8=papers.

00-OD

ISCA

Proceedings of the ISCA
2nd International Conference

INFORMATION REUSE AND INTEGRATION

Honolulu, Hawaii U.S.A.
November 1 - 3, 2000

Editors: S. N. J. Murthy, S-C. Chen

A Publication of
The International Society for
Computers and Their Applications - ISCA

ISBN: 1-880843-36-6

On Database Integration over Intelligent Networks*

Naphtali Rishe, Jun Yuan, Jinyu Meng, Shu-Ching Chen, and Ouri Wolfson+

High Performance Database Research Center
School of Computer Science
Florida International University
University Park, ECS 243, Miami, FL 33199
{rishen, yuanj, mengj, chens@cs.fiu.edu}

+Department of Electrical Engineering and Computer Science
University of Illinois at Chicago
1030 North State Street, Suite M45, Chicago, IL 60610
{ wolfson@uic.edu }

Abstract

The advent of intelligent network technology has brought a kind of new challenge to the database community, especially on the heterogeneous database integration techniques. These advances in public network technology enable the implementation of a new type of distributed database systems with new demand for improved query distribution and query optimization algorithms. Query response time is no longer the only key point regarding distributed query execution. In stead, more and more people are concerning more about their monetary cost for the communication. A research topic that immediately arises is cost minimization for network usage. Previous research on distributed database systems has produced results that can be utilized for this purpose. Nevertheless, Intelligent Network and ISDN usage requires current knowledge to be adapted and extended. In this paper, we present both our database integration methodology over intelligent networks and our prototyping system over ISDN.

1. Introduction

Today, the network technology develops extremely fast to meet the requirement of information sharing among customers and business organizations. From time to time, new versions of these networks are expected to be available to business and consumers because people are always expecting to retrieve information across the networks with faster speed and less payment. For example, Intelligent public networks, such as ISDN, have become available in North America during the recent a couple of years. These networks can be used to provide distributed computing to those who have found it too expensive in the past, and can provide cost savings over traditional distribution techniques. In the mean time, this phenomenon is significant to the database research because it renders many assumptions and parameters in conventional distributed database systems obsolete, and database researchers need therefore to pay more attention to the new challenges brought by those newly emerging technologies.

There have been many theoretical studies in database integration [1][2][3], as well as a number of prototyping systems, such as, Information Manifold [4], Multibase [5], MRDSM [6], Pegasus [7], Carnot [8], SIMS [9], TSIMMIS [10][11], and SEMHDB [15][16], each representing a different methodology. However, all these previous work were performed on basis of traditional distributed database environments where the databases are connected by privately owned networks. As intelligent network technology, for example ISDN, emerges, the distributed databases can now communicate through public networks provided by the telecommunication companies. The transition from private networks to intelligent public networks has changed the characteristics of the networks from static to dynamic in the sense that the network resources can now be dynamically allocated and manipulated by the database systems so as to achieve best utilization and performance. This change requires rethinking of the database integration techniques including the distributed query optimization techniques because the assumptions that were usually made about the networks in the previous studies are no longer true. Therefore, new efforts are desired for database integration over intelligent networks.

This paper will present our new database integration methodology over intelligent networks, By further developing our original idea in [12], it extends the current

* This research was supported in part by Rome Labs (F30602-98-C-0037), NASA (under grants NAGW-4080, NAG5-5095, NAS5-97222, and NAG5-6830) and NSF (CDA-9711582, IRI-9409661, HRD-9707076, and ANI-9876409).

knowledge of database integration techniques to include methods that optimize queries for these new networks. A prototyping system that has been implemented based on our approach will be introduced in this paper as well.

The remainder of the paper is organized as follows. Section 2 presents our proposed query distribution methodology by discussing some main research issues such as our basic model, how to reduce search space for distributed join queries, ISDN bandwidth allocation techniques, how to balance query response time and monetary cost, and the immediate assembly of join results. Section 3 gives a brief overview of the prototyping system that deploys the techniques mentioned in this paper. Finally, in Section 4, we give the conclusion of our work.

2. Query Distribution Methodology

The key point of the query distribution is to find efficient ways to evaluate queries with either reduced communication costs or query response time. It has been known as a class of hard problems, for which no efficient algorithms have yet been found. Nevertheless, efficient algorithms have been proposed for special cases of the problem. Usually, these algorithms solved only very restrictive versions of the general problem. To tackle the general problem, the ideas behind those special case algorithms were adapted as heuristics to find close-to-optimal, or at least good solutions. In this section, an algorithm will be introduced, which is optimal (in respect to certain restrictions on the search space) for a special class of queries called chain queries. The algorithm can be applied to general queries as heuristic to find efficient, if not optimal, query execution plans.

For distributed join queries that involve multiple database sites, the search space of the optimization is defined by many factors, including the join strategies (full versus semi-join), the order in which the tables are drawn into the join tree, the ISDN channel allocation strategies, the number of database sites involved, and etc. As will be shown in this section, the complexity of an exhaustive search is significantly increased due to the new factor of ISDN management.

2.1 Basic Model

The distributed database system can be modeled as a directed graph, denoted *G(V,E)*, which contains of a collection of nodes *V* and a set of edges *E*. Each node in the graph represents a site. An edge *<v, w>*, which connect node *v* to node *w*, indicates that a direct ISDN connection can be established from site *v* to site *w*. The cost imposed on an ISDN call can be expressed by a formula which is a "step" function of the duration of the call. The cost typically consists of a fixed charge for call setup (including the first time unit), and a variable component that is charged against additional time units based on a fix rate. The rate can vary depending on the time when the call is initiated. Thus, for each edge <v, w> in the graph, we associate it with a cost formula:

$$\text{comm-cost} = \begin{cases} SC(t) & \text{if } \Delta t < FTU \\ SC(t) + UC(t) * \perp[(\Delta t - FTU)\, / TU\,] & \text{otherwise} \end{cases}$$

where:

t:	time when the call is made;
Δt:	duration of the call;
SC(t):	fixed cost charged for call setup (including first time unit) at time t;
FTU:	length of first time unit;
UC(t):	cost rate per additional time unit;
TU:	length of an additional time unit;
$\perp[x]$:	lowest integer $\geq x$.

The above information defines the static parameters of the system. To best utilize the ISDN network, dynamic information regarding the ISDN resource allocation must be maintained. Each mode will keep track of the information of the usage of its ISDN lines, for example which channels are currently busy, idle, or available, and to which destinations those established calls are connected. Sometimes, there is a portion of the currently billed time unit remaining after a data transfer completes. It is always beneficial to keep the connection open until this time unit expires, because other queries might be able to utilize the idle connection to transfer data without additional charge. Even if the new transfer extends into another time interval, the setup cost, which is typically higher than the rate for additional time units, can be avoided. Consequently, for those idle connections, we also need to keep track of the surplus time before the last charged time unit expires.

2.2 The search space for join queries

Join queries are queries with join operation involved. As a starting point, we consider only "chain" queries. A chain query involves N tables, each on a distinct database site, that is defined as R1 Join R2 ... Join RN. We define the search space for a chain with the following assumptions:

- The query is evaluated as a sequence of *binary joins*, and only linear join trees are considered. A linear join tree (or a *left-deep first tree*, as it may be referred to in some of the literature) has the restriction that, for each binary join in the tree, at least one operand must be a base relation (i.e. one of R1,..., RN). This restriction limits the otherwise explosive space of join trees to search.
- For each binary join, both *full join* and *semi-join* methods are considered. Either site can be the *data assembling* site (i.e. the site at which the result of the join will be placed). Note that the decision of data assembling site implies the data transfer direction.

- To send a table from one site to another, a shortest path must be used. The shortest path would incur the lowest ISDN communication cost among all possible paths. Note that the least-cost path between two sites is not necessarily the direct connection.

An exhaustive search would explore all the alternatives within the search space. To begin, a relation R_k must first be selected, then an adjacent relation is drawn to join with R_k to produce an intermediate result. The procedure continues, with each adjacent relation being added into the intermediate result one at a time, until all relations are included. Thus, it is not hard to derive that the number of possible join orders, denoted *J(N)*, is:

$$J(N) = \Sigma_{k=1 \text{ to } N} (N-1)!/(k-1)!\,(N-k)!$$

For each binary join, there are two join methods (full join and semi-join), this amounts to 2^{N-1} combinations of join methods for each fixed join order. In addition, the least-cost path search algorithm will contribute to each data transfer a complexity of $O(|V|^2)$, where $|V|$ is the number of sites in the network. Taking all together, the complexity of an exhaustive search algorithm would be in the order of $J(N) * 2^{N-1} * |V|^2$.

Using an exhaustive algorithm that searches the entire space of alternative query execution plans is time-consuming, and in many cases prohibitive because the complexity grows quicker than exponential as the number of tables increases. Pruning techniques can be used to reduce the search spaces for both join order and shortest paths. Below, a kind of pruning technique for obtaining efficient join order is presented.

2.3 Pruning techniques to reduce the search space

The technique we used to prune the join order space is called *dynamic programming* – a widely known and useful technique in reducing combinatorial search space. The algorithm explores the space of linear processing trees by stages. At stage *k*, the best order to join *k* adjacent tables, along with the join methods for each binary join in the partial tree, is found. The determination of the optimal join order at stage *k*, however, depends only on the optimal join orders found at stage *k-1* (for any *k-1* adjacent tables). It is this specific feature of dynamic programming that prunes most of the inferior candidates off the search space. We will explain the principle and analyze the complexity of the dynamic programming algorithm next.

Let *R(I,j,m)*, where $1 \le I \le m \le j \le N$, denote the intermediate result of joining *Ri, Ri+1,..., Rj,* with the result being placed on site m. Let *P(I,j,m)* be the optimal join execution plan (including the join order and the join method for each binary join) for *R(I,j,m).* It can be proved that *P(I,j,m)* can be obtained by selecting the join plan with the lowest cost from among the following join plans:

- If I < m < j, consider among: P(I,j-1,m) Join Rj, and Ri Join P(I+1, j, m)
- If I = m, consider among: P(I,j-1,m) Join Rj, and Ri Join P(idq +1, j, m'), for all I < m' ≤ j
- If j = m, consider among: Ri Join P(I+1, j, m), and P(I, j-1, m') Join Rj, for all $1 \le m' < j$

Note that for each join operation considered above, the costs for both full join and semi-join will be estimated, and the one with the lower cost will be selected as the join method for the join operation. The method, as mentioned earlier, is indeed implemented in an iterative fashion. At stage *k*, it computes *R(I,I+k,m)* for all values of *I* and *m* such that $1 \le I \le m \le I+k < N$.

By fixing *k* and *I*, it is not hard to see that there are *O(k)* candidates plans for *P(I,I+k,m)* (there are indeed $4(k-1.) + 2(k+1) + 2(k+1) = 8k$ options, considering both full and semi-join). Therefore, the complexity of the dynamic programming search, excluding the cost of the shortest path algorithm, is:

$$\sum_{k=1}^{n} \sum_{i=1}^{n-k} 8k = 4/3 * (N^3 - N) = O(N^3)$$

If the shortest path algorithm is invoked for each data transfer, then the final complexity becomes $O(N^3 * |V|^2)$. This can also be very expensive if both *n* and *|V|* are large. However, it is much better than before. A more efficient way could be to find the shortest paths between every two sites involved in the query at the very beginning of the algorithm. This can be done in a time complexity of $O(N * |V|^2)$ by modifying the well-know all-pair shortest paths algorithm. The final complexity would then be reduced to $O(N * |V|^2 + N^3) = O(N * |V|^2)$.

2.4 ISDN bandwidth allocation techniques

The ISDN is different from the traditional networks with its dynamic properties. Therefor, several strategies are used to take advantages of ISDN's tariff structure. These include the following:

- Intelligent connect/disconnect strategies: For long-distant ISDN connections, the charge is based on usage increments, regardless of whether one uses up all the bandwidth of the entire increment or not. Thus, the connection must be kept until the current increment expires so that the remaining increment can be used for another possible data transfer without additional cost.
- Dynamic channel allocation: Weighing between query response time and ISDN monetary cost, tech-

niques are developed to determine how many channels should be allocated to achieve a maximum weighted cost. More details are provided in the next section.

- Least-cost data transfer paths: To minimize the ISDN monetary cost incurred, the least-cost path must be found for each data transfer. We have modified the conventional shortest path algorithm to take into account both ISDN cost and transfer time. Our algorithm not only determines the path via which the data should be routed; it also determines how many channels should be allocated for each direct edge along the path.

2.5 Balance the query response time and monetary cost

Ideally, one would expect to obtain the best performance at the lowest cost. Unfortunately, reducing query response time usually means paying more for bandwidth aggregation. In most cases, minimum communication cost (defined as the minimum cost needed to execute a query) and minimum response time (defined as the minimum time to execute a query) can not be achieved at the same time. An execution plan that bears the minimum communication cost may take, for example, ten times longer to execute the query than does the one with the minimum response time. To balance between communication cost and response time, we use a weighted cost as the objective function for minimization.

The weighted cost function we use bears the following form: $W_{comm} * comm\_cost + W_{resp} * \alpha * resp\_time$, where *comm_cost* (in dollars) and *resp_time* (in minutes) refer, respectively, to the estimated ISDN communication cost and response time incurred by an operation. α is an adjusting constant that makes response time comparable with communication cost. The value of α depends on how much the user thinks his time is worth. For example, if the user is getting paid $30 an hour, then he can assign *AJST_CONS* to be 30/60 = 0.5 dollar/min. This way, the response time of a query can be converted into a unit compatible with the communication cost. For instance, a two-minute query would cost, in addition to the ISDN cost, an equivalent of 0.5*2 = 1.0 dollar to the user. W_{comm} and W_{resp} are the weights for communication cost and response time respectively. Their values are to be determined by the user based on the relative importance of communication cost to response time. In our algorithm, we enforced a constraint that requires the sum of the two weights to be 1.0, i.e., $W_{comm} + W_{resp} = 1.0$.

It is easy to see that setting $W_{comm} = 1.0$ would instruct the algorithm to find the execution plan with the minimum communication cost, while setting $W_{resp} = 1.0$ instructs the algorithm to find the execution plan with the minimum response time at any cost. Setting W_{comm} anywhere between 0 and 1 would instruct the algorithm to find the execution plan of the least weighted cost.

We have devised an optimization algorithm based on the dynamic programming technique stated in section 2.2 and 2.3 that aims at minimizing the weighted cost. The shortest path algorithm was also modified to consider the weighted cost. The modification, however, is non-trivial as one of the cost components – response time – is not *additive*. The reason is that a node may start processing the incoming data as soon as the first byte arrives and sending the output to the next node, affecting both ISDN cost and query response time.

2.6 Immediate assembly of join results

When constructing the execution plans, the algorithm requires the immediate results of each pure or semi- join, which are comprised of combined tuples from both operand tables, to be placed at one of the operand sites. This eliminates the options of restricting the tables first (without producing the combined tuples) and deferring the assembly of the final query result until the very last stage. For example, to evaluate R_1 *join* R_2 *join* R_3 with result to be stored at destination site 4, a possible deferred assembly strategy may do the following:

(1) Restrict R_1 by join attributes of R_2
(2) Restrict R_2 by join attributes of R_1 and R_3
(3) Restrict R_3 by join attributes of R_2
(4) Send the three restricted tables to site 4, and perform a regular join

In the above plan, step (1), (2), and (3) can be processed concurrently. The result is assembled in the final step at the destination site. The advantage is that we may be able to avoid repetitive transfer of projected attributes over the network. It is, however, not always superior to a linear search tree. For example, if all of the table sites are within a local call area on the east coast, the destination site is on the west coast, and the size of the final result is small, then it costs less to join the tables within the local call area and send the final result to the remote destination than sending restricted tables (which could be larger than the final result) to the destination site.

Due to the latency of query compilation and optimization, the weighted graph that is used by the algorithm to find the shortest paths could have changed by the time the query is presented for execution. For example, a connected line might have been broken, or an available channel might have been preempted. Because the time to optimize a query (ranging from hundreds of milliseconds to a few seconds for less than five join tables) is usually much less than the smallest ISDN charge time unit (usually one minute), this is unlikely. If it ever happens that the shortest paths planned by the algorithm can not be set up at query execution time, then we can rerun the shortest path algorithm on the updated graph to find an alternate path.

3. Prototyping System

Based on the methodology introduced before, we have implemented a prototyping system, which deployed all the aforementioned ideas and works well over ISDN. In this section, a brief introduction of our prototype system is given.

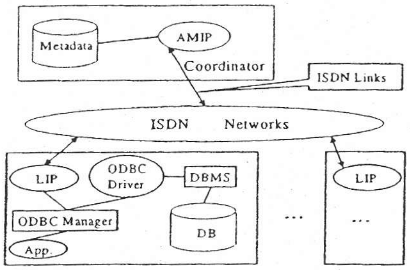

Figure 3.1. System Architecture of the Prototype System.

Figure 3.1 illustrates the basic architecture of our loosely coupled multi-database environment. Each site in the figure runs a DBMS that operates autonomously, and among them, one is designated as the coordinator whose responsibility is to receive inter-database query requests from the other sites and generate the corresponding global execution plan. There are two different kinds of software components in the system. They are the Application Manager Interface Program (AMIP) that runs on the coordinator site and the Local Interface Program (LIP) that runs on the remaining sites. We use ODBC to resolve the potential heterogeneity among the different kinds of DBMSs. The LIP is a kind of specific ODBC driver. The application program first submits ODBC compliant SQL queries to the LIP by means of the ODBC manager. If the queries are only related to the local database, the LIP will select another ODBC driver that is provided by the local DBMS through the ODBC manager to access the local database. If the queries need to access one or more remote databases, the LIP will send the queries to the AMIP via the ISDN network. The AMPI parses and decomposes the received queries into a global execution plan that is composed of a set of sub-queries. The AMPI coordinates the processing of each sub-query by opening ISDN connections among the involved local sites. To reduce ISDN costs, queries and control messages are communicated through D channels while the query results are routed through B channels.

The major components of the AMIP are shown in Figure 3.2. The communication module is responsible for controlling the operations of the corresponding ISDN hardware so as to send and receive necessary information, including messages, intermediate results, query requests, etc, between the AMIP and certain LIPs via the ISDN network. Once a message reaches the AMIP from an LIP site, the message dispatcher interprets it and forwards it to the corresponding module for further processing. If the message consists of SQL query statements, the SQL parser will be invoked to parse and validate the query. If the incoming message concerns query coordination, it will be passed to the coordinator module. The query distribution algorithm in the query optimization module produces a global execution plan for each query request, and the execution plan will subsequently be passed to the coordinator for further executions among the involved LIP sites.

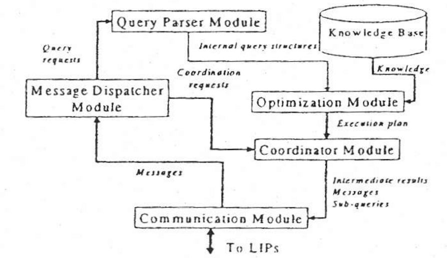

Figure 3.2. System Architecture of AMIP

There is a knowledge base running in the AMIP that contains a collection of essential knowledge, including global schemata and mapping information, to resolve the possible heterogeneity [13], determining schematic and data conflicts between the global schema and local schemata [14]. Usually, such conflicts occur when semantically equivalent data is represented in different ways. In our prototyping system we mainly focus on the four major types of conflicts that are most often cited in the literature. They are:

- *Structural heterogeneity.* Logical data structures (i.e. the number of relations and the foreign key joins between the relations) may vary across databases that contain similar data. This is due to different preferences on how data should be organized and due to variations in database contents.
- *Abstraction heterogeneity.* This class of heterogeneity arises when two databases use different levels of generalization and aggregation and retain different levels of information detail.
- *Naming heterogeneity.* Naming variations can appear in two levels: the relation level and attribute level. Differences in the names of two similar relations or attributes will make them appear different to the database system, thus requiring semantic reconciliation.
- *Domain heterogeneity.* Even for same-named attributes, the underlying domains may be different.

Figure 3.3 presents the architecture of the LIP. The LIP accepts query and sub-query requests from both local users and the AMIP. It accesses the local database through the ODBC driver provided by the corresponding database vendor. The communication module deals with all the communication affairs between the LIP and the other software components. It receives messages as well as queries from either the AMIP or local application programs, and also receives intermediate row sets from the AMIP for the involved inter-database queries. As shown in Figure 4, all incoming information is considered to be a message. The message dispatcher module is able to identify its contents, and therefore dispatch intermediate row sets and sub-queries to the two corresponding modules. Data access modules retrieve data from local databases and send results to the query-processing module, which performs inter-database joins.

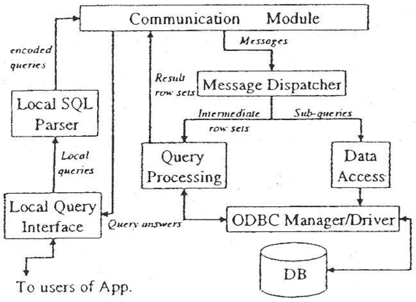

Figure 3.3. System Architecture of LIP.

4. Conclusion

In this paper, a new database integration methodology over intelligent networks has been presented. The contributions of our work in this paper include:

- Principles of optimization process for the ISDN-enabled distributed environment.
- Defining the search space of the optimization and the complexity of an exhaustive search.
- Pruning techniques to avoid exhaustive search of all execution plans.
- Query optimization heuristics to minimize weighted cost by balancing query response time and monetary cost over ISDN.
- An ISDN-enabled distributed database prototype.

Future research direction will include the database integration over hybrid networks, which include Internet, Intranet, and intelligent networks.

Reference:

[1] B. E. Reddy, et al, "A methodology for integration of heterogeneous database." IEEE Transactions on Knowledge and Data Engineering, Vol.6 No.6, pp. 920-933, 1994.

[2] Amit Sheth and Vipul Kashyap. "So far (schematically) yet so near (semantically)." Interoperable Database Systems, pp. 283--312, Elsevier Science Publishers B. V. (North-Holland) 1993.

[3] Amit P. Sheth and James A. Larson. "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Database." ACM Computer Surveys, vol. 22 No. 3, pp.183-235, 1990.

[4] Alon Y. Levy, et al, "Querying Heterogeneous Information Sources Using Source Descriptions" In Proceedings of the 22nd VLDB Conference, pp.251-262, 1996.

[5] Witold Litwin and Abdelaziz Abdellatif. "Multidatabase interoperability." *IEEE Computer*, pp.10-18, 1986

[6] Witold Litwin and Abdelaziz Abdellatif. "An overview of the multi-database manipulation language MDSL." *Proceedings of the IEEE*, 75(5), pp. 621-632, 1987.

[7] Rafi Ahmed et al. "The Pegasus heterogeneous multidatabase system." *IEEE Computer*, Vol.24 No.12, pp.19-27, December 1991

[8] Christine Collet, Michael N. Huhns, and Wei-Min Shen. "Resource integration using a large knowledge base in Carnot." *IEEE Computer*, pp. 55--62, 1991.

[9] Yigal Arens, Chin Y. Chee, Chun-Nan Hsu, and Craig A. Knoblock. "Retrieving and integrating data from multiple information sources." International Journal on Intelligent and Cooperative Information Systems, Vol.2 No.2, 1993.

[10] H. Garcia-Molina et al. "The TSIMMIS approach to mediation: data models and languages" In *Next Generation Information Technologies and Systems.*

[11] Y. Papakonstantinou, S. Abiteboul, and H. Garcia-Molina. "Object fusion in mediator systems" In *Proceedings of the 22nd VLDB Conference*, 1996.

[12] N. Rishe, "Managing network resources for efficient, reliable information systems", Proceedings of the Third International Conference on Parallel and Distributed Information Systems, pp.223-226, 1994.

[13] W. Kim, et al, "On Resolving Schematic Heterogeneity in Multidatabase System", Distributed and Parallel Databases, Vol.1 No. 1, pp.251-279, 1993.

[14] N. Rishe, R. I. Athauda, J. Yuan and S. Chen, "Semantic Relations: The key to integrating and query processing in heterogeneous databases". In Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, 2000, Vol 7, Computer Sciencee and Engineering: Part I, pp.717-722, 2000.

[15] N. Rishe, J. Yuan, R. Athauda , S. Chen, X. Lu, X. Ma, A. Vaschillo, A. Shaposhnikov, D. Vasilevsky, "Semantic Access: Semantic Interface for Querying Databases". To appear in the Proceedings of 26th International Conference on Very Large Databases (VLDB), Cairo, Egypt, Sept. 2000.

[16] N. Rishe, J. Yuan, R. Athauda , X. Lu, X. Ma., "SemWrap: A Semantic Wrapper over Relational Databases, with Substantial Size Reduction of User's SQL Queries" In the Proceedings of the 7th International Conference on Extending Database Technology (EDBT), software demonstrations Track, pp. 13-14, Germany, March 2000.