Distributing CORBA Views From an OODBMS

Eric V

terface Mapping Services (IMS), whose main purpose is to generate the CORBA bridge for the given OODBMS and ORB, both depend on the OODBMS and the ORB used.

Because ORBs followe

ODMG specifies several level of compliance: object model compliance, ODL compliance, OQL compliance, C++ binding compliance and so on.

Several product databases are currently partially compliant with the ODMG standard: O2[2], ObjectStore [10], POET [24], VERSANT [12], ONTOS [4], Objectivity [16].

EYED()Tj 9.35525 0 Td ([30)Tj 13.1797 0 Td (])Tj 5.87739 0 Td (and)Tj 16.7756 0 Td (so)Tj 11.3867 0 Td (on.)Tj /R22 10.9

- class, attribute or method renaming,
- interface attribute or method mapping to an OQL construct,
- interfattribute

`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.

In those both approaches, as in the traditionnal RDBMS view approach and contrary to our approach, the view gathers “production” and “filtering” rules: