terface Mapping Services (IMS), whose main purpose is to

ODMG specifies se

- class, attribute or method renaming,
- interface attribute or method mapping to an OQL construct,
- interface attribute or method mapping to a C++ construct,
- addition of attributes or methods in a target interface.

In this case, as shown in Figure 1, the starting point is:

1. a database schema,
2. hints about the view to be built expressed as an IMDL construct.

the result is:

1. a generated IDL,
2. a full or partial CORBA implementation.

2.4.3 Target View driven CORBA views

In this case, the target view is given a

Person::cstate . Person::children and Address::street . This means that the previously shown IDL interfaces becomes:

```
interface Address {
  attribute string town;
};

interface 9.R25 9.962 (the)64w380.63 Tf-101.991 -10399 Td (attribute)Tj 53.8024 0 Td (string)Tj 37.50name Td (t91.31 Tf-
from  // the  get  OQL  construct:
 %oql{  select   x from  Person
```

This generated IDL calls for a few remarks:

1. as previously introduced, the generated IDL includes by default the file `eyedb.idl` which contains the generic interface of EYEDB. Each generated interface inherits directly or indirectly from the `EyeDB _ORB::idbObject` interface.

2. by default, an interface factory is generated which contains a few methods for each generated interface:

   (a) the method `Address AddressQueryFirst(db, query)` returns the first `Address` instance which matches the input OQL `query` argument.

   (b) the method `AddressList AddressQuery(db, query)` returns all the instances of `Address` which matches the input OQL `query` argument.

   (c) the method `Address asAddress( EyeDB _ORB::idbObject o)` builds an `Address` instance from a generic `idbObject` instance if and only if the generic instance is of dynanic type `Address` . Otherwise a null object is returned.

   (d) the method `Address AddressCreate(db)` creates an empty runtime `Address` instance.

4.2.3 Using the generated CORBA view

To use the CORBA view, the user must:

1. generate the CORBA view by giving the database schema and the IMDL construct to the IMS compiler.

2. generate the CORBA stubs and skeleton by giving the generated IDL to the ORB IDL compiler,

3. compile all the generated code,

4. write and compile the client programs.

Here is a simple example of a client program to get, display and update all the instances of the class `Employee` :

```
// making a OQL query
EyeDB_ORB::idbObjectSeq_var obj_seq =
  db->queryObjects("select Employee");

for (int i
```

The EYEDB OODBMS has been developped at Sysra [**?**] with partial funding from the Agence National de la Valorisation de la Recherche (ANVAR) [**?**] and the Conseil Regional d'Ile de France [**?**].