

The Architecture of a Generalised Distributed Database System – PRECI*

S.M. DEEN, R. R. AMIN, G. O. OFORI-DWUMFUO AND M. C. TAYLOR

PRECI PROJECT, Department of Computing Science, University of Aberdeen, Aberdeen AB9 2UB, Scotland

A research prototype of a generalised distributed database system called PRECI is currently being developed at the University of Aberdeen in collaboration with a number of research centres, mainly in Britain. The system is fully decentralised, with both retrieval and update facilities, permitting heterogeneous and even pre-existing databases as nodes and supporting links to other (external) distributed databases. The system supports both location transparent and transaction-oriented queries to fulfil differing user requirements. Nodal autonomy, data integration, metadata and staged binding of queries may be seen as its other main features. The basic design of the system is now complete, and a partial implementation is in progress.*

1. INTRODUCTION

PRECI(Prototype of a relational canonical interface) is a database research project which was originally undertaken at Aberdeen¹ in order to provide a framework for research in all aspects of databases. To date the project has produced a research prototype called PRECI/H which is a generalised DBMS, based on a canonical data model supporting relational, network and other data models as user views. It uses the ANSI/SPARC architecture, its 'conceptual schema' (called canonical schema) being written in a relational form. The principal data manipulation language used includes an extended relational algebra called PRECI Algebraic Language (PAL), which supports all the traditional relational commands (including functions) and a number of special commands for data integration. The result of any of these commands is another relation, thus all operations have the closure property.

PRECI* (pronounced presi-star) is a research prototype within the PRECI project for a generalised distributed database management system (DDBMS), which is being developed in collaboration with a number of research centres. Its principal architectural characteristics are:

- (1) A decentralised control system with heterogeneous databases (including pre-existing databases) as nodes.
- (2) Both inner and outer nodes to suit differing user requirements (see later).
- (3) Retrieval and update facilities for global users, with full location transparency for the inner nodes.
- (4) Ability to allow new nodes to join.
- (5) Ability to link with other distributed databases (external DDBs) at peer level.
- (6) Maximal nodal autonomy as described below:
 - (i) A nodal database retains its nodal users, independent of the distributed database management system (DDBMS).
 - (ii) A database can join the DDB as a node by contributing only a logical subset of its data, the subset containing none to all of the data.
 - (iii) Each nodal database retains full control of itself, with means to withdraw itself or its data. It can specify who can update and retrieve its data, and under what conditions.

- (iv) Data contributed by a nodal database (home node for these data) to a DDB can be replicated and distributed by the DDBMS to other nodes (foreign nodes for those replicated data), if so permitted by the home node. This facility applies only to the inner nodes.
- (v) A database may participate as a node in several DDBs and can itself be a DDB.

These characteristics have been discussed initially in reference 14, and taken together they distinguish PRECI* from the current prototype implementations²⁻¹³. It should be noted that we have used the term node to imply a logical site, one for each nodal database and its associated global module.

The architecture of PRECI* is meant to support the characteristics listed earlier. In Section 2 we present a five-level schema architecture, followed by a discussion of nodal and global control systems in Section 3. For a clearer understanding of the architecture and its impact, we have outlined the elements of the global query processor in Section 4. A conclusion is given in Section 5.

2. ARCHITECTURAL LAYERS

We define a distributed database as a database representing a logical collection of data from other inter-linked databases. It is possible for the same computer to support several databases, and hence several nodes. Therefore the nodes of a DDB may not necessarily be connected via data communications links. We categorise DDBMS broadly into two classes: closed and open. A closed DDBMS permits only a purpose-built DDB where all nodal DBs are designed to suit the requirements of the DDB. Typically this could be a homogeneous system with the DDBMS having the final control over all data distribution. In contrast, an open DDBMS allows pre-existing databases, ideally of any data model, to join the DDB – potentially at any time. Thus the interfaces provided by an open DDBMS are open to all. An open DDB is typically a confederation of nodes, each node retaining full control over its data. PRECI* is intended as an open DDBMS.

The architectural framework used in PRECI* can be viewed as an extension of the ANSI/SPARC model by

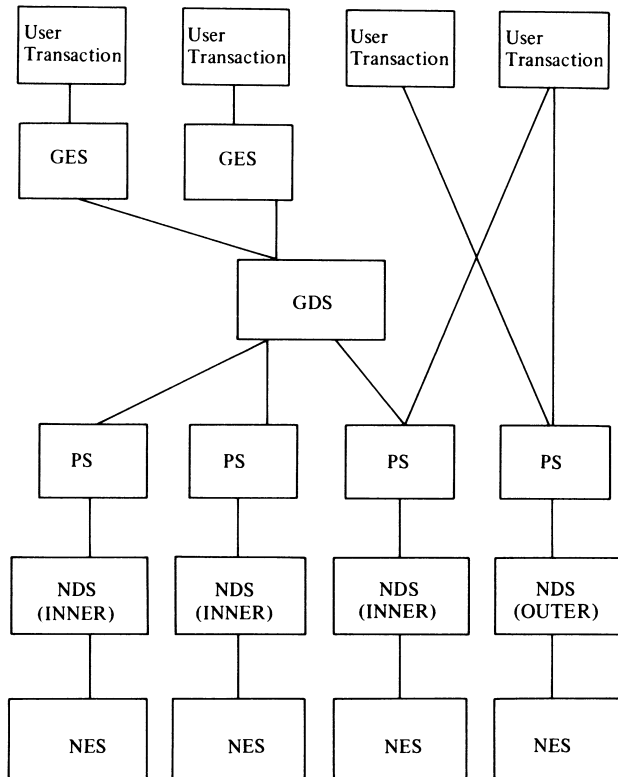


Figure 1. PRECI* schema levels

two additional layers: the global conceptual schema referred to as global database schema (GDS) and the global external schema (GES)(figure 1).

Although a nodal database can be a DDB, we shall generally assume a node to be a leaf node supporting a leaf database (ie a non-distributed data base) except where indicated otherwise. A nodal database in PRECI* is fully autonomous, with its independent nodal DBMS (NDBMS) and nodal external schemata (NESs). It must however provide a relational, preferably a PAL interface to the DDB which uses PAL as the standard language for communications. The DDB may be assumed to be a nodal user via a relational external schema to the nodal database, called the participation schema (see later). A node can participate in a PRECI* DDB in one of two ways:

- (i) as an inner node : which contributes to the GDS
- (ii) as an outer node : which does not contribute to the GDS

We describe them more fully below.

2.1 Inner Nodes

In general, the inner nodes provide the best available service to their global users (inner users), through the global database schema and global external schemata. The special facilities not available to the outer users (the users of the outer nodes) are:

- Integrated data
- Meta data
- Location transparency
- Replicated data (for faster queries)
- Easier query formulation

Global Database Schema (GDS)

The GDS is formed by the participation schemata (PSs) of the inner nodes, each relation in the GDS retaining the identity of its home nodes by means of a logical node name. The presence of this logical node name is not intended to curtail location transparency. It rather enables the user to find the node name of a relation, should he require it. However the user does not have to specify node names in a query unless he wishes it (see also GES). We have chosen a relational representation mainly because of its simplicity and decomposability¹⁵.

The GDS supports integration data and meta data, which are assumed to be stored in a subsidiary database (discussed later). The need for data integration arises chiefly from the different practices at different nodes for expressing the same information. For instance one nodal database may represent distance in miles, and another one in kilometres. Likewise one database may store the price and sales tax of an item separately, while another one might store only the sum of the two as the total price. To take another example, one node may represent three exam marks in a relation R as R(STUDENT EXAM1 EXAM2 EXAM3) while another may use a ternary relation R(STUDENT EXAMNO MARKS) for the same information. The GDS itself does not provide an integrated view, say by converting all distances to miles. Instead we represent the total information at the GDS level without any loss, but recording additionally as integration data, the necessary units, conversion factors and other relevant information. The desired integration can thus be carried out at the global external schema as needed, eg permitting one GES to express distance in miles and another in kilometres. Therefore our GES and not GDS is closer to the global schema of Multibase³.

Figure 2 shows a sample global database schema, where we have deliberately chosen the university names to be absent in node 1 and node 2, thus making communications with node 3 difficult. This difficulty is removed by creating a new relation UNODE as integration data. EID means entity identifier or primary key, which can be composite. The non-EID attributes are identified as ATT. If an attribute is a foreign key, then its type is FKEY qualified by the name of the relation where this attribute is the EID. Unit gives the unit in which a numerical value is expressed, eg £ for pounds sterling.

The meta data described in the GDS represents optional information typically held in a data dictionary. This facility can be used to declare semantic groupings of selected relations, attributes, domains and so on. Thus in figure 2, we have declared a meta relation R (relationnames, attributenames). The user can access these meta relations like any other relations except that their update is restricted.

There is a global mapping schema (GMS) associated with the global database schema. It has two functions:

- (i) to distribute (ie replicate) permitted relations to other nodes (foreign nodes).
- (ii) to associate node names with their participation schema names (including those for subsidiary databases).

ITEM	NAME	TYPE	UNIT
NODE	NODE1{UNIVERSITY}		
REL	DEPT		
EID	DNO	INTE 4	
ATT	DNAME	CHAR 20	
REL	TEACHER		
EID	TDN	FKEY DEPT	/*TDN is teacher's department
EID	TNO	CHAR 4	number*/
ATT	SAL	INTE 5	£
NODE	NODE2{UNIVERSITY-B}		
REL	DEPT		
EID	DNO	INTE 5	
ATT	DNAME	CHAR 25	
REL	TUTOR		
EID	TNO	CHAR 6	
ATT	SAL	INTE 5	\$
ATT	DNO	FKEY DEPT	
ATT	GRADE	CHAR 10	
NODE	NODE3 {STUDENT-UNIONS}		
REL	UNION		
EID	UNIVERSITY	CHAR 20	
ATT	POPULATION	INTE 5	
ATT	FEE	INTE 2	£
NODE	INT {INTEGRATION-DATA}		
REL	UNODE		
EID	UNIVERSITY	CHAR 20	
ATT	NODE	CHAR 6	
NODE	META {META DATA}		
REL	R		
EID	REL-NAME	CHAR 10	
EID	ATT-NAME	CHAR 10	

In relation TEACHER the EID is <TDN><TNO>
whereas in relation TUTOR, the EID is TNO alone.

Figure 2. Global Database Schema

Global External Schema (GES)

The global external schema can support relational and possibly other user views, provided that the data model and the language are convertible into our standard form (that is PAL). The GES with the help of its mapping entries can support additional meta data description and an integrated, location transparent view. As indicated earlier PAL is intended to provide powerful data integration commands and to be used both as a data manipulation and description language. An indication of these facilities is given in Figure 3 which is derived from

the GDS of Figure 2. The entry ATT* implies that the attribute concerned can have null values and hence cannot be used for, say calculation of an average. The Mapping Division given shows some PAL commands for data integration. A publication on PAL data integration and data manipulation facility is under preparation.

Participation Schema (PS)

The participation schema describes the nodal data along with various authorisation controls. An example is shown in Figure 4 where the global users of the NODE2

DATA DIVISION

```

REL  DEPARTMENT
EID  DNO          INTE 5
ATT  DNAME        CHAR 25

REL  TEACHER
EID  DNUM          FKEY DEPARTMENT
EID  TNO           INTE 6
ATT  SAL           INTE 6
ATT* GR            CHAR 10    /*ATT* implies that this attribute can
                               have null values*/

```

MAPPING DIVISION

```
DEPARTMENT == NODE1..DEPT ++ NODE2..DEPT
```

```
TEACHER == (NODE1..TEACHER::REP(TDN BY DNUM),  EXT (GR=NULL)
            ++(NODE2..TUTOR:: REP(DNO BY DNUM),  REP(GRADE BY GR))

```

where N..R indicates relation R of node N ;

R.A indicates attribute A of relation R ;

R::EXT(A) indicates relation R extended by new attribute A

R::REP(B BY C) indicates relation R with attribute B replaced by new attribute C

and "++" and "==" are the symbols for union and definition respectively.

Figure 3. A Sample GES and its Mapping

```

PS      NODE1-PS

REL      DEPT
REPLICATION ALLOWED TO NODE2, NODE3
RETRIEVAL  ALLOWED TO NODE2, NODE3
UPDATE NOT ALLOWED
EID      DNO          INTE 4
ATT      DNAME        CHAR 20

REL      TEACHER
REPLICATION NOT ALLOWED
RETRIEVAL ALLOWED TO NODE3
UPDATE ALLOWED TO NODE2
EID      TDN          FKEY DEPT      UPDATE NOT ALLOWED
EID      TNO          CHAR 4
ATT      SAL          INTE 5

```

Figure 4. A Participation Schema

have a general permission to update relation Teacher but not its attribute TDN. Each participation schema has a version number which is updated every time the PS is changed.

Subsidiary database

Since we allow a DDB to support pre-existing databases, it is unreasonable to expect that they will be changed to incorporate replicated data. We therefore associate with

each inner node a small database, to be called a subsidiary database (SDB), to be managed by an SDBMS (subsidiary database management system) under the control of the DDBMS at this node. The basic contents of the SDB are:

- | | |
|---------------------------------|---|
| (1) Integration data of the GDS | } could be the same in all inner nodes, subject to nodal authorisation. |
| (2) Metadata of the GDS | |

- | | |
|----------------------------|---|
| (3) Metadata of the
GES | } vary from one inner
node to another. |
| (4) replicated data | |

The SDBMS is expected to support a fuller set of PAL commands, with the ability to process external data (see later).

The database schema of the subsidiary database is assumed to act as the participation schema as well, there being no separate participation schema defined for it.

2.2 Outer Nodes

If the number of nodes in a DDB is large, say in the tune of several hundreds, and if the expected usage frequency of the DDB is low, so that a user accesses only a few of those hundreds of databases at any one time, then the overhead of the creation and maintenance of the GDS and GES could be too high. What we require, therefore is a transaction-oriented facility, which permits the user to formulate his query through a suitable language for specific nodes. He could also be allowed, if he needs, to navigate through the nodes until he finds his node of interest. This we may refer to as the seek and search approach.

The need for such a facility without any elaborate GDS and GES has been re-emphasised in a recent EEC feasibility study for a distributed facility to seek and search medical records in the community¹⁷. Typically a doctor in Aberdeen might need to access a medical database in Capri. The doctor might know about his need to access the Capri database, from another database or from an exogeneous source such as the patient. In this case the location transparency is not relevant, the doctor would only want a simple capability to send his query to Capri. We provide this capability within the PRECI* architecture by what we call outer nodes (cf. reference 6).

Our outer nodes do not contribute to the GDS, but they must provide participation schemata in a relational form like any other node; however there is no structural difference between the participation schemata of inner and outer nodes. The outer user can access several participation schemata directly and address queries to one or more of them. The outer user does not normally get meta-data, data integration or replicated data, although he can create a partially integrated view by PAL commands.

Note that, although the outer nodes do not contribute to the GDS, the DDBMS has to maintain the necessary control information on them. Subsidiary databases are optional for outer nodes, but can be used to provide additional PAL commands (see below).

3. NODAL AND GLOBAL CONTROLS

3.1 Nodal Controls

As mentioned earlier, each node is independent and is expected to have its nodal users independent of the DDBMS. By insisting on a node to provide a relational view, we are localising the problem of heterogeneous mappings to the sphere of the nodes but have not resolved it. Some of the issues of such mappings have been discussed in reference 14, and are currently being studied

by us. To minimise the changes in a node we basically require a node to support the following minimal set of operations:

Selection
Join
Division
Projection

either in a relational algebra, or in a calculus, but preferably in PAL. It is recognised that an NDBMS may not be able to process external data, that is, relations not stored in this database, but sent to it over the communications lines for processing. The capability to process such external data, and other PAL commands, is provided by the local SDBMS. However, due to machine restrictions or other limitations, a given SDBMS might not be able to handle all PAL commands. In that event, the DDBMS of this node would despatch the relevant data and the query to the SDBMS of another node which can process the required command.

If a node withdraws from a DDB, then the relevant participation schemata are made inoperative with a null version number. Each node has essentially a bilateral arrangement with other nodes, thus stipulating in the participation schema the conditions under which the data can be accessed. Since the participation schemata are controlled by the node, and not by the DDB, the latter cannot violate the authorisation stipulation of the former. It is recognised that a node may change its data. Any such change should cause the NDBMS to alter the version number of the relevant participation schema.

3.2 Global Control

The decentralised control system of PRECI* is shown in Figure 5, except that the outer nodes do not necessarily have any subsidiary databases. The DDBMS together with SDBMS and SDB, if present, constitutes what we call the global module. An initial description of the functions of a global module is given in reference 14.

PRECI* also supports interactions with other distributed databases. This is done in two ways:

- (i) a subordinate node: in this case the DDB in question acts as a PRECI* node (either inner or outer) complete with a participation schema and a global module. Thus this DDB becomes an internal DDB.
- (ii) an external DDB: in this case the other DDB, called an external DDB, behaves like an outer node but without having the global module of the PRECI*. However an external DDB can provide to PRECI* more than one PS, each being treated as if it were an outer node. Conversely PRECI* can provide suitable external schemata to the other DDB. All commands and data between PRECI* and the external DDB are converted into an intermediate standard form via what is called an external protocol (see also later).

4. TRANSACTION PROCESSOR

All global transactions are compiled and executed in three stages, the compiled version normally being retained for subsequent execution. In order to ensure the integrity of

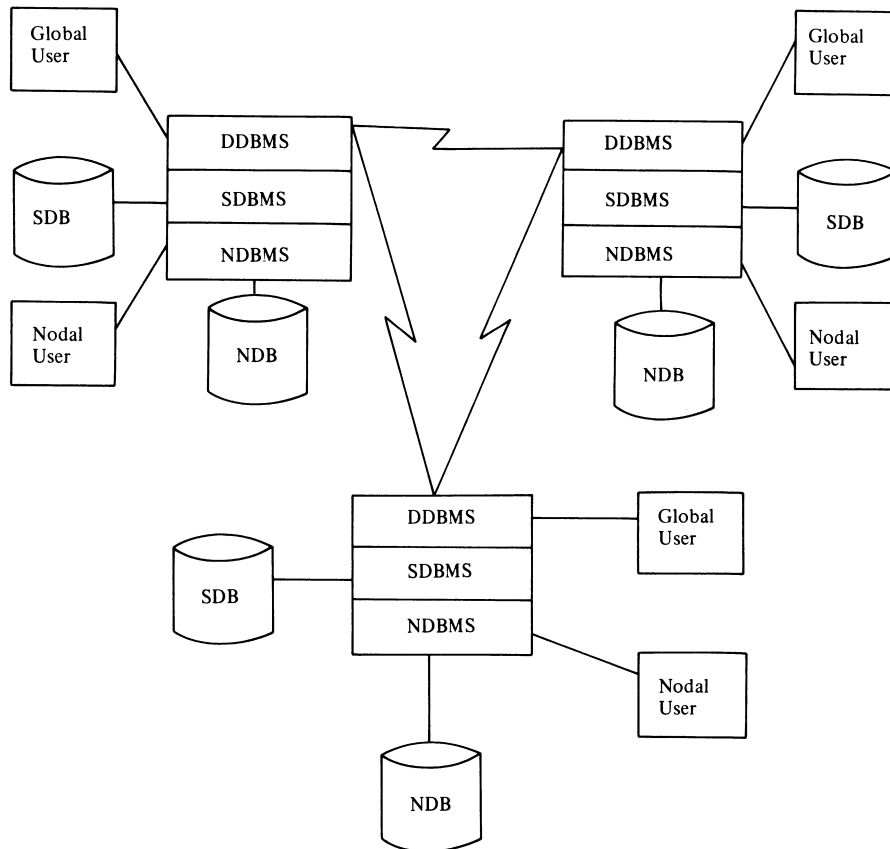


Figure 5. Decentralised Controls with subsidiary database

the DDB, global update is permitted only on base relations. It is assumed that a participation schema will not grant update permission on a relation unless it is a base relation. The queries however can be more complex and operate on any set of authorised relations or their derived views. The basic query processing steps are sketched below:

4.1 Global Query Preprocessor (GQP)

Each user query at the originating node (ie the requester) is validated by the GQP against the GES in the case of inner nodes (inner queries) and against the PS in case of outer nodes (outer queries), along with appropriate authorisation checks. The query is then resolved, where appropriate, into a set of intermediate queries, one or more such queries for a given node (execution node). The intermediate queries are further subdivided, eventually into subqueries, by an optimiser, taking into account:

- (i) replicated data where available and relevant
- (ii) availability of operations. For instance if an intermediate query has an operation that cannot be performed by either the NDBMS or SDBMS of the designated execution node, then the subquery would be divided into secondary subqueries which can be performed at the designated node. The results of these secondary subqueries would be sent to another execution node to evaluate the original subquery.
- (iii) communications cost

The original global query is now transformed into a tree,

made up of subtrees, a subtree representing a global subquery, with one or more subqueries for each selected execution node. As the first stage of compilation the GQP produces a global query plan (Q-plan) containing:

- (1) the tree
- (2) addresses of the execution and destination nodes (where the result should be sent after the execution of a query)
- (3) version numbers of the GES, GDS and PS
- (4) input/output data structures and parameters and, where relevant, a list of actions to be invoked on specified execution conditions.

Note that all inner queries are mapped against the PS entries via the compiled version of the GES and GDS, whereas all outer queries and external queries are mapped directly against the relevant PS.

From the Q-plan, the GQP prepares subquery execution plans (S-plans) one for each subquery. S-plans contain:

- (1) the subtree
- (2) pre-execution instructions
- (3) post-execution instructions (ie what to do with the result)
- (4) relevant input/output data structures and other parameters (such as version numbers of the involved schemata)
- (5) a list of actions to be invoked on specified execution conditions.

S-plans form part of PRECI* protocols, of which there are two:

- (i) internal protocol for communicating with PRECI* nodes, and hence understood by all PRECI* DDBMS.
- (ii) external protocol for communicating with external DDBs. Such protocols are currently being studied in collaboration with several European research centres under EEC grants.

4.2 Global Subquery Preprocessor (GSP)

The principal task of the GSP is to transform a subquery in the S-plan into one or more nodal queries and submit them to the next stage for compilation. There are two reasons why a subquery may require splitting/modifications:

- (i) If an operation of the subquery can be performed only by the SDBMS, but not by the NDBMS.
- (ii) If the NDBMS does not support a PAL interface, in which case the subquery has to be transformed into the relational language supported at that node. This part of the GSP will vary from one node to another.

The GSP eventually generates a nodal query plan (N-plan) which includes compilation/execution instructions.

4.3 Nodal Query Preprocessor (NQP)

NQP is a part of the NDBMS, beyond the control of the DDBMS and hence implementation-dependent. We assume here only its conceptual existence as part of the nodal query processor which must compile/translate each nodal query. (The NQP should not distinguish between a query from a global user and one from a nodal user.) Similar compilation also takes place in the SDBMS, but under the control of the global module.

After a successful compilation, the Q-plan, S-plans and N-plans are suitably updated making them ready to be used for execution. We recognise that some nodal DBMSs may translate and execute a query at the same time, without retaining a compiled version for subsequent use. This requires only a trivial modification to our strategy and hence is not discussed here.

4.4 Query Execution

The execution of a query also proceeds in three stages (via a global query executor (GQE), a global subquery executor (GSE) and a nodal query executor (NQE), paralleling the preprocessing stages. At every execution, the version numbers of GES, GDS and PS are matched where relevant, rejecting the query with an appropriate message in the case of a mismatch. If the query is not affected by the new versions, it can be recompiled immediately without any change. We however, do not expect too many changes of version numbers.

This architecture permits the use of an intermediate node of the DDB to execute subqueries if so required by the optimiser. In Figure 6, the requester node A asks execution nodes B and C to forward relations R1 and R2 respectively to node D, which is required to perform

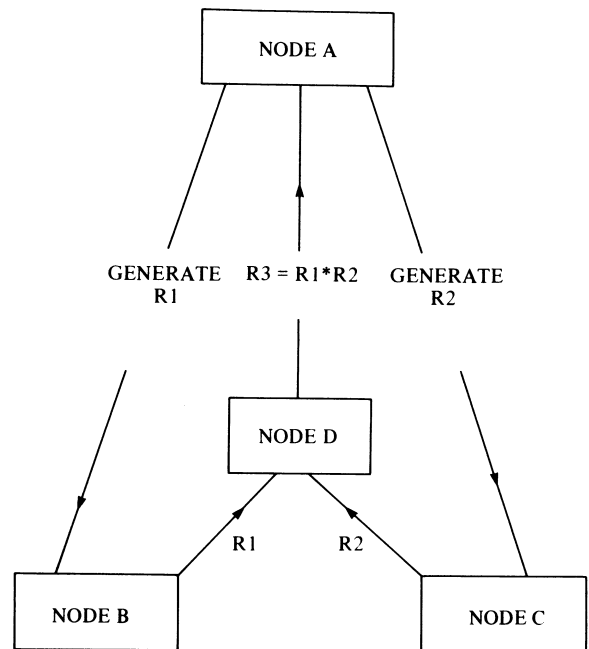


Figure 6. Use of an intermediate node D

a join on them – with the help of its SDBMS if necessary – and to transmit the result back to A.

4.5 Internodal Integrity

By internodal integrity, we mean the consistency of data among the participating nodes. There are two possible sources of inconsistency^{1a}.

- (i) Replicated data: if data is replicated over several nodes, then the database will be inconsistent if all the copies are not updated every time.
- (ii) Dispersed update-unit: if the data in the read/write set of an update transaction is dispersed over several nodes then we have a dispersed update-unit. Such a situation would occur if the employee salary details are maintained in the branches (nodes), and total salary expenditure in the department record at the head office (another node).

Consistency can be weak or strong. If weak, the database can remain in a transient inconsistent state without corrupting the data, whereas if strong, the database must be maintained in a consistent state at all times. Strong consistency must be enforced in the case of dispersed update-units, but not necessarily so in the case of replicated data.

Data is generally replicated for ease of access in cases where the frequency of retrieval exceeds that of update. It seems to us that the retrieval requests do not always need the latest version of the data, and in those cases (referred to as retrieval Mode A) a weak consistency should suffice. Thus our Mode A transactions use replicated data if it is cheaper, but all update transactions, and those retrieval transactions which require the latest data (retrieval Mode L), are directed to the home node which always maintains the latest version of data. Once an update is performed, the home node immediately broadcasts the update message to all

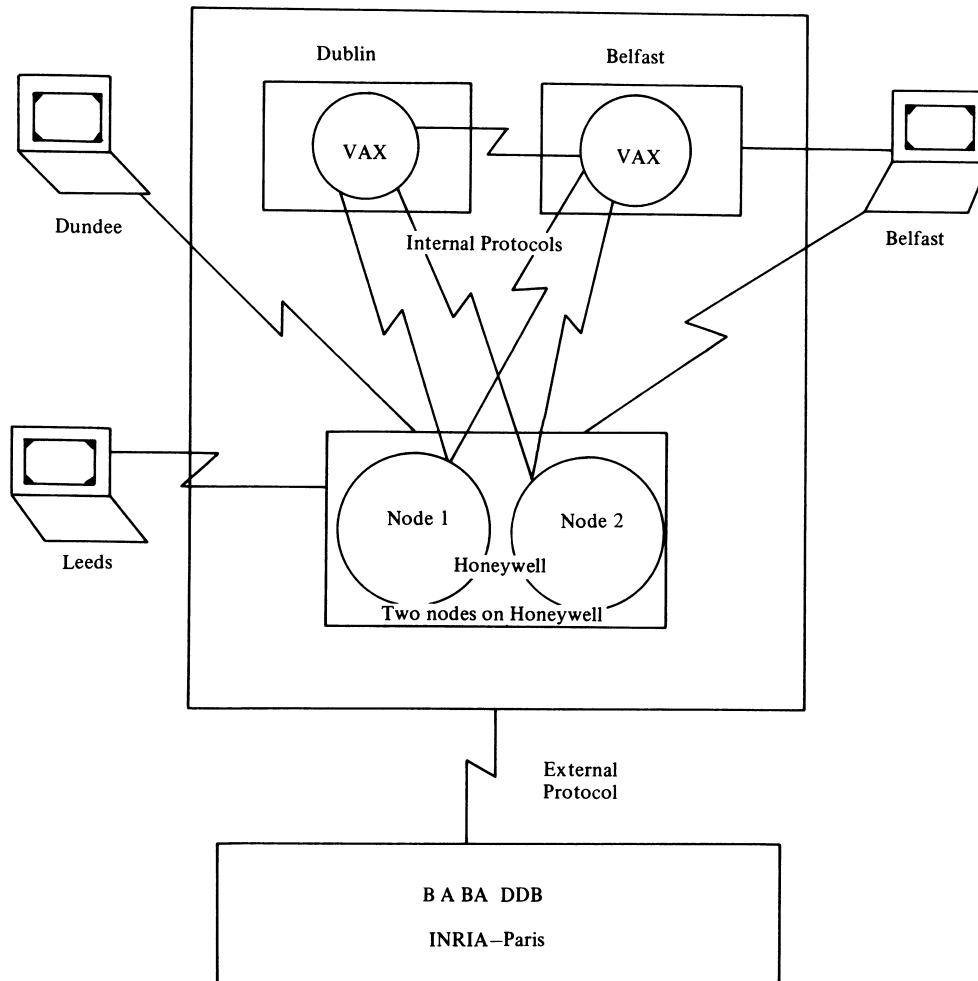


Figure 7. PRECI* Environment

relevant foreign nodes which then copy the update in their databases with the highest priority. The delay in effecting the update in the foreign nodes is expected to be very short, and hence most retrieval users should be satisfied in the transaction Mode A.

A nodal user may not be allowed to update data given to foreign nodes or those which form part of dispersed update-units. This must be enforced at the nodal level, transforming all such nodal transactions into global transactions either manually or automatically.

We assume that each node maintains a set of mail boxes, one for each node in the DDB. The mail boxes are used for holding messages sent out to those nodes but not yet acknowledged. When a node starts a session or restarts after a breakdown, it must first take two recovery actions. It reads its mail boxes at other nodes, and takes action on the messages, such as completing lost updates. Then it broadcasts all its outstanding messages to other nodes, and these messages include those which have been sent out earlier, but their acknowledgements were not received. Such retransmission of a message does not cause any update problem to replicated data, since the update version numbers of replicated data units are always checked before an update is effected. Strong consistency based on what we call a delayed two-phase commit protocol is being planned for dispersed update-units.

5. CONCLUSION

The architecture presented here is meant to capture many of the facilities of an open and generalised DDB, in particular the ability to provide location transparency and transaction-oriented queries, the former providing data integration and meta data. An associated activity is the development of a standard protocol for communications between distributed databases, mentioned in the text as external protocol. The basic design of PRECI* is now complete and we are implementing a pilot system, with only a subset of the design features described above. The pilot system is expected to include an initial version of an external protocol for linking to EEC countries over national data-communication networks. In the implementation we plan to have two nodes at Aberdeen co-existing in the same Honeywell computer, one node at Belfast in a VAX and another one in Dublin, in another VAX, with an external link to Litwin's Multidatabase project (Figure 7). Connections to our collaborators at Leeds and Edinburgh are also envisaged.

Acknowledgements

We wish to thank the British SERC, and the EEC COST 11 BIS for supporting this project by several grants. We also wish to thank all our internal and external collaborators, for suggestions, discussions and comments,

in particular to David Bell of Ulster Polytechnic, Jane Grimson of Trinity College, Dublin, W. Litwin of INRIA (France) and Peter Apers of Vrije University (Amsterdam).

We are grateful to R. Carrick, and D. Kennedy of Aberdeen University for suggesting improvements to the earlier version of this paper.

REFERENCES

- 1 (a) S.M. Deen et al, The design of a canonical database system (PRECI), *The Computer Journal*, Vol.24, No.3, (1981)
- (b) S.M. Deen et al, The run — time system of PRECI, *Proc of the Second British National Conference On Databases*, July 1982, ed. Deen & Hammersley, J. Wiley, (1982).
- 2 J.B. Rothnie et al, Introduction to a System for Distributed Databases (SDD — 1), *ACM TODS* Vol 5:1, (March 1980) p1. There are a number of other articles on SDD — 1 in the same issue of TODS.
- 3 T. Landers and R.L. Rosenberg, An overview of Multibase, *Distributed Databases*, (Proc. of the Second International Symposium on Distributed Data Bases, Berlin, 1982), ed. by H.J. Schneider, (North — Holland) p153.
- 4 R. Williams et al, R*: An overview of the architecture, RJ3325, IBM San Jose, California.
- 5 W. Litwin et al, SIRIUS systems for distributed data management, published in the *Proceedings of the Second International Symposium on Distributed Databases*, Berlin Sept 1 — 3, 1982, ed. H.J. Schneider, North — Holland (1982), p311.
- 6 Litwin et al, reference 5 includes a section on the B A BA project.
- 7 (a) K.C. Toth, et al, The ADD System: An architecture for DDBs, *Proc. of the 4th VLDB*, Berlin 1978.
- (b) K.C. Toth, et al, Query Processing Strategies in a distributed database architecture, as in reference 14, p117.
- 8 E.J. Neuhold and B. Walker, An Overview of the Architecture of the DDBs POREL, as in reference 3, p247.
- 9 R. Munz, Realisation, synchronisation and restart of update transactions in a DDBs, *Distributed Data Bases* (Proc. of the first International Symposium on DDB, Paris 1980), edited by C. Delobel and W. Litwin (North — Holland), p173.
- 10 A.J. Borr, Transaction monitoring in ENCOMPASS... *Proc. of the 7th VLDB*, Cannes 1981, p155.
- 11 R. Elmasri, et al, Notes on DDRs — an apparatus for experimental research in DDBMS, Tech Rep HR — 81 — 252, Honeywell CCSC, Bloomington, Minnesota, February 1981.
- 12 M. Adiba, et al, Polypheme —, as in reference 9, p67.
- 13 P.R. Tillman, ADDAM — the ASWE DDBMS, as in reference 3, p185.
- 14 S.M. Deen, A general framework for the architecture of distributed database systems, *Proc. of International Seminar on Distributed Data Sharing*, Amsterdam, June 1981, edited by W. Litwin and R. Van de Riet (North — Holland), p153.
- 15 E.F. Codd, Relational database: A practical foundation for productivity, *Comm. ACM*, Vol.25, No.2, Feb 1982 (ACM Turing Award Lecture).
- 16 ISO/TC97/SC5/WG3, Report on Concepts and Terminology for the Conceptual Schema and the Information base, ed. J J van Griethuysen, N. V. Philips ISA — TMF — ET, Geb.Hsk, 5600 MD EINDHOVEN, THE NETHERLANDS
- 17 David Bell, EEC Medical Project, Ulster Polytechnic (private communications).