A LESS COSTLY CONSTRAINTS CHECKING FOR JOIN DEPENDENCY

K.P. Tan

Department of Computer Science, National University of Singapore,
Lower Kent Ridge Road, Singapore 0511, Singapore.

Abstract

A set of n tuples in a relation of a relational database design is tested upon the constraints of the join dependency. Some constraint equalities are found to be redundant. To remove this superfluity, the universe of attributes is partitioned into n disjoint sets and a new notation of join dependency is introduced. The checking time in each run of n tuples is significantly reduced by a factor of (n-1)/2 when n > 3. The result of less costly constraints checking is of great importance for a large number of tuples in a relation.

Key Words and Phrases: database design, join dependency, constraint equalities

1. Introduction

In relational database theory, a relation is simply a set of tuples or a table with one column for each attribute and one row for each tuple. Normalization rules provide guidelines for relational schema design in order to prevent update anomalies and data inconsistencies. The study of integrity constraints was initiated by Codd in the functional dependency (FD) on the second and the third normal forms in 1971 [Cal] and [COG]. Six years later, the fourth normal form was defined in terms of multivalued dependency (MVD) in a many-to-many relationship [Fa1]. These two classes of dependencies have captured a great amount of semantic information of the real world. As an extension of MVD, mutual dependency (MD) [Nt] and hierarchical dependency (HD) [He] were proposed. In 1978, Kissanen introduced a very powerful data constraint, known as the join dependency (JD), [K]. The characteristics of its lossless join decomposition leads to the formation of the fifth normal form [Fa2]. Except FD, the other above mentioned dependencies are just the special cases of JD. Because of the particular importance of JD in the database design, a great many of the papers have explored the properties and the complete axiomatization of the JD, [ABU], [BV] and [Sc]. Checking whether a relation instance is in the fifth normal form, we need to check it against the JD. In this paper, we will provide a method to reduce the cost of constraints checking for a particular class of JD which is to be characterized.

2. Join Dependency

The join dependency [Ma] and [U1] is defined below:

Let \( R = \{ X_1, X_2, \ldots, X_n \} \) be a set of relation schemes over the universe \( U \). A relation \( r(U) \) satisfies the n-JD, \( *[X_1, X_2, \ldots, X_n] \) if \( r \) decomposes lossless onto \( n \) projections, \( X_1, X_2, \ldots, X_n \):

\[
r = \eta_{X_1}(r) \bowtie_{X_2}(r) \bowtie \cdots \bowtie_{X_n}(r)
\]

that is, \( r \) is the natural join of its projections onto the \( X_i \)'s.

In other words, if \( r \) contains tuples \( t_1, t_2, \ldots, t_n \) such that

\[
t_i(X_i \cap X_j) = t_j(X_i \cap X_j)
\]

for all \( i \) and \( j \)
then

\[ r \text{ must contain a tuple } t \text{ in } R \]

such that

\[ t(x_i) = t'(x_i) \quad 1 \leq i \leq n \quad (3) \]

3. Redundant Constraints Checking in JD

Assume \( R(U) \) is a relation scheme over 4 subsets, \( X_1, X_2, X_3, \) and \( X_4 \) in the universe \( U \) and each \( X_i \) is made up of some of the attributes in the universe, such as \( A, B, C, D, E \) and \( F \). For instance,

\[
\begin{align*}
X_1 & = ABCOE \\
X_2 & = CDEF \\
X_3 & = ABDEF \\
X_4 & = ABCF
\end{align*}
\]

According to the definition of 4-JD, all instances of \( R \) will obey JD if any set of 4 tuples, say \( t_1, t_2, t_3, \) and \( t_4 \), is in an instance such that

\[
\begin{align*}
t_1[CDE] & = t_2[CDE] \quad (5) \\
t_1[ABDE] & = t_3[ABDE] \quad (6) \\
t_1[ABC] & = t_4[ABC] \quad (7) \\
t_2[DEF] & = t_3[DEF] \quad (8) \\
t_2[CF] & = t_4[CF] \quad (9) \\
t_3[ABF] & = t_4[ABF] \quad (10)
\end{align*}
\]

there exists a tuple \( t \) in the instance such that

\[
\begin{align*}
t[D] & = t_1[D] \\
t[F] & = t_2[F] \\
t[AB] & = t_3[AB] \\
t[C] & = t_4[C]
\end{align*}
\]

When the first two constraint equalities are satisfied, then it follows that \( t_2[DE] = t_3[DE] \) and therefore testing equality (8). Obviously, given four tuples and checking whether they obey constraint equalities \( (5) \) to \( (10) \) in the above example will lead to some redundant checkings (i.e., equalities \( (b) \) and \( (9) \)). However, testing the following set of four equations:

\[
\begin{align*}
t_1[DE] & = t_2[DE] = t_3[DE] \quad (15) \\
t_2[F] & = t_2[F] \quad (16) \\
t_3[AB] & = t_4[AB] \quad (17) \\
t_4[C] & = t_1[C] = t_4[C] \quad (18)
\end{align*}
\]

is sufficient to ensure that equalities \( (5) \) to \( (10) \) are satisfied. Suppose we refer to the comparison of an attribute value for two tuples (e.g. \( t_3[B] = t_4[B] \)) as an elementary checking. Testing the first set of equalities requires 18 elementary checkings whereas only 12 elementary checkings are needed for the second set. Although the saving of 6 elementary checkings for a set of 4 tuples is not much, similar equality checkings have to be done for every set of 4 tuples taken from the relation and the number of tuples in a relation may be very large. Therefore, the cost of checking whether a relation obeying the JD can be diminished significantly by using the second set of equalities. In the following section, we will introduced a new notation which is a particular class of JD and will yield directly the reduced set of constraint equalities.

4. Notation of New n-JD

The new n-JD is defined below:

Let \( R(U) \) be a relation over subsets \( X_1, X_2, \ldots X_n \) in the universe \( U \),

where \( U = \bigcup_{i=1}^{n} X_i \)

and it is possible to partition \( U \) into \( n \) disjoint sets, \( Y_1, Y_2, \ldots Y_k \)

where \( U = \bigcup_{k=1}^{n} Y_k \)

such that

1) each \( X_i \) is a union of \((n-1) Y_k \)'s in the cyclic form,

2) each \( Y_k \) is a union of some attributes in the universe

The n-JD, \( ^n \{X_1, X_2, \ldots X_n \} \) holds in \( R(U) \)

iff whenever there are \( n \) tuples \( t_1, t_2, \ldots t_n \) \in \( R \)

such that

\[
\begin{align*}
\Psi_{i=1}^{k} & = t_1[X_i] = \ldots \\
\Psi_{k=1}^{n} & = \bigcup_{i=1}^{n} X_i
\end{align*}
\]

where \( p = n-2 \), the number of intersections among \( X_i \)'s.

\( \exists \) a tuple \( t \in R \)

such that

\[
\Psi_{i=1}^{k} \quad t[X_i] = t_1[X_i]
\]

Remark:

1) the notation \( (\bigcup_{i=q}^{n-q'}) \) (plus modulo \( n \)), is defined as follows:

\[
given \text{ integers } q \text{ and } q' \leq n
\]

Proceedings of the Tenth International Conference on Very Large Data Bases.
\[ q + q' = \begin{cases} 
q + q' & \text{if } q + q' \leq n \\
q + q' - n & \text{if } q + q' > n 
\end{cases} \]  
(21)

2) \[ \bigwedge_{i=k}^{p} x_i = x_k \cap x_{k+1} \cap \ldots \cap x_p \]  
(22)

3) \[ Y_k \text{'s in the cyclic form in } X_i \text{ means that} \]

\[ Y_{i=1}, \ldots, n \cdot X_i = \bigcup_{k=1}^{n} Y_k \]  
(23)

Besides the modification on the constraint equations, the result (i.e., the new tuple generated) of the new \( n \)-JD is the same as that of the old \( n \)-JD.

Consider the example in Section 3. The new \( n \)-JD is satisfied whenever there are tuples \( t_1, t_2, t_3 \) and \( t_4 \) in \( R \) such that

\[ t_1[X_1 \cap x_2 \cap x_3] = t_2[X_1 \cap x_2 \cap x_3] \]
\[ = t_3[X_1 \cap x_2 \cap x_3] \]  
(24)

\[ t_2[X_2 \cap x_3 \cap x_4] = t_3[X_2 \cap x_3 \cap x_4] \]
\[ = t_4[X_2 \cap x_3 \cap x_4] \]  
(25)

\[ t_3[X_3 \cap x_4 \cap x_1] = t_4[X_3 \cap x_4 \cap x_1] \]
\[ = t_1[X_3 \cap x_4 \cap x_1] \]  
(26)

\[ t_4[X_4 \cap x_1 \cap x_2] = t_1[X_4 \cap x_1 \cap x_2] \]
\[ = t_2[X_4 \cap x_1 \cap x_2] \]  
(27)

there exists a tuple \( t \) in \( R \) such that

\[ t[X_1] = t_1[X_1] \]  
(28)

\[ t[X_2] = t_2[X_2] \]  
(29)

\[ t[X_3] = t_3[X_3] \]  
(30)

\[ t[X_4] = t_4[X_4] \]  
(31)

As a result, equations (15) to (18) are derived from the new constraint equations from (24) to (27) and are equivalent to those useful constraints, such as (5), (7), (8) and (10) in the old JD. Also, equations (28) to (31) lead to the results in equations (11) to (14).

5. Comparison of Old JD with New JD

In order to compare the old JD with the new

Proceedings of the Tenth International Conference on Very Large Data Bases.

Singapore, August, 1984
Table 1. Comparison of Constraint Equalities and Elementary Checkings of Old n-JD with New n-JD

Where \( n \) is the number of subsets and \( k \), the total number of attributes in the universe \( U \). \( X_i \) and \( Y_k \) denote a subset and a disjoint set of attributes respectively.

<table>
<thead>
<tr>
<th></th>
<th>Old n-JD</th>
<th>New n-JD</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of ( X_i )'s</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>No. of ( Y_k )'s in each ( X_i )</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>No. of constraint equations</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>No. of equalities in each equation</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>No. of intersections in each equation</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>No. of resultant ( Y_k )'s in each equation</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>No. of occurrences of each ( Y_k )</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>No. of ( Y_k ) checks</td>
<td>3x1</td>
<td>6x2</td>
</tr>
<tr>
<td>No. of elementary checks</td>
<td>1xk</td>
<td>3xk</td>
</tr>
</tbody>
</table>

Appendix: Comparison of the Constraints Between Old 4-JD and New 4-JD

In the new 4-JD, we have
1) 4 subsets of attributes, \( X_1, X_2, X_3, \) and \( X_4 \) in the universe,
2) each subset contains 3 out of 4 disjoint sets, \( Y_1, Y_2, Y_3 \) and \( Y_4 \) in the cyclic union.
3) each \( Y_k \) may contain any number of attributes, \( A_i \) say, \( j=1 \) to 10,
e.g., \( Y_1 = A_1A_2A_3, Y_2 = A_4A_5, Y_3 = A_6A_7A_8, \) and \( Y_4 = A_9A_{10} \)

Thus, we have
\[
\begin{align*}
X_1 &= Y_1Y_2Y_3 \\
X_2 &= Y_2Y_3Y_4 \\
X_3 &= Y_3Y_4Y_1 \\
X_4 &= Y_4Y_1Y_2 \\
\end{align*}
\]

So, each \( Y_k \) is a domain, containing some attributes which are related to each other.

Thus,
\[
\begin{align*}
&Y_1 \\
&Y_2 \\
&Y_3 \\
&Y_4 \\
\end{align*}
\]

Take the supermarket inventory as an example, we may have 10 attributes in the universe \( U \), such as

- \( \text{INO} \) = Item number
- \( \text{INAME} \) = Item name
- \( \text{ISP} \) = Item specification
- \( \text{MINST} \) = Minimum stock
- \( \text{MAXST} \) = Maximum stock
- \( \text{QOH} \) = Quantity on hand
- \( \text{QOO} \) = Quantity on order
- \( \text{QSTD} \) = Quantity of total sale-to-date
- \( \text{CPK} \) = Cost price
- \( \text{PKGU} \) = Package unit

Thus, each \( A_i \) is a domain, containing some attributes which are related to each other.

Proceedings of the Tenth International Conference on Very Large Data Bases.

Singapore, August, 1984
<table>
<thead>
<tr>
<th>Old 4-JD</th>
<th>New 4-JD</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ t_1(x_1 \cap x_2) = t_2(x_1 \cap x_2) ]</td>
<td>[ t_1(x_1 \cap x_2 \cap x_3) = t_2(x_1 \cap x_2 \cap x_3) ]</td>
</tr>
<tr>
<td>[ t_1(x_1 \cap x_3) = t_3(x_1 \cap x_3) ]</td>
<td>[ t_2(x_2 \cap x_3 \cap x_4) = t_3(x_2 \cap x_3 \cap x_4) ]</td>
</tr>
<tr>
<td>[ t_1(x_1 \cap x_4) = t_4(x_1 \cap x_4) ]</td>
<td>[ t_3(x_3 \cap x_4 \cap x_1) = t_4(x_3 \cap x_4 \cap x_1) ]</td>
</tr>
<tr>
<td>[ t_2(x_2 \cap x_3) = t_3(x_2 \cap x_3) ]</td>
<td>[ t_4(x_4 \cap x_1 \cap x_2) = t_1(x_4 \cap x_1 \cap x_2) ]</td>
</tr>
<tr>
<td>[ t_2(x_2 \cap x_4) = t_4(x_2 \cap x_4) ]</td>
<td>[ t_2(x_4 \cap x_1 \cap x_2) = t_2(x_4 \cap x_1 \cap x_2) ]</td>
</tr>
<tr>
<td>[ t_3(x_3 \cap x_4) = t_4(x_3 \cap x_4) ]</td>
<td></td>
</tr>
</tbody>
</table>

That is, \[ t_1(Y_{1,2}) = t_2(Y_{2,3}) \]
\[ t_1(Y_{3,1}) = t_3(Y_{3,1}) \]
\[ t_1(Y_{2,1}) = t_4(Y_{1,2}) \]
\[ t_2(Y_{3,4}) = t_3(Y_{3,4}) \]
\[ t_2(Y_{2,4}) = t_4(Y_{2,4}) \]
\[ t_3(Y_{4,1}) = t_4(Y_{4,1}) \]

1) 6 equalities in 6 equations
2) 1 intersection in each equality
3) 2 resulting disjoint sets in each equality
4) each disjoint set occurs 3 times in the 6 equations
5) need (3x4) or (6x2) disjoint set checkings

Suppose \( Y_1, Y_2, Y_3 \) and \( Y_4 \) contains \( n_1, n_2, n_3 \) and \( n_4 \) attributes respectively. There are \( k \) attributes altogether,

i.e., \( k = n_1 + n_2 + n_3 + n_4 \) e.g., \( k = 10 \) in the above example

6) need 3xk elementary checkings

adapting the new JD, we need only two-thrds of the time required for the old JD. Therefore, the checking using the new JD is always less time consuming.

References


Proceedings of the Tenth International Conference on Very Large Databases.


