some r_f in T_f.

The following theorem shows that the chase rules can be also viewed as a sound and complete inference procedure for LOFDs.

Theorem 9 *Let F be a set of LOFDs over R and f be an LOFD $X \rightsquigarrow Y$. Then $CHASE(T_f, F) \models f$ if and only if $F \models f$.*

Proof. (*IF:*) Assume $CHASE(T_f, F) \not\models f$. By Definition 24, there exists $r_f \in T_f$ such that $CHASE(r_f, F) \not\models f$ but $CHASE(r_f, F) \models$ F. Note that $CHASE(r_f, F)$ is a template which can be viewed as an instance. Therefore, we have a valuation mapping ρ to generate a relation

instance $\rho(CHASE(r_f, F))$ and by Proposition 8, $\rho(CHASE(r_f, F)) \models$ F but $\rho(CHASE(r_f, F)) \not\models f$. This leads to a contradiction.
(*ONLY IF:*) We let w_1, w_2 be any two tuples in a relation r such that $w_1 \leq^l_X w_2$. We claim $w_1 \leq^l_Y w_2$. Let $s_f \in T_f$ be the template relation such that $\rho(t_1) = w_1$ and $\rho(t_2) = w_2$. We can always find such an s_f because T_f exhausts all possibilities of two tuples which satisfy the condition $w_1 \leq^l_X w_2$. Thus we have $\rho(s_f) = \{w_1, w_2\}$ and $\rho(s_f) \models$ F. By Proposition 8, we have $s_f \models$ F. It follows by Proposition 7 that $s_f = CHASE(s_f, F)$. Since we have assumed that $CHASE(T_f, F) \models f$, we have $CHASE(s_f, F) \models f$. Thus, $\rho(CHASE(T_f, F)) = \rho(s_f) = \{w_1, w_2\}$, which implies that $w_1 \leq^l_Y w_2$ as required. □

The following corollary is an immediate result of Theorem 9.

Corollary 10 *Let F be a set of LOFDs over R. The chase procedure is a decidable, sound and complete inference algorithm for LOFDs.*

Proof. The result immediately follows from Theorem 9 and the notions of soundness and completeness. □

The above corollary shows that the chase rules together with tableaux can be used to provide a systematic way to solve the implication problem for LOFDs. We summarise the relationships between the satisfaction of POFDs, LOFDs and FDs in a relation r by the following proposition.

Proposition 11 *Let r be a relation. The following statements are true.*

1. *If* $r \models X \hookrightarrow Y$, *then* $r \models X \rightarrow Y$.
2. *If* $r \models X \rightsquigarrow Y$, *then* $r \models X \rightarrow Y$.

Proof. Let $t_1, t_2 \in r$ such that $t_1[X] = t_2[X]$. Thus, we have $t_1[X] \leq^p_X t_2[X]$ and $t_2[X] \leq^p_X t_1[X]$. By the assumption in Part 1, it follows that $t_1[Y] \leq^p_Y t_2[Y]$ and $t_2[Y] \leq^p_Y t_1[Y]$. So $t_1[Y] = t_2[Y]$. The proof is similar for Part 2 (replacing $\leq^p$ by $\leq^l$). □

From the above proposition, we can deduce that the set of relations which satisfy a set of POFDs (or LOFDs) is a subset of relations which satisfy the corresponding set of FDs F^*, where F^* is defined as $\{X \rightarrow Y \mid X \hookrightarrow Y \in$ F (or $X \rightsquigarrow Y \in$ F)$\}$.

4.3. Database Design Issues with Respect to OFDs

Relational database design plays an important role in relational database theory and thus it is extensively covered in most database textbooks [38, 30, 3]. Relational database design can be viewed as the process of replacing a relation schema R, together with a set of data dependencies over R by a set of relational schemas $\mathbf{R}$. We call $\mathbf{R}$ a *decomposition* of R if $\bigcup^n_{i=1} R_i = R$ and $R_i \subseteq R$ for all $R_i \in \mathbf{R}$.

There are many criteria suggested in the literature to capture the notion of an appropriate decomposition in conventional databases [30]. One desirable property is that a decomposition $\mathbf{R}$ possesses the property of *lossless join* (or simply is lossless), meaning that $\bowtie^n_{i=1} \pi_{R_i}(r) = r$, where $\bowtie$ is the *natural join* operator [38, 3]. This is because in practice a query usually involves the join of many relations and this property guarantees that a relation can be recovered from its projections. Another desirable property which leads to good database design is Boyce-Codd Normal Form (BCNF) in conventional databases. BCNF requires that for every FD $X \rightarrow Y$ over $R_i \in \mathbf{R}$, X is a superkey. This property takes into consideration the importance of FDs in conventional databases, since they generalise the important notions of entity integrity and keys [9]. The definitions of a key, a superkey and BCNF can be naturally extended into the context of ordered databases.

Definition 25 Let F be a set of POFDs (or LOFDs) over $\mathbf{R}$ and let $R \in \mathbf{R}$. A sequence of attributes $X \subseteq R$ is a *superkey* for R with respect to F if F $\models R : X \hookrightarrow R$ (or F $\models R : X \rightsquigarrow R$). A sequence of attributes $X \in R$ is a *key* for R with respect to F if X is a superkey for R and there does not exist a proper subset Y of X such that Y is a superkey for R. A database schema $\mathbf{R}$ is in *Boyce-Codd normal form* (BCNF) with respect to a set of OFDs F over $\mathbf{R}$ if, for every OFD, X is a superkey for R.

We now examine a basic result related to database design in ordered databases, which states that if an FD $X \rightarrow Y$ holds in a database over a schema $R = XYZ$, then the decomposition $\mathbf{R} = \{XY, XZ\}$ of R is lossless, meaning that $r = \pi_{XY}(r) \bowtie \pi_{XZ}(r)$ (c.f., Theorem 7.5 in [38]). This property of FDs forms the basis of an algorithm to obtain a BCNF database schema, resulting in the lossless join of a decomposition having two schema components. We present the similar result of lossless decomposition for OFDs as follows.

Theorem 12 *Given a relation scheme $R = XYZ$ with an OFD, either $X \hookrightarrow Y$ or $X \rightsquigarrow Y$, then the relation scheme R has a lossless decomposition into two schema components $R_1 = XY$ and $R_2 = XZ$.*

Proof. By Proposition 11, it follows that $X \hookrightarrow Y$ or $X \rightsquigarrow Y$ implies $X \rightarrow Y$. Thus, it is a lossless join. □

The converse of the above theorem holds [36] in the context of conventional FDs. However, we observe that a similar result does not hold for OFDs, even when we consider unary OFDs. Let us consider the following counter-example.

Example 8 Consider a relation r over $R = \{A, B, C\}$ decomposed into r_1 over $R = \{A, B\}$ and r_2 over $R_2 = \{B, C\}$, all of which are given in Figure 14. It is clear that neither of the following holds in r: $B \hookrightarrow A$, $B \hookrightarrow C$, $B \rightsquigarrow A$ or $B \rightsquigarrow C$.

$r =$

A	B	C
1	4	5
2	3	6

$r_1 =$

A	B
1	4
2	3

$r_2 =$

B	C
4	5
3	6

Fig. 14: A Decomposition of r into r_1 and r_2 □

In order to investigate whether there is any necessary condition for having a lossless decomposition in an ordered relation, we are still working on characterisation of the set of OFDs such that the converse of Theorem 12 can hold.

5. CONCLUDING REMARKS

We have extended the relational data model to incorporate linearly ordered domains, which are essential to the existing primitive data types used in DBMSs as well as many advanced applications such as temporal information. Within the extended model, we defined ordered databases in which we introduced OFDs. We used the notion of SOI to give a new view of conventional FDs in the context of ordered relations as stated in Theorem 2. Furthermore, we extended Armstrong's axiom system for FDs to object relations in Theorem 4. We have studied the implication problems of OFDs, which are classified into two categories, POFDs and LOFDs, according to whether they arise from pointwise-orderings or lexicographical orderings on the Cartesian products of underlying domains. In the special case of unary OFDs, these two categories are identical. We presented a sound and complete axiom system for POFDs in Theorem 5. We also presented a set of sound and complete chase rules for LOFDs in Definition 20, which can be employed as a theorem-proving tool for LOFDs, as indicated in Theorem 9.

We believe that the scope of the application of the chase rules for LOFDs has not been fully developed and thus we are still investigating the further use of them. For example, the chase may be employed as a starting point to design new inference procedures for investigating the interactions between LOFDs and other known data dependencies such as INDs and JDs. Besides, it would also be interesting to study the semantics of INDs in ordered databases, since INDs generalise the notions of referential integrity and foreign keys [9]. We are also currently generalising the linearly ordered domain to partially ordered domains in order to capture richer semantics of ordered data.

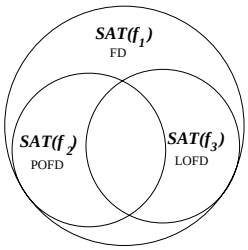

Fig. 15: Satisfaction of Various OFDs in Databases

In this case the chase for LOFDs in Definition 20 should be extended, since there may not be a unique maximum (or minimum) element in a partially ordered set.

Finally, we compare the satisfaction of an OFD in ordered databases introduced in this paper as the diagram given in Figure 15 (the scale here is irrelevant). We let $SAT(f)$ be a set of database instances that satisfy a data dependency f, and $f_1 = X \rightarrow Y$, $f_2 = X \hookrightarrow Y$ and $f_3 = X \rightsquigarrow Y$. We remark that if X and Y are unary, then in general we have $SAT(f_2) = SAT(f_3)$.

Acknowledgements — The author would like to thank Mark Levene, and the anonymous referees for their constructive comments.

REFERENCES

[1] S. Abiteboul and S. Ginsburg. Tuple sequences and lexicographical indexes. *Journal of the Association for Computing Machinery*, **33**(3):409-422 (1986).

[2] W.W. Armstrong. Dependency structures of data base relationships. In *Proceedings of the IFIP Congress*, Stockholm, pp. 580-583, North-Holland (1974).

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

[4] B.R. Badrinath and T. Imielinski. Replication and Mobility. In *Proceedings of the 2nd IEEE Workshop on Management of Replicated Data*, Paris, pp. 9-12, IEEE Computer Society (1992).

[5] C. Beeri and P.A. Bernstein. Computational problems related to the design of normal form relational schemas. *ACM Transactions on Database Systems*, **4**(1):30-59 (1979).

[6] J. Biskup. Boyce-codd normal form and object normal forms. *Information Processing Letters*, **32**:29-33 (1989).

[7] P. Buneman, A. Jung and A. Ohori. Using powerdomains to generalise relational databases. *Theoretical Computer Science*, **9**(1):23-55 (1991).

[8] E.F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, **13**(6):377-387 (1970).

[9] E.F. Codd. Extending the database relational model to capture more meaning. *ACM Transactions on Database Systems*, **4**(4):397-434 (1979).

[10] C.J. Date. *Relational Database Writings 1985-1989*. Addison-Wesley (1990).

[11] C.J. Date. *A Guide to the SQL Standard*. Addison-Wesley, 4th edition (1997).

[12] D.S. Johnson and A. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. *Journal of Computer and System Sciences*, **28**(1):167-189 (1984).

[13] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Co., New York (1979).

[14] S. Ginsburg and R. Hull. Order dependency in the relational model. *Theoretical Computer Science*, **26**(1-2):149-195 (1983).

[15] S. Ginsburg and K. Tanaka. Computation-tuple sequences and object histories. *ACM Transactions on Database Systems*, **11**(2):186-212 (1986).

[16] S. Ginsburg and R. Hull. Sort sets in the relational model. *Journal of the Association for Computing Machinery*, **33**(3):465-488 (1986).

[17] G. Grahne. Dependency satisfaction in databases with incomplete information. In *Proceedings of the 10th International Conference on Very Large Data Bases*, Singapore, pp. 37-45, Morgan Kaufmann (1984).

[18] G. Gratzer. *General Lattice Theory*. NewYork, Academic Press (1978).

[19] R.H. Guting, R. Zicari and D.M. Choy. An algebra for structured office documents. *ACM Transactions on Office Information Systems*, **7**(4):123-157 (1989).

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

[21] P. Honeyman. Testing satisfaction of functional dependencies. *Journal of the ACM*, **29**(3):668-677 (1982).

[22] G. Koch and K. Loney. *Oracle: The Complete Reference, 3rd Edition*. Osborne McGraw-Hill (1995).

[23] M. Levene and G. Loizou. Maintaining consistency of imprecise relations. *The Computer Journal*, **39**(2):114-123 (1996).

[24] M. Levene and G. Loizou. Null inclusion dependencies in relational databases. *Information and Computation*, **136**(2):67-108 (1997).

[25] L. Libkin. *Aspects of Partial Information in Databases*. Ph.D. Thesis, University of Pennsylvania, United States (1996).

[26] N.A. Lorentzos. DBMS support for time and totally ordered compound data types. *Information Systems*, **17**(5):347-358 (1992).

[27] D. Maier, A.O. Mendelzon and Y. Sagiv. Testing implication of data dependencies. *ACM Transactions on Database Systems*, **4**(4):455-469 (1979).

[28] D. Maier and B. Vance. A call to order, In *ACM Sympsium on Principles of Databases Systems*, Washington, D.C., pp. 1-16, ACM Press (1993).

[29] H. Mannila and K-J Raiha. *Generating Armstrong Databases for Sets of Functional and Inclusion Dependencies*. Research Report, A-1988-7, University of Tampere, Finland (1988).

[30] H. Mannila and K-J Raiha. *The Design of Relational Databases*. Addison-Wesley (1992).

[31] W. Ng and M. Levene. An extension of OSQL to support ordered domains in relational databases. In *IEEE Proceedings of the International Database Engineering and Applications Symposium, IDEAS97*, Montreal, Canada, pp. 358-367, IEEE Computer Society (1997).

[32] W. Ng and M. Levene. The development of ordered SQL packages for modelling advanced applications. In *LNCS 1308: Database and Expert Systems Application, 8th International Conference, DEXA '97, Proceedings,* Toulouse, France, pp. 529-538, Springer-Verlag (1997).

[33] K.V.S.V.N. Raju and A.K. Majumdar Fuzzy functional dependencies and lossless join decomposition of fuzzy relational database systems. *ACM Transactions on Database Systems*, **13**(2):129-166 (1988).

[34] D. Raymond. *Partial Order Databases*. Ph.D. Thesis, University of Waterloo, Canada (1996).

[35] R. Read. *Towards Multiresolution Data Retrieval via the Sandbag*. Ph.D. Thesis, University of Texas at Austin, United States (1995).

[36] J. Rissanen. Independent components of relations. *ACM Transactions on Database Systems*, **2**(4):317-325 (1977).

[37] P. Seshadri, M. Livny and R. Ramakrishnan. The design and implementation of a sequence database system. *Proceedings of the 22th International Conference on Very Large Data Bases*,, Mumbai, India, pp. 99-110, Morgan Kaufmann (1996).

[38] J.D. Ullman. *Principles of Database and Knowledge-Base Systems, Vol. I*, Rockville, MD., Computer Science Press (1988).

[39] S.L. Vandenberg and D.J. DeWitt. Algebraic support for complex objects with arrays, identity, and inheritance. In *Proceedings of ACM SIGMOD Conference*, Denver, CO, pp. 158-168, ACM Press (1991).

[40] C. Zaniolo. Database relations with null values. *Journal of Computer and System Science*, **28**(1):142-166 (1984).