ОБЪЕКТНО-ОРИЕНТИРОВАННЫЕ СИСТЕМЫ БАЗ ДАННЫХ В СТАНДАРТЕ ODMG

Социальный прогресс требует стандартизации людей,
и эту стандартизацию называют равенством.

Э. Фромм

Пост-реляционные системы баз данных

Содержание

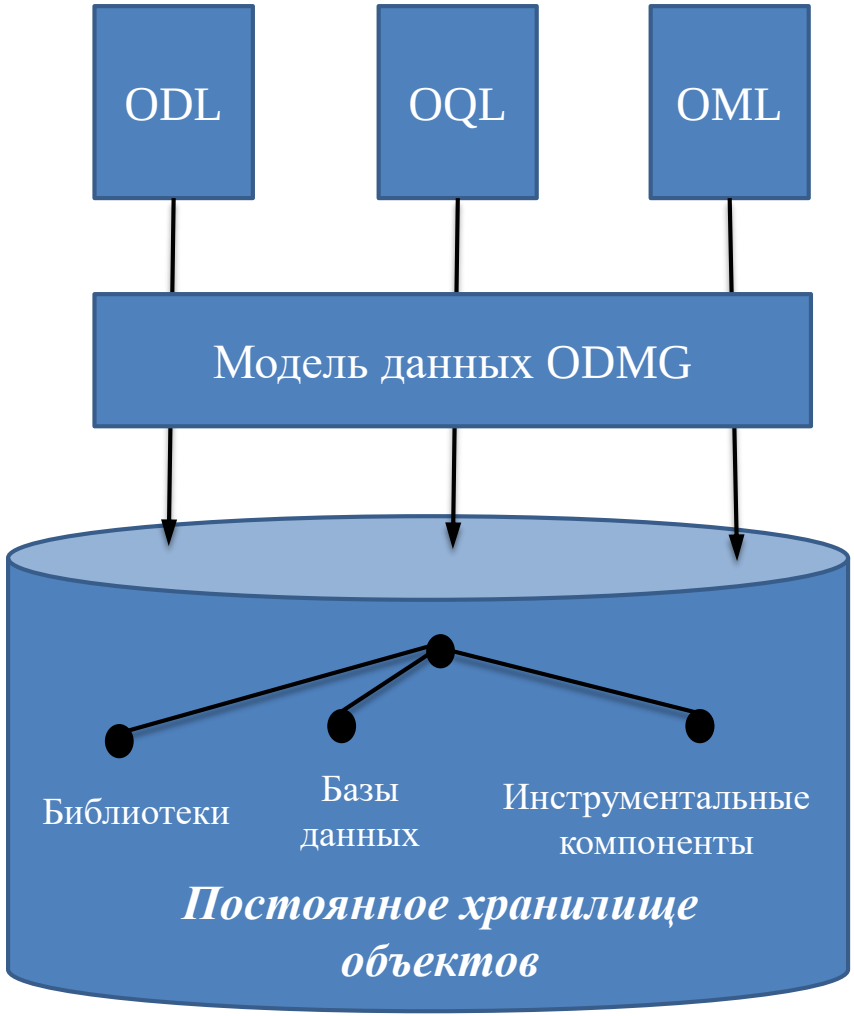
- Консорциум ODMG
- Архитектура ODMG для ОО систем баз данных
- Язык описания ODL
- Язык запросов OQL

ODMG

- Консорциум ODMG (*Object Database Management Group*, впоследствии *Object Data Management Group*) образован в 1991.
- Тесно связан с более многочисленным консорциумом OMG (*Object Management Group*), который образован в 1989.
- Основная цель ODMG – выработка промышленного стандарта объектно-ориентированных баз данных (общей модели). За основу была принята базовая объектная модель OMG COM (*Core Object Model*). ODMG опубликовала три базовых версии стандарта (последняя – ODMG 3.0).
- Некоторые стандарты OMG опираются на объектную модель ODMG, например, спецификация языка OCL (Object Constraint Language), являющаяся частью общей спецификации языка UML 1.4 и UML 2.0.

Архитектура ODMG

- Архитектура ODMG определяет способ хранения данных и различные виды пользовательского доступа к "хранилищу данных".
- Единое хранилище данных доступно из языка определения данных, языка запросов и ряда языков манипулирования данными.
- Модель данных представляет организационную структуру, в которой сохраняются все данные, управляемые ООСУБД.
- Языки ODL, OQL, OML опираются на модель данных. Архитектура допускает существование разнообразных реализационных структур для хранения моделируемых данных, но все программные библиотеки и поддерживающие инструментальные средства обеспечиваются в ОО-рамках и должны сохраняться в согласовании с данными.



Компоненты архитектуры ODMG

- Объектная модель данных
- Язык определения данных (Object Definition Language, ODL).
- Язык объектных запросов (Object Query Language, OQL)
- Языки манипулирования объектами (Object Manipulation Languages, OML)
- Постоянное хранилище объектов
- Инструментальные средства и библиотеки

Компоненты архитектуры: объектная модель данных

- Все данные, сохраняемые ООСУБД, структурируются в терминах конструкций модели данных. В модели данных определяется точная семантика всех понятий.

Компоненты архитектуры: язык определения данных (ODL)

- ODL позволяет описывать схему в виде набора интерфейсов объектных типов, что включает описание свойств типов и взаимосвязей между ними, а также имен операций и их параметров.
- ODL не является полным языком программирования; реализация типов должна быть выполнена на одном из языков категории OML.
- ODL является виртуальным языком программирования.
  - В стандарте ODMG не требуется реализация ODL в программных продуктах ООСУБД, которые считаются соответствующими стандарту. Допускается поддержка этими продуктами эквивалентных языков определения, включающих все возможности ODL, но адаптированных под особенности конкретной системы.
  - Тем не менее, наличие спецификации языка ODL в стандарте ODMG является важным, поскольку в языке конкретизируются свойства модели данных.

Компоненты архитектуры: язык объектных запросов (OQL)

- OQL имеет синтаксис, похожий на синтаксис языка SQL, но опирается на семантику объектной модели ODMG.
- В стандарте допускается прямое использование OQL и его встраивание в один из языков категории OML.

Компоненты архитектуры: языки манипулирования объектами (OML)

- OML представляет собой интегрирование объектно-ориентированного языка программирования с моделью ODMG для программирования реализаций операций и приложений.
- Интегрирование производится за счет определенных в стандарте правил языкового связывания (language binding).
  - В языках программирования непосредственно не поддерживается стабильность объектов.
  - Чтобы разрешить программам на этих языках обращаться к хранимым данным, языки должны быть расширены дополнительными конструкциями или библиотечными элементами. Эту возможность и обеспечивает языковое связывание.
- В одной ООСУБД могут поддерживаться несколько OML. В стандарте ODMG -2 специфицированы правила связывания для языков C ++, Smalltalk и Java.

Компоненты архитектуры: постоянное хранилище объектов

- Логическая организация хранилища данных любой ООСУБД, совместимой со стандартном ODMG, должна основываться на модели данных ODMG.
  - Физическая организация у разных ООСУБД может различаться, но в любом случае она должна обеспечивать эффективные структуры данных для хранения иерархии типов и объектов, являющихся экземплярами этих типов.
  - Иерархия типов связана не только с данными, но и с различными библиотеками и компонентами инструментальных средств, поддерживающими разработку приложений.
  - В ООСУБД, совместимой со стандартом ODMG, хранилище представляет собой интегрированную систему, где согласованным образом сохраняются данные и программный код.

Компоненты архитектуры: инструментальные средства и библиотеки

- Инструментальные средства, поддерживающие, например, разработку пользовательских приложений и их графических интерфейсов, программируются на одном из OML и сохраняются как часть иерархии классов.
- Библиотеки функций доступа, арифметических функций и др. также сохраняются в иерархии типов и являются единообразно доступными из программного кода разработчика приложения.
- Ассортимент инструментальных средств и библиотек в стандарте не определяется.

Объектная модель ODMG

- Модель включает возможность описания как объектов, так и литеральных значений.
- Модель подстраивается под специфику систем баз данных:
  - Для баз данных, схем и подсхем обеспечивается набор встроенных объектных типов.
  - Модель включает ряд встроенных структурных типов, позволяющих применять традиционные методы моделирования баз данных.
  - Модель одновременно включает понятия объектов и литералов.
  - В модели связи между объектами отличаются от атрибутов объектов (как в E/R-модели).

Объекты и литералы

- Объекты обладают свойством идентифицируемости.
  - При обращении к объекту по его ИД для получения доступа к базовым значениям данных накладные расходы могут сильно замедлить работу приложений.
  - В модели ODMG допускается описание данных в терминах объектов и использование традиционных *(литеральных)* значений.
    - Возраст человека может задаваться целочисленным литералом, а не объектом, имеющим свойство *возраст*. Значение возраста будет сохраняться как часть структуры данных объекта *человек*, а не в отдельном объекте.
- Объект может входить в состав нескольких других объектов, а литерал – нет.
- Схема базы данных в модели ODMG состоит из набора объектных типов, но компонентами этих типов могут быть типы литеральных значений.
- Объект идентифицируется своим OID, который отделен от значений компонентов объекта, а литерал идентифицируется значениями своих компонентов.

Объектные и литеральные типы

- Объекты обладают свойствами идентифицируемости и индивидуального существования, а литералы являются компонентами объектов.
- Объектные типы существуют в иерархии объектных типов; литеральные типы похожи на типы, характерные для обычных языков программирования.
- Литеральные типы – базовые скалярные числовые типы, символьные и булевские типы (атомарные литералы), конструируемые типы литеральных записей (структур) и коллекций.
- Конструируемые литеральные типы могут основываться на любом литеральном или объектном типе. Даты и время строятся как литеральные структуры.

Объектный тип

- *Объектный тип* состоит из интерфейса и одной или нескольких реализаций.
- *Интерфейс* описывает внешний вид типа: какими свойствами он обладает, какие в нем доступны операции и каковы параметры у этих операций.
- В *реализации* определяются структуры данных, реализующие свойства, и программный код, реализующий операции.
- Интерфейс составляет общедоступную часть типа, а в реализации при необходимости могут вводиться дополнительные частные свойства и операции.
- Все объектные типы (системные или определяемые пользователем) образуют иерархию типов, в корне которой находится предопределенный объектный тип Object.

Объектный тип

- Интерфейс:
  - *имя*;
  - набор *супертипов*;
  - имя поддерживаемого системой *экстента*;
  - один или более *ключей* для ассоциативного доступа;
  - набор *атрибутов*, каждый из которых может быть объектом или литеральным значением;
  - набор *связей*, каждая из которых указывает на некоторый другой объект;
  - набор *операций*.
- Атрибуты и связи совместно называются *свойствами*, а свойства и операции совместно называются *характеристиками* объектного типа.

Связи

- Связи неявно моделируются как свойства, значениями которых являются объекты.
  - Если человек работает на некоторую компанию, то у каждого объекта-человека должно иметься свойство, которое можно назвать worksFor и значением которого является соответствующий объект-компания.
- При определении связи должна быть определена ее инверсия.
  - Если worksFor определяется как связь, должно быть явно указано, что инверсией является свойство employees (множество всех объектов-служащих данной компании) объекта-компании, а при определении свойства employees должна быть указана инверсия worksFor .
- Система баз данных должна поддерживать согласованность связанных данных, что позволяет сократить объем работы программистов приложений и повысить надежность их программ.

Язык определения объектов ODL

- ODL используется для описания интерфейсов типов приложения, а не для программирования реализации.
- Описание на языке ODL – это набор определений типов, констант, исключительных ситуаций, интерфейсов типов и модулей.

Типы в ODL

- Атомарные
  - integer, float, character, string, boolean, enum
- Составные (Объектные и литеральные)
  - Struct и struct { тип имя, тип имя, … } – структура
  - Коллекции
    - Set и set <тип> – множество
    - Bag и bag <тип> – мультимножество
    - List и list <тип> – список
    - Array и array <тип, количество> – массив
    - Dictionary и dictionary <типКлюча, типЗначения> – словарь
- Ограничения на типы связей
  - не разрешается использовать атомарный тип: Set <integer>
  - не разрешается использовать структуры: Struct { Supplier s, Part p }
  - не разрешается применять конструктор типа-коллекции более одного раза: Set <Array<Part, 100>>

Описания в ODL: атрибуты


```
class Supplier {
    attribute string name;
    attribute struct addrType  {
        string street, string city, string zip } address;
    attribute float rating; };
class Part {
    attribute string name;
    attribute enum colorType { red, white, green } color;
    attribute Supplier::addrType address;
    attribute float price; };
```

Описания в ODL: связи


```
class Supplier {
    attribute string name;
    attribute Struct addrType  {
        string street, string city, string zip } address;
    attribute integer rating;
    relationship Set <Part> supplierOf; };
class Part {
    attribute string name;
    attribute enum colorType { red, white, green } color;
    attribute Supplier::addrType address;
    attribute float price;
    relationship Set <Supplier> suppliedBy; };
```

Описания в ODL: обратные связи


```
class Supplier {
    attribute string name;
    attribute Struct addrType  {
        string street, string city, string zip } address;
    attribute integer rating;
    relationship Set <Part> supplierOf          inverse Part::suppliedBy; };
class Part {
    attribute string name;
    attribute enum colorType { red, white, green } color;
    attribute Supplier::addrType address;
    attribute float price;
    relationship Set <Supplier> suppliedBy inverse Supplier::supplierOf; };
```

Связи в ODL

E/R связь	ODL связь	
	C1	C2
1:1	relationship <C2> relToC2	relationship <C1> relToC1
1:M	relationship ***collectType*** <C2> relToC2	relationship <C1> relToC1
M:1	relationship <C2> relToC2	relationship ***collectType*** <C1> relToC1
M:N	relationship ***collectType*** <C2> relToC2	relationship ***collectType*** <C1> relToC1

collectType – Set, List, Bag, Array

Описания в ODL: многосторонние связи


```
class Supplement {
    attribute integer qty;
    relationship Supplier theSupplier inverse Supplier::supplierFor;
    relationship Part  thePart  inverse Part::partFor; };
class Supplier {
    …
    relationship Set <Supplement> supplierFor inverse Supplement::theSupplier ; };
class Part {
    …
    relationship Set <Supplement> partFor inverse Supplement::thePart ; };
```

Описания в ODL: операции

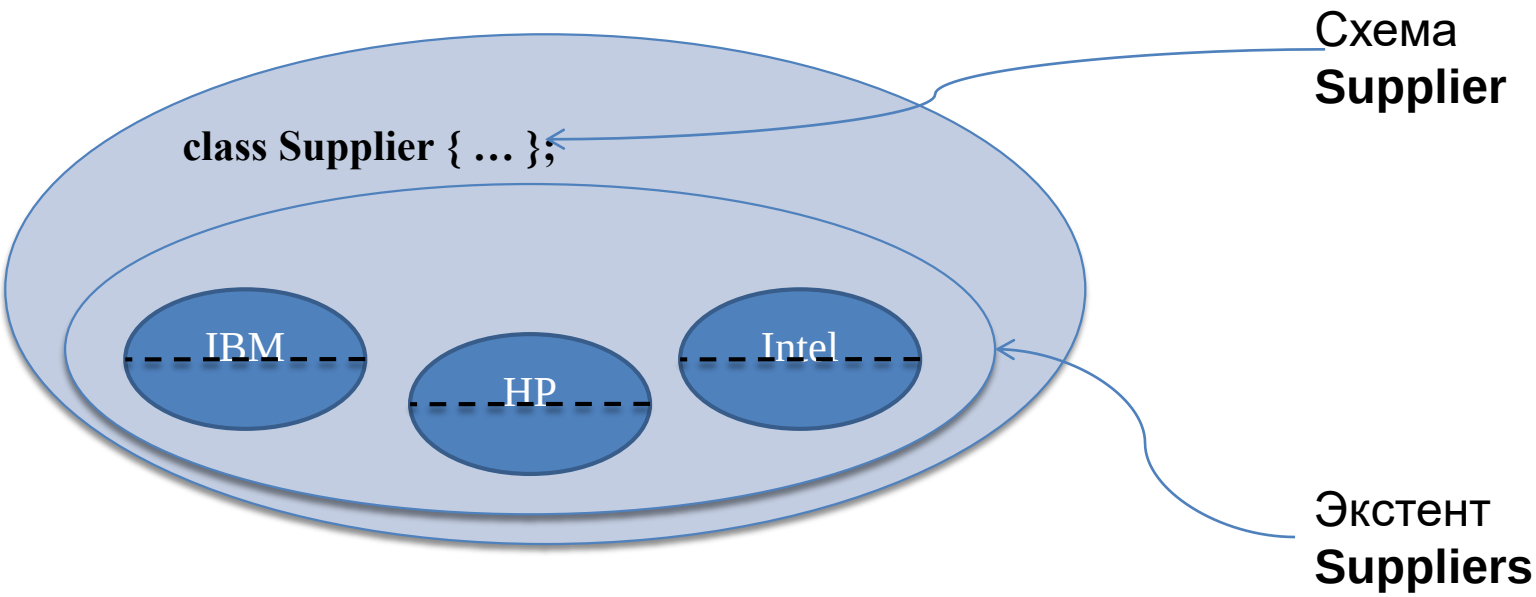

```
class Supplier {
    attribute string name;
    attribute Struct addrType  {
        string street, string city, string zip } address;
    attribute integer rating;
    relationship Set <Part> supplierOf
        inverse Part::suppliedBy;
    integer totalPartsQty() raises (noSuppliedParts);
    integer suppliedPartsQty(in Set <string>) raises (noSuppliedParts);
    void suppliedPartNames(out Set <string>) raises (noSuppliedParts); };
```

Описания в ODL: наследование

- Обычное наследование
  - class Ancestor { атрибуты, связи, методы };
  - class Descendant extends Ancestor {
    новые атрибуты, связи, методы };
- Множественное наследование
  - class Ancestor1 { … };
  - class Ancestor2 { … };
  - class Descendant extends Ancestor1 : Ancestor2 {
    новые атрибуты, связи, методы };
  - Возможны конфликты имен атрибутов, связей, методов, наследуемых от разных классов.

Описания в ODL: экстент

- Экстент объектного типа – множество существующих объектов данного типа.
- class Supplier (extent Suppliers) { … };



Описания в ODL: ключи

- *Ключ* – уникальный атрибут (атрибуты) класса.
  - Не заменяет (дополняет) OID.
  - Аналог потенциального ключа отношения в реляционной модели.
- Примеры:
  - class Supplier (extent Suppliers
    key (name)) {…};
  - class Supplement (extent Supplements
    key (theSupplier, thePart)) {…};
  - class Employee (extent Employees
    key (empID), (ssNo)) {…};

Язык объектных запросов OQL

- OQL опирается на объектную модель ODMG.
- OQL близок к SQL/92. Расширения относятся к ОО-понятиям (сложные объекты, ОИД, выражения пути, полиморфизм, вызов операций, отложенное связывание).
- В OQL обеспечиваются высокоуровневые примитивы для работы с множествами объектов, а также для работы со структурами, списками и массивами.
- Результат любого запроса OQL обладает типом, принадлежащим к модели типов ODMG, и поэтому к результату запроса может быть применен новый запрос.
- OQL не является вычислительно полным языком, он представляет собой простой язык запросов.
- Операторы языка OQL могут вызываться из любого языка программирования, для которого в стандарте ODMG определены правила связывания. И наоборот, в запросах OQL могут присутствовать вызовы операций, запрограммированных на этих языках.
- В OQL не определяются явные операции обновления, а используются вызовы операций, определенных в объектах для целей обновления.
- В OQL обеспечивается декларативный доступ к объектам. По этой причине OQL -запросы могут быть легко оптимизированы.

OQL: примеры запросов

- Найти даты рождения служащих, зарплата которых превышает 20000 руб.
  - select distinct E.birthDate
    from employees E
    where E.salary > 20000
  - Результатом запроса будет литеральное значение типа set <date>, т.е. литеральное значение-множество, элементами которого являются значения типа date.

OQL: примеры запросов

- Найти имена и даты рождения служащих, зарплата которых превышает 20000 руб.
  - select distinct struct (name: E. name, birth: E.birthDate)
    from employees E
    where E.salary > 20000
  - Результатом запроса является литеральное значение-множество, элементами которого являются значения типа struct { string <20> name; date birth }.

OQL: примеры запросов

- Найти имена и даты рождения служащих, зарплата которых превышает 20000 руб.
  - в классе EMP определена операция age (возраст соответствующего служащего)
  - select distinct struct (name: E.name, age: E.age)
    from employees E
    where E.salary > 20000.00
  - Результатом запроса является литеральное значение-множество, элементами которого являются значения типа struct { string <20> name; interval age }

OQL: примеры запросов

- Получить номера начальников отделов и тех сотрудников их отделов, зарплата которых превышает 20000 руб.
  - select distinct struct (mgr: D.mgr, dhs:
    (select E
    from d.consistsOf as E
    where E.salary > 20000))
    from departments D
  - Запрос похож на SQL-запрос с вложенным подзапросом, но это только внешнее сходство!
    - В разделе FROM "подзапроса" имеется переход по связи consistsOf от экземпляра объектного типа DEPT ко множеству экземпляров объектного типа EMP.
    - Результат запроса имеет тип set < struct { integer mgr ; bag < EMP > DHS } >.

OQL: примеры запросов

- Хранимые в базе данных объекты могут иметь индивидуальные имена.
  - Пусть в базе данных сохраняется объект типа DEPT с именем mainDepartment
  - mainDepartment
    - Результатом запроса будет соответствующий объект
  - mainDepartment.consistsOf
    - Результатом запроса будет литеральное значение-множество, состоящее из объектов типа EMP, ассоциированных через связь consistsOf с объектом mainDepartment .
- Имя экстента можно трактовать как имя объекта-множества.
  - EMPLOYEES
    - Результатом будет литеральное множество, состоящее из всех объектов, которые содержаться в указанном экстенте.

OQL: примеры запросов

- Пусть имеются определения объектных типов:
  - typedef Set <interval> ages;
  - class persInfo {
    attribute string <20> n;
    attribute date b; };
  - typedef Bag <persInfo > info;
- ages (select distinct E.age
  from employees E
  where E.salary > 20000)
  - Результатом запроса является объект-множество, включающий литеральные значения типа interval.
- info (select persinfo (n: E.name, b: E.birthDate)
  from employees E
  where E.salary > 20000)
  - По литеральным значениям имени и даты рождения каждого служащего, размер зарплаты которого превышает 20000 руб., конструируется объект типа persInfo, а на основе литерального мультимножества этих объектов конструируется объект-мультимножество типа info .

OQL: примеры запросов

- Пусть имеются определения объектных классов
  - class DEPT {
...
relationship EMP managedBy
inverse EMP :: managerOf
}
  - class EMP {
...
relationship DEPT managerOf
inverse DEPT :: managedBy
}
- mainDepartment.managedBy
  - Результатом запроса будет объект типа EMP.
- mainDepartment.managedBy.name
  - Результатом запроса будет значение типа string.
- Получить имена всех сотрудников отдела
  - mainDepartment.consistsOf.name – неверно (для связи “один-ко-многим” такие путевые выражения в OQL не допускаются)
  - select E.name from mainDepartment.consistsOf as E

OQL: примеры запросов

- Получить имена служащих и номера их отделов для служащих, работающих в отделах с фондом заработной платы, размер которого превышает 1000000 руб.
  - select struct (name: E.name, dept: D.deptNo)
    from employees E, E.worksAt D
    where E.worksAt <>NIL andthen
    D.deptTotalSalary > 1000000

OQL: примеры запросов

- Выбрать всех служащих, у которых имеются однофамильцы.
  - select distinct E
    from employees E, employees E1
    where E <> E1 and E.name = E1.name

OQL: примеры запросов

- Выбрать всех служащих, работающих в отделе.
  - select distinct E
    from employees E
    where is_defined(E.worksAt.deptNo)
  - select distinct E
    from employees E
    where E.worksAt <> NIL

OQL: примеры запросов

- Найти название отдела, в котором работают служащие, средняя зарплата которых меньше средней зарплаты служащих любого другого отдела.
  1. Построить множество объектов-служащих, для которых известен номер отдела
     - define employeesD () as
       select e from employees E
       where E.works is not NIL
  2. Сгруппировать служащих с известными номерами отделов по номерам отделов и вычислить размер средней зарплаты для каждого отдела
     - define deptAvgSal () as
       select deptNo, avgSal : avg (select X.E.salary from partition X)
       from employeesD() E
       group by deptNo : E.deptNo
  3. Отсортировать полученное мультимножество по значениям атрибута avg Sal
     - define sortedDeptAvgSal() as
       select S from deptAvgSal () as S order by S.avgSal
  4. Выбрать значение атрибута deptNo из элемента списка sortedDeptAvgSal с наименьшим значением avgSal (этот элемент стоит в списке первым)
     - first (sortedDeptAvgSal ()).deptNo

Критика стандарта ODMG

- Объектная модель недостаточно обоснована.
  - В модель стремились поместить свойства реализационных моделей всех коммерческих ООСУБД, существовавших к моменту написания стандарта.
- Разделение языков ODL, OCL и OML затрудняет целостное понимание модели.
- OQL очень сложен для использования непрограммистами.

Заключение

- Рассмотрена архитектура ODMG для ОО систем баз данных
- Рассмотрен язык описания ODL
- Рассмотрен язык запросов OQL