Базы данных

Модуль 3. Перспективные направления развития систем обработки данных. Обзор современных СУБД

Лекция 15

Объектные и распределенные СУБД.

Преподаватель:
Семенов Геннадий Николаевич, к.т.н., доцент

2023

Объектно-ориентированные базы данных (ООБД)

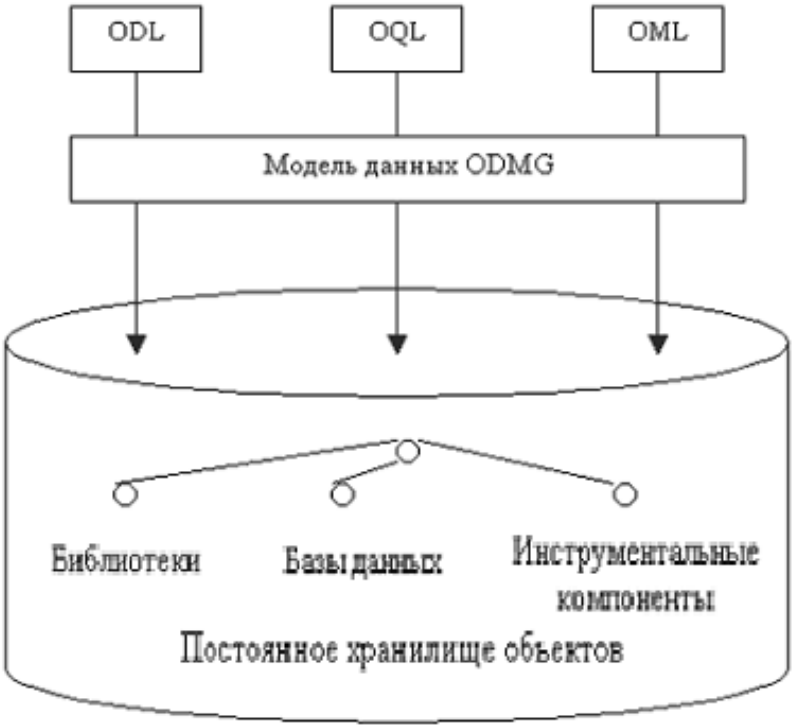
История

- ✓ Конец 80-х гг. – образование промышленных компаний и рынка систем управления объектными базами данных (СУОБД).
- Области применения: САПР, промышленность программного обеспечения, географические информационные системы, финансовая сфера, медицин… …ющие информ…
- Лето 19… …о (ODMG … консорц… …ных участни…
- Конец 1… объектн… В состав… …Object Design, … …ены ODMG …

Архитектура СУОБД, согласно ODMG-93

ODL – ***Object Definition Language (язык определения объектов)*;**
OQL – ***Object Query Language (язык объектных запросов);***
OML – ***Object Manipulation Language (язык манипулирования объектами).***

- *Объектная модель данных.* Все данные, сохраняемые СУОБД, структуризуются в терминах конструкций модели данных.
- *Постоянное хранилище объектов.* Логическая организация хранилища данных любой СУОБД, совместимой со стандартном ODMG , должна основываться на модели данных ODMG



Архитектура Систем Управления ООБД

Инструментальные средства и библиотеки.

- Инструментальные средства, поддерживающие разработку пользовательских приложений и их графических интерфейсов, программируются на одном из OML и сохраняются как часть иерархии классов.
- Библиотеки функций доступа, арифметических функций и другие сохраняются в иерархии типов и являются доступными из программного кода приложения.

Язык определения данных (ODL). Схемы баз данных описываются в терминах языка ODL, в котором конструкции модели данных конкретизируются в форме языка определения (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).

- ***Язык объектных запросов (ODL).*** Язык 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.
- ***Языки манипулирования объектами (OML).*** Для программирования реализаций операций и приложений требуется наличия объектно-ориентированного языка программирования. OML представляется собой интегрирование языка программирования с моделью ODMG; это интегрирование производится за счет определенных в стандарте правил ***языкового связывания:*** в самих языках программирования, не поддерживается стабильность объектов. Чтобы разрешить программам на этих языках обращаться к хранимым данным, языки должны быть расширены дополнительными конструкциями или библиотечными элементами.

Введение в объектную модель ODMG

•Модель ODMG – объектная модель данных, включающая возможность описания как объектов, так и литеральных значений.

•Специфика для 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 (аналогично тому, как это делается в ER -модели).

Объекты и литералы:

Отличие *объекта* от значения: **1-е -** наличие у объекта уникального идентификатора – **OID** (объекты обладают свойством идентифицируемости).

Недостаток: накладные расходы, требуемые для обращения к объекту по его идентификатору с целью получения доступа к базовым значениям данных, *могут сильно замедлить работу приложений*.

•Допускается описание всех данных в терминах объектов и использование традиционного типа значений (***литеральные значения***).

•Объект может входить в состав нескольких других объектов, а литерал – нет.

•Схема базы данных главным образом состоит из набора объектных типов, но компонентами этих типов могут быть типы литеральных значений.

Объекты и литералы:

2-е отличие объектов и литералов – понятие ***изменчивости***.
Предположим, например, что данные о человеке составляют структуру:

<имя, возраст, адрес_проживания>

Тогда возможны два варианта хранения этих данных:

1. Если человек представляется в виде объекта, то компоненты описывающей его структуры данных могут изменяться (например, может изменяться адрес), но объект (человек) остается тем же самым (поскольку объектный идентификатор не изменяется). (объекты обладают свойством изменчивости)
2. Если же данные о человеке сохраняются в виде литеральной структуры, и один из компонентов этой структуры изменяется, то вся структура трактуется как новое значение. Если данные о человеке не должны изменяться, то не может изменяться ни один элемент структуры, и она является ***неизменчивым литералом***.

Связи и атрибуты

Связи являются неотъемлемой характеристикой предметных областей.

- В большинстве объектных систем связи неявно моделируются как свойства, значениями которых являются объекты.
- В модели ODMG , подобно ER-модели, различаются два вида свойств – атрибуты и связи.
- ***Атрибутами*** называются свойства объекта, значение которых можно получить по OID объекта, но не наоборот. Значениями атрибутов могут быть и литералы, и объекты, но только тогда, когда не требуется обратная ссылка.
- ***Связи*** – это инверсные свойства. В этом случае значением свойства может быть только объект, поскольку литеральные значения не обладают свойствами. Поэтому возраст служащего обычно моделируется как атрибут, а компания, в которой работает служащий, – как связь.
- При определении связи должна быть определена ее инверсия. После этого система баз данных должна поддерживать согласованность связанных данных, что позволяет сократить объем работы программистов приложений и повысить надежность их программ.

Объектные и литеральные типы

- В модели ODMG база данных представляет собой коллекцию *различимых значений (denotable values)* двух видов – объекты и литералы.
- Модель данных содержит конструкции для спецификации объектных и литеральных типов.
- Объектные типы существуют в иерархии объектных типов; литеральные типы похожи на типы, характерные для обычных языков программирования (например, C или Pascal).
- В модели ODMG не используется термин *класс*. Единственная классификационная конструкция называется *типом*, и типы описывают как объекты, так и литералы.

Литеральные типы:

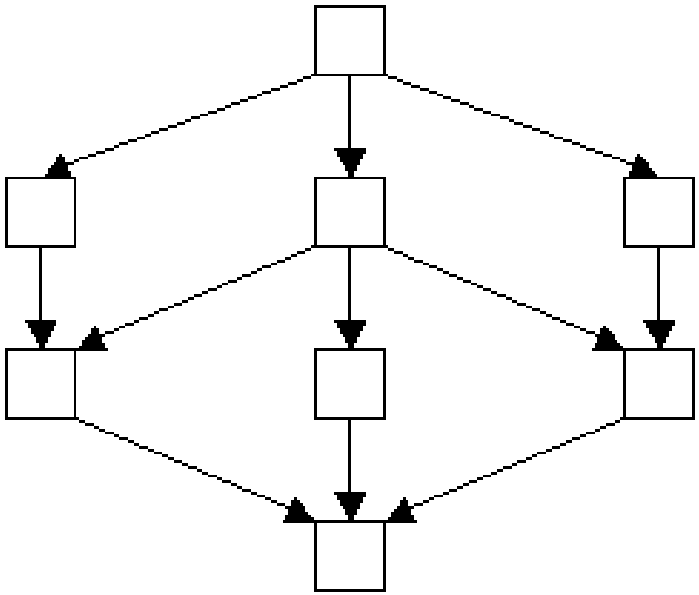
базовые скалярные числовые типы, символьные и булевские типы (атомарные литералы), конструируемые типы литеральных записей (структур) и коллекций.

Конструируемые литеральные типы могут основываться на любом литеральном или объектном типе, но считаются неизменчивыми.

Даты и время строятся как литеральные структуры.

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 (*public*) часть типа, а в реализации при необходимости могут вводиться дополнительные частные (*private*) свойства и операции.
- Все объектные типы (системные или определяемые пользователем) образуют решетку (*lattice*) типов, в корне которой находится предопределенный объектный тип Object



Объектный тип

Интерфейс объектного типа состоит их следующих компонентов:

- *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.

Точно так же, как имеются атомарные и конструируемые литеральные типы, существуют атомарные и конструируемые объектные типы.

В стандарте ODMG 3.0 говорится, что атомарными объектными типа являются только типы, определяемые пользователями.

Конструируемые объектные типы включают структурные типы и набор типов коллекций.

Объектный тип

Определение

- Объектный тип можно определить с помощью двух разных синтаксических конструкций языка ODL – **interface** и **class**.
- Определение класса отличается от определения интерфейса наличием двух необязательных разделов: **extends** и **type_property_lis**t . Наиболее важным отличием класса от интерфейса является возможность наличия второго из этих разделов.
- В списке свойств могут присутствовать элементы **extent** и **key**. Для каждого класса может быть определен только один экстент, являющийся множеством всех объектов этого класса, которые после создания должны сохраняться в БД.
- Ключ – это набор свойств объектного класса, однозначно идентифицирующий состояние каждого объекта, входящего в экстент класса (это аналог возможного ключа реляционной модели данных).
- Для класса может быть объявлено несколько ключей, а может не быть объявлено ни одного ключа даже при наличии определения экстента. В последнем случае в экстенте класса могут существовать разные объекты с одним и тем же состоянием.

Язык определения объектов ODL

- ODL – это язык определения данных для объектной модели ODMG.
- ODL используется исключительно для описания интерфейсов типов приложения (язык описания схем баз данных), а не для программирования реализации.
- «Программа» на языке ODL – это набор определений типов, констант, исключительных ситуаций, интерфейсов типов и модулей.
- Язык обеспечивает широкие возможности для определения литеральных типов.
- Типы коллекций. Можно определить четыре разновидности типов коллекций:
  - Типы категории **set** – это обычные типы множеств.
  - Типы категории **bag** – эти типы мультимножеств (в значениях которых допускается наличие элементов-дубликатов).
  - Значениями типов категории **list** являются упорядоченные списки значений (среди них допускаются дубликаты).
  - Значениями типы **dictionary** являются множества пар <ключ, значение> , причем все ключи в этих парах должны быть различными.

Определения типов имеют рекурсивную природу.

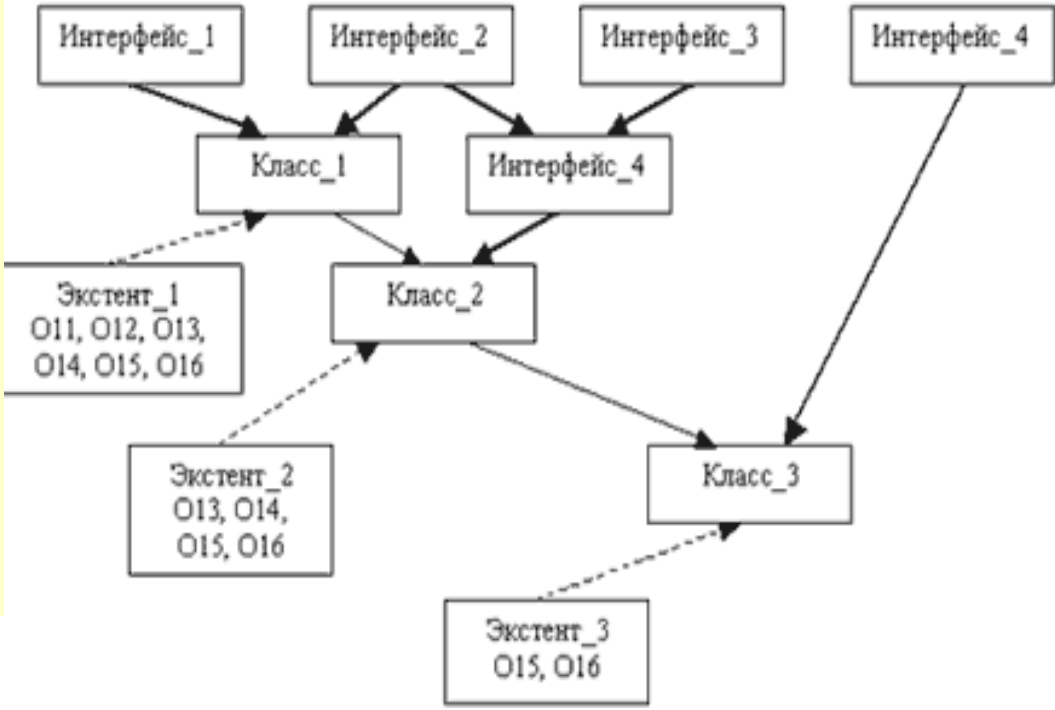
Объектный тип

- Допускается создание объектов только тех объектных типов, которые определены как классы.
- Объектные типы, определенные как интерфейсы, могут использоваться только для порождения новых объектных типов (интерфейсов и классов) на основании (вообще говоря) множественного наследования.
- Классы могут определяться на основе множественного наследования интерфейсов и одиночного наследования классов.
- Стабильность – это свойство объекта сохранять состояние между сеансами работы программы. Объектная база данных – это, по существу, хранилище стабильных объектов.
  - ★ Достигается это, в зависимости от реализации, либо введением в язык программирования нового ключевого слова, либо предоставлением специального метода для создания объектов, либо наследованием от специального предопределенного типа. Программисту, таким образом, достаточно объявить объект "стабильным", а далее СУБД берет на себя черновую работу по отслеживанию изменений, отмене ссылок на удаленные объекты, созданию версий объектов.

Наследование

Механизм наследования от интерфейсов называется в ODMG наследованием *IS - A* (жирные стрелки), а механизм наследования от классов – *extends* (обычные).

В модели ODMG поддерживается *семантика включения*, (экстент любого подкласса является подмножеством экстента любого своего суперкласса прямого или косвенного).
Стандарт не требует обязательного определения экстента.
В этом случае ответственность на поддержку работы с множествами объектов ложится на прикладного программиста.



Краткая характеристика языка запросов OQL

- OQL опирается на объектную модель ODMG.
- OQL очень близок к SQL/92. Расширения относятся к сложным объектам, объектным идентификаторам, путевым выражениям, полиморфизму, вызову операций и отложенному связыванию.
- В OQL обеспечиваются высокоуровневые примитивы для работы с множествами объектов, а также примитивы для работы со структурами, списками и массивами.
- 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. 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.

Определение связей

Связи явно определяются путем указания *путей обхода (traversal paths)*.

Пути обхода указываются парами, по одному пути для каждого направления обхода связи.

Пример: в базе данных “служащие-отделы-проекты” служащий работает (works) в одном отделе, а отдел состоит (consists of) множества служащих. Тогда путь обхода consists_of должен быть определен в объектном типе DEPT, а путь обхода works – в типе EMP. Тот факт, что пути обхода относятся к одной связи, указывается в разделе inverse обоих объявлений пути обхода.

Определения типов DEPT и EMP могли бы выглядеть следующим образом:

```
class DEPT {      ...
      relationship set <EMP> consists_of
          inverse EMP :: works
        ... }
class EMP {      ...
       relationship DEPT works
           inverse DEPT :: consists_of
         ... }
```

Объекты как результаты запросов:

- коллекция объектов;
- индивидуальный объект;
- коллекция литеральных значений;
- индивидуальное литеральное значение.

Вызовы методов

- В OQL допускается вызов метода объекта с указанием действительных параметров или без их указания (в зависимости от сигнатуры метода объекта, или класса) в любом месте запроса, если тип результата вызова метода соответствует типу ожидаемого результата запроса.
- Если у метода отсутствуют параметры, то нотация вызова метода в точности совпадает с нотацией выбора атрибута или перехода по связи.
- Если у метода имеются параметры, то их действительные значения задаются через запятую в круглых скобках.
- Если в классе (или интерфейсе) содержатся одноименные атрибут и метод без параметров, то для разрешения конфликта имен в вызове метода должны содержаться круглые скобки.
- Пусть в классе EMP определена операция HIGHER_PAID , которая для данного объекта-служащего производит множество объектов-служащих того же отдела с большим размером зарплаты. Тогда возможен следующий запрос:

```
SELECT DISTINCT E.HIGHER_PAID.participate_in.PRO_NAME
        FROM EMPLOYEES AS E
                WHERE E.EMP_NO = 632
```

Объектно-ориентированные системы

- **GemStone** корпорации GemStone Systems, Inc. была одной из первых коммерчески доступных; традиционно сосредоточена на рынке Smalltalk; была выпущена версия для C и C++. Система основана на трехзвенной архитектуре клиент-сервер, в которой прикладная обработка данных производится на среднем уровне между процессом взаимодействия с пользователем и процессом, поддерживающим хранилище данных.
- **Objectivity/DB** компании **Objectivity, Inc.** подходит для приложений, которые работают в распределенных средах, требуют гибкой модификации данных, организации сложных связей, а также нуждаются в высокой производительности и работы с большими объемами данных.
- СУБД **ONTOS** занимается развитием сервера объектно-ориентированной СУБД, но в последнее время придает особое значение своим Службам Интеграции Объектов (Object Integration Services).
- **OpenODB** фирмы **Hewlett-Packard** существует в версии для C++. Благодаря глубокой интеграции, SoftBench распознает файлы приложений OpenODB для установки оптимальной конфигурации, может создавать базы данных формата OpenODB из своей интегрированной среды, обеспечивает оперативную помощь из среды разработки и т. д.
- СУБД **ObjectStore** занимает лидирующее положение в отрасли, осуществляя около 33% поставок на рынке объектно-ориентированных СУБД, поддерживает версии своей СУБД как для C++, так и для Smalltalk

Объектно-ориентированные системы

- ***Jasmine*** наиболее известная объектная СУБД. Будучи одной из новейших объектных баз данных, вобрала в себя опыт предыдущих работ, добавив много нужных для объектного программирования новшеств. Как коммерческий продукт, пожалуй, имеет наибольший потенциал на рынке СУБД, снабжена визуальной средой разработки Jasmine Studio, которая отличается удобным, продуманным и интуитивно понятным интерфейсом
- ***Versant***. С использованием Versant реализован целый ряд крупных проектов (в первую очередь в области телекоммуникаций).
- ***O2*** живая "классика" среди объектных баз данных. Чаще используется в области телекоммуникаций и систем специального применения.
- **Российская объектная СУБД - ODB-Jupiter**
- Это оригинальная разработка научно-производственного центра "Интелтек Плюс". По сравнению с приведенными выше системами управления базами данных эта СУБД находит более широкое применение в России.
- **Matisse** - ориентирована на большие базы данных с богатой семантической структурой и может манипулировать с очень большими объектами, такими как изображения, фильмы и звуки. Хотя она поддерживает основные понятия ООБД, в частности, множественное наследование, но не налагает сильных ограничений на модель данных, а скорее служит мощной машиной ООБД.
- **Cache** - работает в операционных системах семейства Windows, Linux, хранит данные в виде многомерных разреженных массивов– глобалов. Уникальная многомерная модель Cache позволяет избежать проблем, присущих реляционным СУБД, оптимизируя доступ к данным уже на уровне их хранения.