Database Interoperability Issues in Intelligent Networks

Antoni Wolski

Technical Research Centre of Finland
VTT Information Technology
P.O. Box 1201, FIN-02044 VTT, Finland

wolski@vtt.fi

Abstract

Intelligent Networks are evolving as data-intensive multi-layer information systems. Databases can be found within or associated with various functional entities of the IN reference model. The aspects of database distribution and heterogeneity are present both within the functional entities and between them, in a real implementation. In particular, the Service Data Function (SDF) has to be implemented in such a way that two different physical entities: the Service Control Point and the Service Management Point have access to the SDF. What makes an IN system a different kind of a distributed and heterogeneous information system is that correctness and reliability requirements vary vastly among different physical entities–and yet transactions are supposed to span many physical entities (real systems). A reference model of IN data management is proposed whereby the full service lifecycle is addressed, including service creation, deployment, provisioning and execution. Essential data management requirements are presented. The needs of transaction processing is presented from the point of view of service provisioning and execution. A classification of transactions interacting with the SDF is proposed, and a corresponding correctness criterion is suggested.

1 Introduction

The term *Intelligent Network* (IN) was introduced within the telecommunications industry to denote a telephone network offering services beyond simple connectivity [AMS89]. A few examples of best known IN services are listed below.

Sixth ERCIM Database Research Group Workshop on Deductive and Interoperable Databases, November 1–2, 1994, Barcelona.
Revised November 14, 1994.

- *Freephone* (or 800-number service) allows callers to make transparent connections to pre-defined, and possibly changing, service provider's routing (destination) numbers. The calls are billed to the service provider.
- *Premium Rate* allows callers to reach services in the same way as in the Freephone service, with the difference that the calls are billed to the caller, using special, service-specific, rates.
- *Virtual Private Network* enables to simulate a private exchange (PBX) in a public network.
- Plethora of *Calling Card* applications allow for personal customisations of the telephone service with regards to billing and connectivity.

The first generation IN systems were introduced in the USA at the end of 1970's. A first generation IN system made use of programmable, computer-based switches which started to proliferate at that time. The required services had to be implemented into numerous switches in a telephone network. This was a costly and cumbersome process for network operators, taking into account the proprietary nature of switch programming techniques. Additionally, it turned out that data management issues became an overwhelming problem: the data of thousands of customers and service specifications had to be maintained in a flexible way, and new service had be introduced promptly. The switch-based technology did not allow for any of that.

In response to market demands, in mid-1980's, Bellcore started a vast effort to define a new standard technology (called also Advanced Intelligent Network, AIN, to differentiate it from the first-generation technology). The effort became quickly a field of co-operation and standardisation. Currently, the standardisation activity is concentrated in ITU-T (International Telecommunication Union, formerly CCITT), Study Group XI. ITU-T has produced the so-called Q.1200-series of recommendations [ITU93] specifying a first phase of IN systems, known as *Capability Set 1* (CS-1). CS-1 includes most of currently known IN services. They are characterised by the centralised (or single-point) control of a service instance and a strong telephony orientation. Future Capability Sets will take into account distributed services and, among others, multimedia services.

First implementations of the standard IN technology are already on the market. They offer new potential but also introduce new problems which are discussed in this paper.

In terms of the IN reference model, an IN system is a distributed information system, with databases possibly located within different (hardware or software) components. From the point of view of the network operator, the IN system is, in the long run, inherently heterogeneous, because the components may be acquired from different sources. To cope with this problem, suitable interoperability techniques have to be defined. In Section 2, various data realms with an IN system are discussed and one of them–the data related to the service

life cycle support–is selected for further analysis. In Section 3 a reference model for database interoperability within IN is proposed.

An IN system is tightly coupled with a switching equipment. Because the switching equipment is a real-time component, there are also real-time components (i.e. such that deadlines have to be taken into account) in a IN system. There are also components that are non-real-time. Because of different natures of the two types of components, maintaining data consistency within the system may turn out to be very difficult, and new transaction management techniques are required. The transaction management issues are discussed in Section 4. The Summary concludes the paper.

2 Data realms in an IN system

Running IN services requires a lot of data to be in place and to be moved throughout the system. The first data realm involves the data needed to manage the system. These are data related to [Ahn94]:

- configuration management
- accounting management and billing
- performance measurements and statistics
- fault management
- security management
- network operations planning

The concepts of Telecommunications Management Network (TMN) and Management Information Base (MIB) are related to the above data. Because of lack of space, this realm will not be dealt with in this paper. For more on the subject, see, e.g. [Ahn94].

The other data realm deals with the maintaining of an IN service through its lifecycle. The lifecycle of a service may be characterised by the following phases:

(1) Service creation. A new generic service (service type), which has been conceptualised, is represented in a computer-readable form required to make the service operational. This may involve many steps and various representations, e.g. a program may be written to implement a missing elementary service component (a SIB, service-independent building block), a the decision logic of the service may be represented using a graphical or textual language, and database structures necessary for the service execution and provisioning may be defined using some data definition language.

(2) Service testing. A new generic service is tested in a simulated environment. The service definitions produced in step (1) are used.

(3) Service deployment. A new generic service is moved to the operational environment. From now on, it can be provisioned (i.e. sold to customers) and used.

(4) Service provisioning. Customers subscribe to the service. Service instances are created. Some instances may be direct (unmodified) instances of the generic service (e.g. a simple Freephone service with a single routing number). Other instances may be created by modifying the generic service. i.e. a service subclass is created and instantiated (e.g. a Freephone service with routing depending on time of day and day of week).

(5) Service execution. Callers activate the service instances by making IN calls (i.e. telephone calls triggering IN service execution). In some cases customers can modify the service instance by changing the service parameters (e.g. changing a routing number).

(6) Service deletion. Both the service type and instance data are removed from the system. The service can be neither provisioned nor used any more.

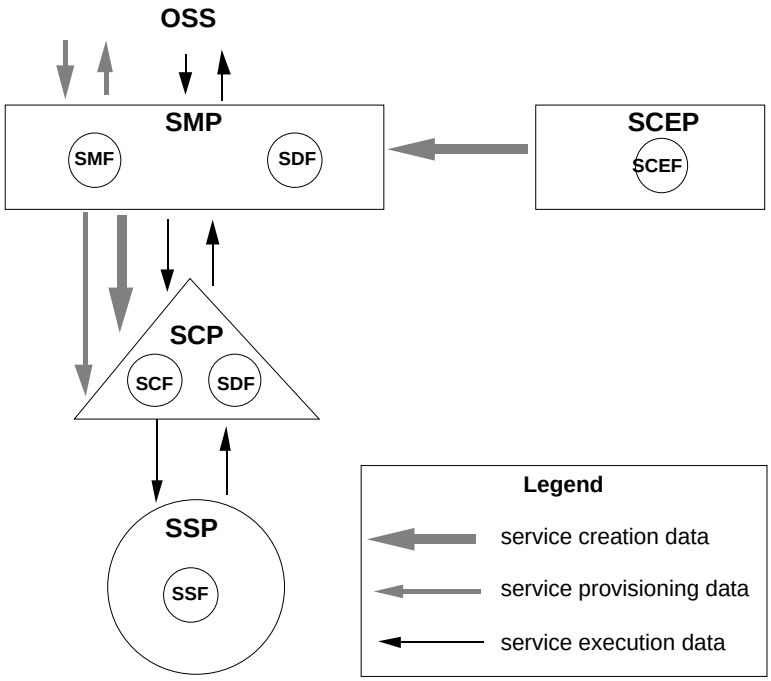
3 Interoperability and service lifecycle

Different IN system components are involved in different phases of the service life cycle. The IN reference model uses four different conceptual *planes* to picture dependencies among different components: the *Service Plane*, the *Global Functional Plane*, the *Distributed Functional Plane* and the *Physical Plane*. For brevity, only the Physical Plane will be discussed here. It depicts relationships between *physical entities* (real systems) and abstract *functional entities*. A typical (one of many possible) IN configuration represented on the Physical Plane is shown in Fig. 1.

The SSP is connected to a telephone switch and is responsible for capturing *IN calls* and directing them to the SCP. An IN call is a telephone call where the dialled number has been associated with an IN service. Such a number is called a *service number*. If a service number is recognised by the SSP, a trigger is activated, and the IN service request is passed to the SCP. The SCP is the operational execution engine of IN services. It uses the SCF to execute the service logic and the SDF to access the data related to service types and instances. In most cases, eventually, the SCP tells the SSP which routing number the call should be connected to. The SMP is used to manage the service type and instance information for the sake of SCP. Among others, new services and new customers are added using SMP. SCEP is used to create new service types. The arrows in Fig. 1 show different kinds of service data and metadata flows in the system, characterised by the lifecycle phase the data is created in. Note, that

adding of a new service or a new customer does not affect the SSP and underlying switches, which has been the purpose of the new architecture[1].

OSS in Fig. 1 denotes Operations Support System encompassing other systems at the telecommunications operator, which may be connected to the IN system. However, the role of and interaction with OSS is not discussed in this paper.



Physical Entities
SSP: Service Switching Point
SCP: Service Control Point
SMP: Service Management Point
SCEP: Service Creation Environment Point
OSS: Operations Support System

Functional Entities
SSF: Service Switching Function
SCF: Service Control Point
SDF: Service Data Function
SMF: Service Management Function
SCEF: Service Creation Environment F.

Fig. 1. An example of the Physical Plane configuration.

One functional entity deserves special attention: the SDF. It can be understood as a database system. The IN reference model does not provide any indication on the implementation techniques, and in particular whether it should be distributed or not. From an abstract point of view, only one instance of SDF is required [Raa94], shared by the SCP and SMP. It has been already recognised that, because of different timing requirements related to different data, it should be implemented partly a main-memory-based and partly as a disk-based database system [Tai94]. However, we can see that a single implementation of SDF would cause at least the following problems:

- Traditional, SMS-originated transactions would block the SCP-originated transactions for unacceptable long periods of time.

[1] In reality, adding of a new service may require modifying trigger specifications in SSP.

- Because the system availability requirements of SCP would exceed those of SMP, a shared solution may be of inferior availability.
- Different transactional requirements require different transactional mechanisms (more on this in Section 4).

Therefore, there exist justifications for a further claim: because of different database requirements at SCP and SMP, there is a need for at least two corresponding implementation instances of SDF, as shown in Fig.1. There may be more of them if SCP or SMP sites are multiplied.

Distribution of SDF in a real system introduces various interoperability problems. They are illustrated in Fig. 2, where different service life cycle phases (exclusive of service testing and deletion) of a SIB-based service are shown using the notation of the ANSI/SPARC DBMS model [Jar77].

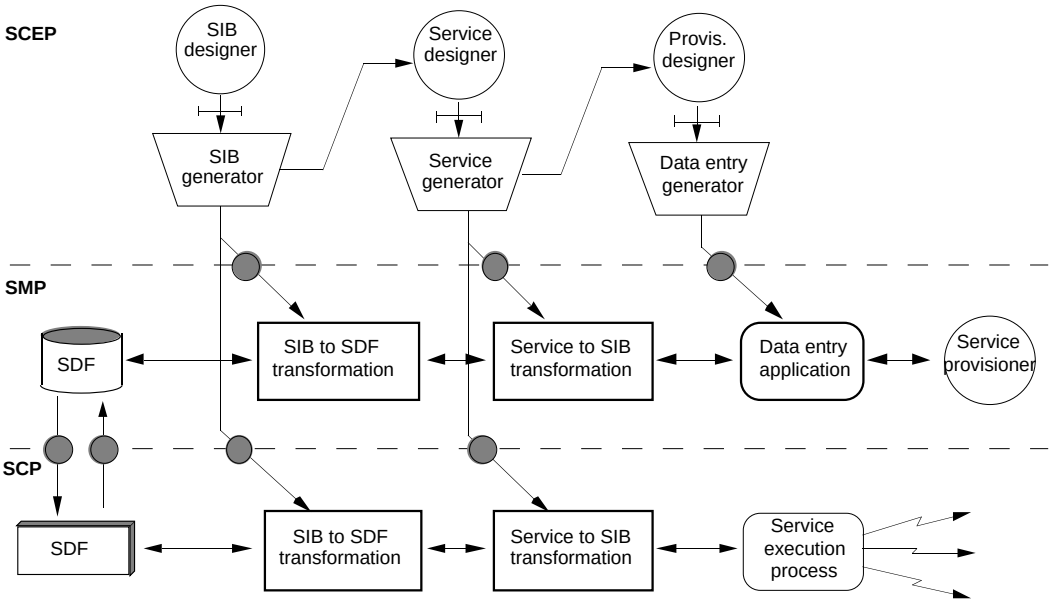


Fig. 2. Service life cycle management model.

The uppermost part of the picture illustrates the service creation phase.

Firstly, a set of SIBs has to be created. A SIB is an elementary building block of a service. It can be understood as an object type encapsulating an elementary behaviour (e.g. "connect to the routing number X" or "charge number Y") and data attributes being, essentially, volatile date, i.e. data maintained only during the call instance or such that otherwise need not be persistent. A simplest way to develop SIBs may be to write them in a programming language like C or C++. The SIB specifications have to be translated to a form which is executable at the SCP and also to specific transformations (mappings) enabling to call a SIB from within a service, at the service run-time, and to transaformations enabling provision the SIB during the provisioning run-time.

Secondly, a service type is defined. It is a control structure including used SIBs and relationships thereof. Persistent data structures are also defined. The data may be static (created and changed only by the provisioning process) or dynamic (possible to change in effect of a caller's activity). An example of the latter one is a "follow-me" routing number which a customer may change at will. Usually, a special language is used to specify services. Necessary data mapping have to be created both for the service execution and provisioning.

Thirdly, a provisioning application has to be specified. This process is similar to building a typical database application. Data entry screens have to be defined and the necessary procedural part created. To automate this process, the SIB and service type specifications can be used.

According to Fig. 2, once the activities of the SIB creation, the service creation and the service provisioning implementation are completed at SCEP, both the provisioning system and the execution system for the service are in place. The middle part of Fig. 2 illustrates the interaction of run-time components of the service provisioning activity, and the bottom part corresponding aspects of the service execution activity.

Both the provisioning activity and the execution activity access data which are logically the same data. The data are however in different physical databases, meaning the data are replicated. This means a replication activity is required to maintain the copy consistency. This activity is shown in Fig. 2. as data flows between the SCP-base SDF and the SMP-based SDF.

If we deal with a heterogeneous environment, and the three parts of Fig. 2 are implemented using incompatible systems, necessary format conversions (shown as shaded circles in Fig. 2) have to be applied. As in a heterogeneous database, various schema, command and data conversion techniques [SL90] can be utilised. Unfortunately, no standardised interfaces or related representation formats exist for IN systems.

4 Transaction management requirements

In a real IN system, transactions span different physical components. Let us concentrate on interworking of SMP and SCP, where most severe problems occur. The problems result from the significant differences in nature of the two systems.

The SCP may be characterised in the following way:

- A real-time system. Although there are no hard deadlines, the service execution requests are to be processed promptly, with the median service execution time of the order of 100 ms.
- A main-memory-based database. Because the required SDF response time is of the order of 10 ms or less, the SDF is implemented in main memory.

- Relaxed transaction processing. In order to meet response time and throughput requirements, real implementations do not support some of the transaction capabilities: most often the atomicity and durability are compromised.
- High availability. SCP should support uniterruptable and continuos operation. The required availability figure is of the order of 99.99%. This is achieved by fault-tolerant techniques, namely the hot stand-by architecture with either synchronous (duplicate components are fully consistent at all times) or asynchronous (there is a delay of updating the stand-by) replication.
- Relaxed recovery. Because of stress on fault-tolerance, the recovery capabilities are of secondary importance. Some data (namely, the volatile and the dynamic data) may be lost during recovery.

The characteristics of SMP are the following:

- A traditional on-line transaction processing (OLTP) system. No real-time requirements, response time tailored to human operators, usually required to be within seconds.
- A disk-based database.
- Full transaction capabilities.
- Moderate availability. Availability typical for business data processing systems is required, i.e. in the range of 95–98%.
- Full transaction-based recovery.

Both SCP and SMP use, in most parts the same service-type-specific and service-instance-specific persistent data. Because of the differences in the nature of the systems, the data have to be replicated. Maintaining transaction processing in the presence of a replicated database, in such a heterogeneous environment, is a challenge. No synchronous copy processing (i.e. with a transaction) is possible. The reasons are:

(1) SCP-originated transactions should not access the SMP database because of time constraints and availability requirements.

(2) SMP-originated transactions should not access the SCP database because this would possibly jeopardise the data availability at SCP.

The above characteristics also precludes use of known copy algorithms which require to maintain a master copy or to access synchronously a number of sites. It is also obvious that achieving serialisable global histories in the system is impossible.

In the rest of this section, a characterisation of the involved transaction types is proposed and a new correctness criterion is suggested. The discussion is based on the assumption asynchronous data replication techniques are used.

Let us assume a transaction involving the SDF in SCP and SMP falls into one of the following disjoint classes (Fig. 3):

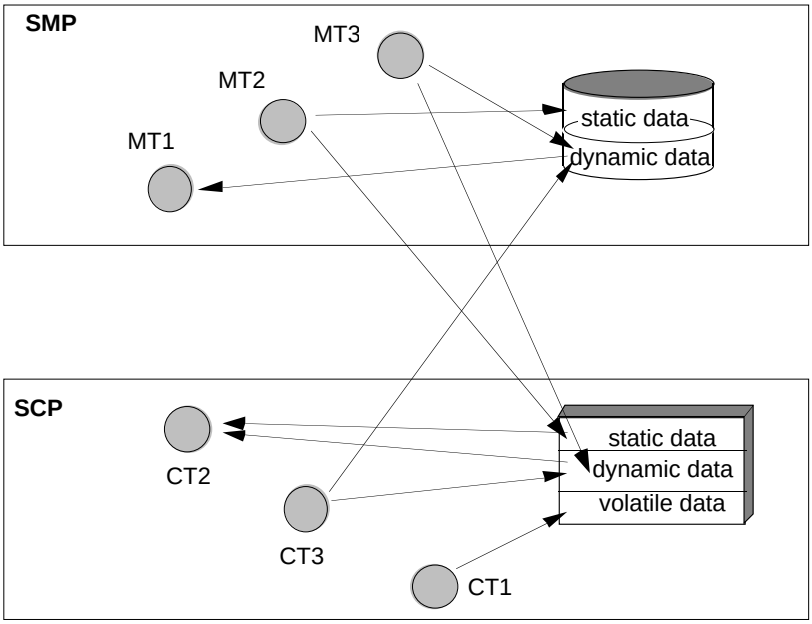


Fig. 3. Types of transactions spanning SCP and SMP.

CT1 Service control transaction accessing volatile data only. Volatile data are data generated by the SCF and are of no interest to the SMF. Such are e.g. state variables and various measurement counters. The summaries of the latter ones are transferred to the MIB part of the system.

CT2 Service control transaction reading static and dynamic data.

CT3 Service control transaction modifying (replicated) dynamic data. Dynamic data modifications are initiated by the service user who may, e.g. enter a new routing number. In CS-1, the data modifications are "blind writes" (the written value does not depend on any read value). CT3 updates, logically, replicated data items. It is implemented as two transactions: the base transaction and the data propagation transaction.

MT1 Service management transaction reading dynamic data.

MT2 Service management transaction modifying static data. MT2 is implemented as two transactions: the base transaction and the data propagation transaction.

MT3 Service management transactions modifying (replicated) dynamic data. MT3 is implemented as two transactions: the base transaction and the data propagation transaction.

We assume certain implementation of the distributed transactions. Any transaction T_i writing replicated data items (i.e. being of the CT3 or MT3 type) is decomposed into two detached *part-transactions :*

T_{bi} the base transaction writing the item(s) at the primary site,

T_{pi} the propagation transaction writing the replicated item(s) at the secondary site.

The primary site is SCP for CT3 transactions and SMP for MT3 transactions. The replication transparency is achieved by executing explicitly the base part-transaction T_{bi} only. The propagation transaction T_{pi} is invoked automatically by an asynchronous triggering mechanism. The failure atomicity of the T_{bi}–T_{pi} pair is provided by using replication logs. However no concurrency atomicity (isolation) is provided

All of the transactions are required to be atomic. In cases when only one item is updated, the atomicity requirement is reduced to the requirement that the execution of a database command is atomic. All the MTx transactions are required to be durable. The durability is not mandatory for CTx transactions (because the updates by the CTx transactions are of secondary nature, and the probability of a failure is small in a fault-tolerant system).

We say that any two transactions T_1 and T_2 *conflict* with each other if there is there is a pair of the corresponding operations o_1 and o_2 conflicting with each other in terms of the traditional r-w conflicts [BGH87]. Possible conflict types are: read-write, write-read and write-write.

Transactions of the CT1 class do not conflict with transactions of any other class. Because the implementation of this transaction class is trivial, the class is not discussed on the sequel.

Mutually conflicting are the following transaction type pairs involving SCP and SMP originated transactions:

- CT2–MT2 (read-write conflict). The requirement is that the transactions are serialised with respect to each other.
- CT2-MT3 (read-write conflict). The requirement is that the transactions are serialised with respect to each other.
- CT3–MT1 (read-write conflict). The requirement is that the transactions are serialised with respect to each other
- CT3–MT3 (write-write conflict). The requirement is that the transaction should be

As concerns correctness of interleaved transaction executions, the primary requirement is that the committed projection [BGH87] of the global history over the MTx transactions is serialisable. Another requirement is that the inclusion of the CT3 transactions produces also a serialisable history. The problem here is that, in the case of the CT3–MT3 conflict, the effects of a data modification transaction is demonstrated by the data propagation transaction executing at a replication site once the base transaction has been already committed at the primary site. This may lead to serialisability errors of the lost update type. However, if the conflict is detected, it can be resolved by removing (deleting) one of the offending transactions from the history, without invalidating the his-

tory correctness. This is comparable to aborting a committed transaction and avoiding cascading aborts [BGH87].

In fact, the goal is possible to achieve within stated assumptions. Note, that all the CT3 transactions use blind writes. This means no transaction writing to the database in SCP, as a primary site, is dependent on data written by another transaction. Thus, a committed CT3 transaction may be *deleted* without invoking cascading aborts. By deleting a TC3 transaction we mean aborting the propagation transaction and compensating the base transaction which has been already committed.

We claim this will guarantee sufficient consistency of the SMP database at all times. As concerns the SCP database there will be intermittent inconsistent states (when a CT3 transaction is committed and not compensated yet). We suggest to accept such states. In a quiescent state, the SCP and SMP database will be consistent (having identical values of replicated items) with each other.

The other conflicts types listed above will be dealt with using the data propagation transactions and local serialisability-preserving concurrency control mechanisms.

Based on the above discussion, we propose the following correctness criterion for a replicated SDF.

> **Definition**: SM-correctness (service-management-correctness).
>
> An interleaved execution of SCP and SMP originated transactions over a replicated SDF database is SM-correct if
>
> (1) both the SMP SDF and SCP SDF produce locally serialisable histories
>
> (2) replicated data are maintained using asynchronous data propagation transactions.
>
> (3) if a CT3-MT3 pair invokes globally nonserialisable history, the CT3 transaction is deleted.

The above correctness criterion is acceptable if the service semantics allows for lost of some user-originated transactions. There is a suitable analogy within the basic connection service: some calls fall through and are not connected. What is important from the service quality point of view is that the failure rate is held within a certain limit.

Another dimension of difficulty is introduced by further replication of data among parallel SCPs or SMPs. To cope with the problem, special methods to guarantee certain level of the overall database consistency, such as. the Atomic Delayed Replication method of [GJK94] are needed.

Summary

Intelligent network represents a formidable display of interoperability problems. We have proposed a reference model identifying database access problems in an IN system, with respect to the service life cycle. The inherent database

heterogeneity in an IN system, resulting from different purposes of various components imperils transaction processing in the system. We have analysed different transaction types and the associated transaction processing requirements. We can see that satisfactory correctness of the IN system operation may be achieved by allowing for some SCP-originated transactions to be lost. The corresponding SM-correctness criterion has been defined.

References

[Ahn94] Ilsoo Ahn. Database Issues in Telecommunications Network Manegement. *Proc. SIGMOD-94 Conf.*, May 24-27, Minneapolis, Minnesota, USA, pp. 37-43.

[AMS89] W. Ambrosh, A Mahler, B. Sasscer. *The Intelligent Network.* Springer-Verlag, 1989.

[BHG87] P.A. Bernstein, V. Hadzilacos and N. Goodman. *Concurrency control and recovery in database systems.* Addison-Wesley Publ. Comp., 1987.

[ITU93] International Telecommunication Union (ITU), Standardization Sector. Q.1200: Q-Series Intelligent Network Recommendation. ITU-T Study Group XI, March 1993.

[Jar77] D. A. Jardine (ed.). The ANSI/SPARC DBMS model. *Proc. Second SHARE Working Conference on Database Mananagement Systems,* Montreal, Canada, 26-30 April 1976, North-Holland Publishing Co.

[GJK94] R. Gallesdorfer, M. Jarke and K. Klabunde. *Intelligent Networks as a Data Intensive Application (INDIA).* Tech. rep. 94-20, RWTH Aachen, Fachgruppe Informatik, Aachener Informatik-Berichte, 1994.

[Raa94] K. E. Raatikainen. Database Access in Intelligent Networks. *Proc. Third Summer School on Telecommunuications, Volume I: Workshop on Intelligent Networks,* August 8-9, Lappeenranta, Finland, 1994, Lappeenranta University of Technology, Dept. of Information Technology, pp. 163-183.

[SL90] A. P. Sheth and J. A. Larson. Federated Database Systems: for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, Vol. 22, No. 3 (September 1990), pp. 183-236.

[Tai94] J. Taina. Problem Classes in Intelligent Network Database Design. *Proc. Third Summer School on Telecommunuications, Volume I: Workshop on Intelligent Networks,* August 8-9, Lappeenranta, Finland, 1994, Lappeenranta University of Technology, Dept. of Information Technology, pp. 185-199.