Abstract

This paper presents an overview of the data management framework and the object query services needed for future telecommunication networks. These networks called Information Networks are currently being defined by the world-wide TINA project (Telecommunication Information Networking Architecture) [1], which groups the main telecommunication operators and telecommunication manufacturers. Information Networks increase the need to manage data shared by various applications like management services or communication services. To serve this purpose, France Telecom/CNET is contributing to TINA [4] to propose a data management framework for these Information Networks. This work is the starting point for a joint experiment between the CNET and PRiSM laboratories on a telecom query service as a TINA auxiliary project [5].

1. Introduction

For the time being, telecom networks are based on different standards. Each one addresses a specific purpose: (1) provisioning of basic communication services (i.e., voice, data), (2) network supervision and management (i.e., telecommunication management networks), (3) provisioning of added value services (i.e., free phone, credit card, mobile, videotext, web...). This leads to a bad integration of new technologies such as
object-oriented approach and broadband, to a multiplication of heterogeneous tools, to a bad network interoperability preventing to build sophisticated composed services, to a bad flexibility preventing to introduce easily new services.

To cope with these issues, the TINA project is currently defining a common information networking architecture which can be applied to a wide variety of networks. The main challenge is to integrate, as much as possible, technology coming from the computer science, while keeping compatibility with certain telecom standards.

Within this context, data management is considered as highly relevant. For example, Telecommunication Management Networks manage entities like pieces of software or data. These are grouped into Management Information Bases (MIB) queried with a telecom query protocol (CMIS: Common Management Information Service [7]). Within value added networks, a lot of databases are also needed to store communication service information (e.g. subscription information, charging information, credit card information, routing information...). Finally, new multimedia services based on a web approach will also require data management facilities.

Even if these data management needs seem conventional, the requirements imposed by the telecom environment are not conventional due to: (1) specialized telecom query facilities, (2) a big number of fine grain data and real-time constraints, (3) a federation of real time and conventional bases, (4) a wide area distribution spanning over countries, (5) the interworking with legacy telecom bases using specific telecom protocol interfaces.

In TINA, Information Networks will be based on the ISO Open Distributed Processing (ODP) standard [2] which provides a general framework to design distributed systems and will capitalize on object oriented technology provided by OMG [3] to engineer these systems, this technology being specialized to meet telecom requirements. We are then faced to the integration of the data management approach within ODP and to the specialization of OMG technology to cope with TINA data requirements. This paper intends to make an ODP-like description of these TINA data requirements and addresses a solution we intend to provide: a telecom query service.

The data management framework proposes to application and network designers, a unified methodology to cope with database and object distribution. The main issues are the design of group of objects that can be queried and the integration of these groups of queryable objects into the ODP distribution architecture while keeping compatibility with telecom standards (TMN/ODMA [17]). The Data Management Framework
mainly deals with set oriented manipulation of objects and provides a flexible answer to the integration of ODP and data base distribution approaches within a telecom context.

The telecom query service answers to the main engineering issues related to the query and federation aspects. For the time being in telecom networks, objects are queried with a protocol approach (CMIS [7], X500 [9]). To remain compatible with ODP and OMG approach, query languages are needed for these protocols (as already done in the data base field with the query language SQL and the protocol RDA[25]). Hence we capitalize on the database technology to propose a query language for CMIS: CMIS-L, while keeping the semantic prescribed by the CMIS protocol. This allows to query ODP/OMG objects with a "telecom perfume" and to interwork with existing systems providing protocol interfaces. Another constraint is the variety of queryable objects ranging from stand-alone objects to fine grain objects grouped into collections. These objects are supported by heterogeneous managers: object brokers providing no query facilities, real-time data managers providing limited query facilities, conventional DBMS (relational and object oriented) providing complex query facilities. Hence we propose to federate these various managers starting from the OMG query service [6], but specialized to support telecom query languages.

The remaining of this paper is organized as follows : Section 2 presents the data management framework in TINA following the ODP approach and illustrates it through an example. Section 3 proposes a telecom query service dedicated to TINA and presents the CMIS-L language which constitutes the core of this query service. Finally, Section 4 discusses architectural issues to support this service.

2. Data management framework

The data management framework intends to define the user requirements on data management in a TINA network. It capitalizes on the current status of the ODP standard, the OMG standard, data management standards (mainly ODMG [8]) and certain telecom standards, with the aim of bridging the gap between them. This section briefly recalls the ODP baselines and presents the information and computational ODP viewpoints of the data management framework in TINA. These viewpoints are then illustrated through a sample of a telecom application.

2.1 ODP overview

The ODP Reference Model [2] defines an architecture which supports distribution and interoperability. An ODP system is mainly described with four viewpoints:
• the information viewpoint focuses on the conceptual aspect of a system, without bearing in mind the realization of this system. The semantics of applications is defined by the means of information objects.

• the computational viewpoint represents a distributed system as seen by application programmers. Distributed applications are designed as computational objects. The computational objects provide multiple interfaces defined using an Interface Definition Language (IDL). They can be grouped. The distribution transparency requirements related to computational object interfaces are expressed at this stage. They relieve application programming from object distribution management, protocol knowledge.

• the engineering & technological viewpoints are the concern of the operating systems and protocol experts. They provide support to transparency functions. They focuses on the technical components and products from which a system is built.

The remaining of this section focuses on the information and computational viewpoints of the data management framework. The engineering & technological viewpoints are addressed in Sections 3 and 4.

2.2 Overview of the Data management framework

Data are modeled within the architecture as Object Data [8] and must not be confused with ODP computational objects. But there is a need to realized information searches that not only acts on object data but also on computational objects [10]. Conventional or telecom query languages are used for these searches: OQL [8], SQL2, CMIS (Common Management Information Service) [7], X500query [9]. X500queries are used for directories. In the remaining of the document, X500query and CMIS will be confound as they are similar.

To cope with these query issues, the notion of Object Base (OB) is proposed. An object base is a group of objects which can be queried within the same querying space. An object base can be composed of object data, computational objects or both. Let us call them queryable objects. An object pertaining to an object base can be completely, partially or not at all managed by a conventional database manager. These objects can be persistent or not. The object base approach generalizes the object-oriented database concept. It completes the OMG data service approach with an information and a computational viewpoints. It integrates more classical database client/server approach, and more telecom oriented query languages such as CMIS [11].
2.3 Information viewpoint

Following the ODP reference model, we identify certain ODP information objects that model queryable objects pertaining to an object base ("queryable information objects"). No restriction is placed on the representation of queryable information in a real system (this choice is done at the computational level) or on the means to manipulate them (query languages), or on their life time (a queryable object can be persistent or transient). We call an Information Base (IB) a set of information objects defined by means of a schema and logically related to the same administration and naming authority.

From a conventional database approach, an information base is a conceptual schema, but modeling objects which are not necessarily stored in a database. The semantics of an information base is defined using the OMT graphical language [19] and refined with the GDMO specification language [12] (issued from ISO management field). These languages are also used to defined conventional information objects in TINA. The conceptual schema expressed containments, relationships and collections. The containment models hierarchic naming (also issued from ISO management field) which is used by telecom query languages. The collections [8] are introduced to model the basic grouping which will be used by query languages at the computational level.

Lastly, following usual methodology in database design [13], it is useful to federate information bases when they are designed in an autonomous way. The schemas describing information objects of one base can be different from those describing the same kind of objects in another base, through they can share some part of information. In that case information bases can be coupled to provide a common querying space. The federation is necessary to integrate legacy databases.

One telecom example of information base is a Management Information Base (MIB) [15] which models queryable information of a managed system. Another example is the model of the database of a Service Data Function in Intelligent Networks. A last example is the model of a group of computational objects that will execute on a OMG/CORBA platform and that will be queried with a query language. These examples illustrate the generality of this concept and its specialization for different legacy or new networks.

2.4 Computational viewpoint

An object base is a group (TINA computational concept) of queryable objects (object data or computational objects) handled by an Object Base Manager (OBM). An object base manager is a generalization of various concepts: conventional database manager, query manager proposed by OMG [6], multiple object request server proposed by ISO/ODMA [17] or any object querying other objects. A
computational object using an object base relies on the services that this object base offer at its interfaces and on the object base query languages.

• **Interfaces**

Object base interfaces are defined using an interface definition language: IDL from OMG [3], IDL from TINA (variant of IDL with multiple interface object). Queries are transmitted through these interfaces with location and access transparencies. Query language heterogeneity is retained. Several profiles of query languages will be provided: some for conventional data management needs (OQL & SQL profiles) and other for telecom needs (CMIS profiles). The main long term objective is to promote a convergence between these various languages.

• **Object base definition**

Query languages act on objects modeled following certain rules. These objects must be defined with an object base definition language that may differ from the interface definition language. Such a language defines objects which are interpreted by an object base manager. As an object base may be composed of object data or of computational objects, several object base definition languages are used: ODL/ODMG, IDL. There is no limitation in this field and future languages like SQL3 data definition language will also be used.

• **Federation/interworking**

The large scale and the wide area distribution of our networks lead to a partitioning of object bases into autonomous local object bases managed by one object base manager and to global object bases managed by federated object base managers. The schema of these federated object bases are designed in an autonomous way. A special case of object base federation is the interworking with legacy telecom databases, the schema are not only different but the interfaces also: CMIP protocol with ASN1 type system for legacy management bases. Another case of partitioning is the federation of conventional DBMS and real-time object managers providing reduced query facilities.

• **Transparencies**

As any ODP computational interface, object base interfaces can be characterized by distribution transparencies. Programmers only express qualitative requirements (location hiding, replication hiding, failure hiding...); they are relieved from the task of programming the corresponding engineering mechanisms.

The query interfaces provides location and access transparencies which are mandatory. This means that query expression as seen by programmers must be independent of the protocol used to transmit the query at
the engineering level. This can be achieved for OQL and SQL, but this leads to define true telecom query languages CMIS-L (or X500-L) as telecom query facilities are for the time being too much protocol oriented (CMIS is the call level of the CMIP protocol). The global query interfaces provided by global object base managers allow to provide a "distributed query transparency".

- **Object base trading**
  The trader is an enhanced directory object. It enables a client computational object to get interface references when it needs a service provided by server computational objects [20]. In our context, the trader provides object base offers. It enables an application object to ask for a service (object base name, query language profile, distribution information, meta information...) related to an object base and to ignore which object base manager is addressed.

To sum up, within the architecture, data distribution can be handled at two levels: (1) at the trading level, in that case the distributed queries must be decomposed into local queries by the application objects, and (2) at the global object base level, in that case technology derived from federated DBMS [13, 18] can be used to provide distributed query transparency (but the trading can also be used internally by global object base managers). The first case is well suited to wide area distribution, the second one is more suited to local area distribution. In any case, the choices between these two alternatives are left to the network designers.

### 2.5 Case study

This section illustrates the data management information and computational viewpoints through a sample of a typical telecom application.

**Information viewpoint**

The information bases are composed of queryable information objects specified using OMT and GDMO specification languages. TINA examples of queryable information objects are:

- **Subscription information objects** modeling services provided to users via a subscription. The subscription describes the subset of services used by a subscriber and the charging count of this subscription.

- **Charging information objects** modeling charging mode applied to a subscription and objects acting to realize charging: quota objects which do time metering, calendar objects providing reduced tariff information, tariff objects providing information about quotas.
It is interesting to define several information bases corresponding to information objects defined by different design teams. This could be: (1) a subscription information base that groups all the objects pertaining to the first category, (2) a charging information base that groups objects of the second category. The subscription information base is composed of fine grain objects grouped into collections while charging information base is composed of stand-alone objects.

Example of subscription and charging information bases defined using OMT graphical notation are illustrated by the picture given in Figure 1. Ellipses represent information grouped into collections. Rectangles represent stand-alone information. Containment links represent a naming tree.

![Subscription Information Base](image1)

![Charging Information Base](image2)

**Figure 1**: Information base definition with OMT.

The subscription information base and charging information base can be federated for management purposes. The simplest federation mechanism consists in defining a global containment hierarchy (global naming tree), but more complex federations are possible (naming and structure translation).

**Computational viewpoint**

The information bases are mapped into object bases, the transformations can be more or less complex as for conventional database design methodology. The stand-alone objects are defined using the IDL definition language (computational objects) and collections are defined using the ODL/ODMG definition language (object data). The naming tree definition is introduced with distinct declaration statements, together with the relative distinguished name attribute needed for each collection or stand-alone object definitions. The subscription information base can be mapped into a subscription object base composed of: a subscription_contract collection, a service_template collection, a user_profile collection. The charging information can be mapped into a charging object base composed of: service_provider_tariff objects, quota objects, calendar objects. The two object base definitions are provided respectively by the Figures 2 and 3.
<table>
<thead>
<tr>
<th><strong>Object_Base</strong></th>
<th><strong>Subscription_obase</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Information</strong></td>
<td><strong>Subscription_contract</strong></td>
</tr>
<tr>
<td><strong>Collection</strong></td>
<td><strong>Subscription_contracts</strong></td>
</tr>
<tr>
<td><strong>keys</strong></td>
<td><code>subscription_id</code></td>
</tr>
<tr>
<td><strong>attribute</strong></td>
<td><code>subscription_id</code> (integer)</td>
</tr>
<tr>
<td><strong>attribute</strong></td>
<td><code>subscription_rdn</code> (string)</td>
</tr>
<tr>
<td><strong>attribute</strong></td>
<td><code>tariff_mode</code> (integer)</td>
</tr>
<tr>
<td><strong>attribute</strong></td>
<td><code>charge</code> (integer)</td>
</tr>
<tr>
<td><strong>attribute</strong></td>
<td><code>services</code> (set&lt;service_id&gt;)</td>
</tr>
<tr>
<td><strong>attribute</strong></td>
<td><code>users</code> (set&lt;user_id&gt;)</td>
</tr>
<tr>
<td><strong>operation</strong></td>
<td><code>increment_charge()</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Interface</strong></th>
<th><strong>User_profile</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Collection</strong></td>
<td><strong>user_profiles</strong></td>
</tr>
<tr>
<td><strong>keys</strong></td>
<td><code>user_id</code></td>
</tr>
<tr>
<td><strong>attribute</strong></td>
<td><code>user_id</code> (integer)</td>
</tr>
<tr>
<td><strong>attribute</strong></td>
<td><code>user_rdn</code> (string)</td>
</tr>
<tr>
<td><strong>attribute</strong></td>
<td><code>subscription_id</code> (integer)</td>
</tr>
<tr>
<td><strong>attribute</strong></td>
<td><code>terminal_type</code> (integer)</td>
</tr>
<tr>
<td><strong>operation</strong></td>
<td><code>Nap()</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Interface</strong></th>
<th><strong>Service_template</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Collection</strong></td>
<td><strong>Service_templates</strong></td>
</tr>
<tr>
<td><strong>keys</strong></td>
<td><code>service_id</code></td>
</tr>
<tr>
<td><strong>attribute</strong></td>
<td><code>service_id</code> (integer)</td>
</tr>
<tr>
<td><strong>attribute</strong></td>
<td><code>service_rdn</code> (string)</td>
</tr>
<tr>
<td><strong>attribute</strong></td>
<td><code>service_type</code> (integer)</td>
</tr>
<tr>
<td><strong>attribute</strong></td>
<td><code>service_provider_id</code> (integer)</td>
</tr>
<tr>
<td><strong>attribute</strong></td>
<td><code>service_provider</code> (Person)</td>
</tr>
</tbody>
</table>

---

**Figure 2**: Subscription Object Base Definition.

<table>
<thead>
<tr>
<th><strong>Object_Base</strong></th>
<th><strong>Charging_obase</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Interface</strong></td>
<td><strong>Service_provider_tariff</strong></td>
</tr>
<tr>
<td><strong>attribute</strong></td>
<td><code>tariff_rdn</code> (string)</td>
</tr>
<tr>
<td><strong>attribute</strong></td>
<td><code>service_provider_id</code> (integer)</td>
</tr>
<tr>
<td><strong>operation</strong></td>
<td><code>find_quota</code> (in <code>tariff_mode</code>, in <code>rate</code>, out <code>quota_ref</code>)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Interface</strong></th>
<th><strong>Calendar</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>attribute</strong></td>
<td><code>calendar_rdn</code> (string)</td>
</tr>
<tr>
<td><strong>attribute</strong></td>
<td><code>service_provider</code> (Person)</td>
</tr>
<tr>
<td><strong>operation</strong></td>
<td><code>comp_rate</code> (in <code>date</code>, out <code>rate</code>)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Interface</strong></th>
<th><strong>Quota</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>attribute</strong></td>
<td><code>quota_rdn</code> (string)</td>
</tr>
<tr>
<td><strong>attribute</strong></td>
<td><code>quota_type</code> (integer)</td>
</tr>
<tr>
<td><strong>attribute</strong></td>
<td><code>time_period</code> (integer)</td>
</tr>
<tr>
<td><strong>operation</strong></td>
<td><code>activate_metering</code> (in <code>accountable_object_ref</code>)</td>
</tr>
<tr>
<td><strong>operation</strong></td>
<td><code>deactivate_metering</code> (in <code>accountable_object_ref</code>)</td>
</tr>
<tr>
<td><strong>operation</strong></td>
<td><code>change_quota</code> (in <code>time_period</code>)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Naming_tree</strong></th>
<th><strong>Subscription_nt</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>root</strong></td>
<td><strong>Subscription_obase</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Interface</strong></th>
<th><strong>subscription_contract</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>named_by</strong></td>
<td><strong>subscription_obase</strong></td>
</tr>
<tr>
<td><strong>with</strong></td>
<td><code>subscription_rdn</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Interface</strong></th>
<th><strong>user_profile</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>named_by</strong></td>
<td><strong>subscription_contract</strong></td>
</tr>
<tr>
<td><strong>with</strong></td>
<td><code>user_rdn</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Interface</strong></th>
<th><strong>service_template</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>named_by</strong></td>
<td><strong>subscription_obase</strong></td>
</tr>
<tr>
<td><strong>with</strong></td>
<td><code>service_rdn</code></td>
</tr>
</tbody>
</table>

---

**Figure 3**: Charging Object Base Definition.
3. Telecom Query Service

3.1 Telecom query service requirements

The engineering and technological viewpoints of TINA are based on specialized OMG platforms called Distributed Processing Environment (DPE) [16]. The philosophy is to start from the OMG approach and to specialize it in order to match the telecom requirements. OMG platforms provide engineering mechanisms to support the distribution of computational objects: interface definition language compilers, generation of engineering objects such as proxy, skeleton, locators... The TINA refinements imply the addition of trading services, some ODP transparencies, specific transaction services [22, 23] and in our case a telecom query service [5] designed starting from the Object Query Service OQS proposed by the OMG service architecture.

The object base management in a telecom environment introduces specific requirements as mentioned in the introduction. First, query managers must support telecom query languages as conventional complex query languages are not really needed by all the applications (especially pure communication applications) and as there is a certain reserve to use conventional query languages in the telecom field (this last argument which may seem strange is important and must not be underestimated). Second the queryable objects are managed by heterogeneous data managers like standard DBMSs supporting full query facilities, real-time data managers supporting reduced query facilities and stand-alone object managers providing no query facilities. To cope with this issue, object base managers must provide support to telecom query languages and must federate heterogeneous data and object managers. In this section, we concentrate on the query language definition and on the federation of object bases. The functional architecture proposed to support this language is presented in Section 4.

Although the conventional query management needs encountered in certain part of Information Networks should be covered by the future implementations of the OMG Object Query Service [24], we are faced to the satisfaction of the TINA specific requirements. To deal with such requirements, a "CMIS like" query language and a telecom query service designed starting from OQL and OMG query service are proposed. The main throughput of this approach are the compatibility with OQS (the telecom query service can interwork with it and can reuse certain of its components) and the interworking possibilities with legacy telecom equipments.
3.2 Querying objects in telecom networks

For the time being there are two main approaches to query objects in telecom networks:

- The database approach which relies on query languages bound to programming languages. These query languages can be used in a distributed environment (client/server DBMS for example) with access and location transparencies. Data manipulation protocols used in these distributed environments like RDA (Remote Data Access) are not seen by the applications.

- The telecom approach where objects are queried by means of protocols like CMIS/CMIP (or X500). These protocols are directly manipulated by applications and access transparency is not provided.

The database approach is compatible with ODP. Client/server architectures, masking remote data access protocols already exist. The telecom approach is too closely linked to a "protocol view". To remain compatible with the ODP approach, a uniform approach must be adopted and a query language syntax is needed for CMIP. An OQL-like syntax is a good candidate. Queries are then expressed on "OMG objects", but the semantics of the queries remains that prescribed by the protocol. With this approach CMIS queries can act on "OMG object" and can be transported with another protocol than CMIP, an OMG protocol (IOP) for example.

It should be noted that with this approach there is a great similarity between the proposed CMIS language called CMIS-L and OQL, the query language for object databases promoted by ODMG [8]. CMIS-L is a complete language providing object creation and deletion operations. But it is less powerful than OQL as pure query language. Hence, it is interesting to provide enriched CMIS profiles including certain OQL operations.

3.3 CMIS-L language

CMIS-L allows to query objects defined using OMG/IDL or ODMG/ODL languages. These queryable objects may be stand-alone computational objects or data objects grouped into collections. The semantics of CMIS-L is the same as that of CMIP but its syntax is different. The queryable objects and their attributes are designated by their literal names within the query as in any other query language (instead of ASN1 oids). The literal names are the name of the attributes appearing in GDMO specifications and object base definitions. CMIS-L offers operations to create, delete, search and update one or more objects as well as to initiate actions on them. The operations can be performed on a hierarchy of objects defined by a naming tree following two ways:
• Navigational operation where the object is designated by its distinguished name. For example, the following CMIS-L query retrieves the subscription_id and the tariff_mode of the subscription_contract whose distinguished name is equal to 10.

\[
\text{Get } x.\text{subscription}\_id, x.\text{tariff}\_mode \\
\text{Scope DN } \{\text{subscription}\_rdn = 10\} \\
\text{From } x \text{ in Subscription_contracts}
\]

• Assertionnal operation where the objects concerned by the operation (the scope of this operation) are selected with a scoping and filtering mechanisms. For example, the CMIS-L query given hereafter retrieves all the subscription contracts having a user_id lower to 10.

\[
\text{Get } x.\text{subscription}\_id \\
\text{Scope DN } \{} \text{ Level = 2} \\
\text{From } x \text{ in Subscription_contracts, y in user_profiles} \\
\text{Where } y.\text{user}\_id < 10
\]

The scope indicates the sub-tree, rooted at a base object (defined by a distinguished name: DN), which is to be used for the selection. The depth of a search (level) may be: the base object alone, the base object and all of its subordinates up to the n-th sub-level, the base object and all its subordinates. The filter specifies the predicate that the objects pertaining to the scope must satisfy to be selected. Each elementary predicate about an attribute may be a test for equality, ordering, a presence test or a set comparison. The elementary predicates may be grouped in a recursive way using the logical operators AND, OR and NOT. The matching rules of elementary predicates may be defined by users.

Let us illustrate, with an example, the scoping mechanism which is one of the main particularity of telecom query languages (the same mechanism is used by X500 queries). Given the following naming tree where the nodes are indiced to facilitate the explanation.

![Naming tree example](image)

**Figure 4**: Naming tree example.
A scope can be for example:

- a base object alone:
  node 1 designated by its DN \{subscription_rdn = x1\}.
- a base object and 1 sub-level:
  scope starting at node 1 designated by its DN \{subscription_rdn = x1\},
  as level = 1 the filter is applied to nodes 11, 12, 13.
- a base object and all sub-levels:
  scope starting at node 1 designated by its DN \{subscription_rdn = x1\},
  the filter is applied to all the subordinate nodes 11, 12, 13, 111, 112, 121, 131.

The scoping and filtering mechanisms are used by selection, deletion, update and action operations. The result of a selection is a projection of attributes pertaining to any selected objects. The action operation allows to call methods of a subset of objects selected with the scoping and filtering mechanisms, the signature of all these methods being the same.

### 3.4 Querying federated object bases

The subscription information base and charging information base defined in Section 2 can be federated for management purposes. The simplest federation mechanism consists in defining a global containment hierarchy (global naming tree), but more complex federations are possible (naming and structure translation).

Let us define in Figure 5, a global naming tree which performs the federation of object bases. In this example we assume that we have three local object bases located on different servers: two charging object bases (i.e., the French one and the English one) managed by object managers, and one subscription object base managed by a real-time data manager. An OMT specification can be defined for specifying the global naming tree. For example, it can contain several subjects; each subject containing several countries and each country containing only one object base which corresponds in fact to its local naming tree root. The definition of a global naming tree following the CMIS semantic enables us to express global CMIS-L queries in the same way as local ones.
4. Architectural issues

4.1 Distributed environment

The telecom query service allows to engineer object bases following the OMG/OQS proposal but applying them to telecom query languages. At this level object bases are mapped onto OMG servers (or clusters in ODP terminology) which are the logical unit of distribution. A local object base is located on one server while a global object base is distributed on several servers. Servers support implementation objects (i.e., C++ objects, smalltalk objects, data...).

In the context of telecom query service, four object servers must be considered as depicted in Figure 6. Each server offers some data manipulation capabilities: (1) the queryable collection interface allows to evaluate queries expressed in a specific or conventional query languages (i.e., a telecom query language, SQL, OQL), (2) the collection interface gives facilities to iterate over collections (i.e., first, next, end), and (3) the object interface offers accesses to stand-alone computational objects. For example, the charging object base defined in Section 2 could be managed by an object manager providing an object interface while the subscription object base could be managed by a more powerful server offering either a queryable collection or a collection interface. These three levels of granularity are necessary to offer the extensibility and the performance needed in telecom environment. Indeed, the architecture should remain open in order to integrate easily new object servers with their own capacities of querying.
The telecom query service we propose, does not only rely on the CORBA communication services, but also tries to be as much as possible close to the object query service proposal (OQS) done by OMG [6]. The Object Query Service (OQS) provides a basic framework to manage distributed queries on OMG platforms. OQS conforms to the OMG object model and assumes the use of CORBA-compliant Object Request Broker (ORB) and Interface Repository. It is composed of two main types of engineering objects, namely query manager and collection, providing an independence from technological choices (e.g., DBMS products):

- **The query manager** provides query interfaces to application objects. Queries are transmitted as string and results can be either a value transmitted back or a reference of a collection of queryable objects.
- **The collections** can only be sequentially scanned and the **queryable collections** can be queried through query languages (i.e., SQL, OQL). Both types of collections provide lifecycle operations (create, delete).

At the moment two query language profiles are proposed, namely OQL and SQL2. The query manager and the collections communicate via interfaces defined in IDL. Through these interfaces, we can evaluate queries on objects, collections and queryable collections.

### 4.2 Query service functional architecture

Figure 7 depicts the main components of the telecom query service architecture and shows the interactions between them.

The global query manager deals with the object base federation and is responsible for delegating the evaluation of local queries to the appropriate servers. The global naming tree and local naming trees are stored in a repository called the CMIS repository. Since the CMIS repository only deals with the meta information provided by the information and computational viewpoints, it only stores object base definitions...
(i.e., interface definition and naming trees). The trader provides object base offers. Starting from information
given by the CMIS repository, it provides the references of local object base interfaces.

The decomposer/executor decomposes a CMIS-L query accordingly to the interface level offered by the
participating servers. In order to recompose sub-results and to evaluate filters on selected objects, the query
decomposer generates an OFL program. As detailed in Section 4.4, an OFL program is a functional
expression of a query (or subquery) execution plan which can be evaluated by an OFL evaluator on any kind
of abstract collections of objects. The objects resulting from CMIS-L subqueries can be stored on abstract
collections locally managed by a data manager. This data manager can be an object-oriented DBMS, a real-
time data manager or a main memory data manager.

For local servers, we need to provide CMIS-L interface for example on top of standard DBMSs and on top of
real-time data managers. For standard DBMSs, translators are responsible for translating CMIS-L subqueries
into SQL or OQL queries. For real-time data managers which do not offer high-level querying capabilities, a
CMIS-L compiler and an OFL evaluator are proposed to provide a CMIS-L compliant queryable collection
interface. Roughly speaking, the combination of a CMIS-L compiler with an OFL evaluator produces a local
query manager. The main difference between a global query manager and a local one is the management of
subqueries.

4.3 CMIS-L query processing
The global query manager receives CMIS-L queries expressed on a federated object base. The CMIS-L query manager is responsible for: (1) analyzing the syntax and the semantic of the CMIS-L query, (2) delegating the evaluation of local queries to the appropriate servers, and (3) recomposing the result of sub-queries to produce the final result.

The main tasks of the query manager are the query decomposition and recomposition. Indeed, the query decomposition should be done accordingly to the interface level offered by each participating server. For example if a server provides an object interface, this means that the server only deals with stand-alone objects. In this case, the query decomposer should generate as many calls as there are stand-alone computational objects defined in the local naming tree to retrieve all the objects participating to the query. Then, it will be responsible to evaluate the filter on each object and to merge all the selected objects to compute the final result. In the opposite, if the server offers a queryable collection interface, the query decomposer can generate a CMIS-L query and send it directly to the local server who will be responsible for a local evaluation. Note that the performance of the evaluation of a CMIS-L query strongly depends on the interface level offered by local servers.

4.4 Queryable collection interface

One important task of the telecom query service is to offer as much as possible queryable collection interface on top of local servers to achieve performance. Implementing this interface on top of real-time managers or object managers providing no query facility may require a lot of effort. To address this issue, we propose generic components that can be reused and specialized by several managers. They allow to parse a CMIS-L query and to locally evaluate it. Therefore, we propose a CMIS-L compiler that generates an execution plan expressed in OFL (Object functional language). OFL can be seen as a target intermediate language for object-oriented query compilers [14]. OFL is independent of any data model and any storage model. Moreover, the language comes with a flexible execution model supporting both set-oriented and pipelined (i.e., one object at a time) strategies. The aim of OFL is to provide a unified and optimized execution support for general purpose (e.g., OQL) as well as for domain specific (e.g., CMIS-L) query languages.

OFL introduces the concept of abstract collections for manipulating in a unified way any kind of collection of implementation objects, independently of the type of the collection instances and of the storage organization of the collection. For example, an abstract collection can model a class extent, a multivalued attribute (e.g., a set, a list or an array) as well as system collections like indexes or temporary collections resulting from the evaluation of subqueries. Abstract collections are encapsulated by traversal functions that allow to iterate
over the elements of a collection independently of its physical organization. Traversal functions are similar to collection iterators defined by OMG. Functions map collections of objects to collections of objects. Functions can be used as building blocks to define algebraic operators (e.g., filter, join, reference join, union, intersection, difference, aggregate operators) as well as to express a complete query execution plan. Functions are general enough to express collection traversals, to evaluate unary, binary and n-ary predicates on the current instances of all collections involved in a query, and to apply simple or complex projections on the selected instances.

5. CONCLUSION

This paper gives an overview of data management requirements for the future Information Networks designed within the TINA project. It provides ODP viewpoints of the data management framework allowing to facilitate the design of applications using object bases; the object base concept being a generalization of database concept for Information Networks. To match some of these requirements, the architecture of a telecom query service derived from OMG is proposed, with a CMIS-L query language approach for telecom query facilities. The telecom query service complies with OMG Object Query Service. It is also designed to facilitate interworking with legacy telecom equipments providing pure CMIS interfaces. This dedicated telecom query service is currently being designed by CNET and PRISM laboratory as a TINA data auxiliary project on behalf of the CNET. The main waited results are a first step in the validation of the data management framework proposed for TINA and a technical and practical skill enabling to derive later an industrial version of our prototype.

References


