Объектно-ориентированные базы данных

История вопроса

Начало пути:

- 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, медицинские применения, телекоммуникации, мультимедиа, управляющие информационные системы.
- Лето 1991 г. – образование в США Object Database Management Group (ODMG) – Группы Управления Объектными Базами Данных – как консорциума фирм-производителей СУОБД и других заинтересованных участников для разработки стандарта СУОБД.
- Конец 1993 г. – группа завершила работу опубликованием Стандарта объектных баз данных ODMG-93.
- В состав ODMG во время разработки стандарта ODMG-93 входили: Object Design, Objectivity, Ontos, O2 Technology, SunSoft, Versant. Фирмы-члены ODMG представляли 90% рынка СУОБД.

Постулат того времени: новая парадигма в корне изменит способы проектирования и разработки приложений баз данных!

К разговору об определениях

- Что такое ООСУБД?
- Что такое объектно-ориентированная база данных?

Объектный мир определен в целом настолько расплывчато и нечетко, что невозможно однозначно говорить об основных объектных возможностях.

Сегодня под ООСУБД следует понимать системы, которые следуют духу *Манифеста систем баз данных третьего поколения* и букве стандарта ODMG:1993.

В качестве примеров реализаций ООСУБД можно привести:

GemStone (компания GemStone Systems Inc. (http://www.gemstone.com/).

ITASCA (http://www.ibex.ch).

ObjectStore (компания Progress Software (http://www.objectstore.net).

Objectivity /DB (Компания Objectivity (http://www.objectivity.com).

Versant (компания Versant (http ://www .versant .com /).

Объектно-ориентированные базы данных

Манифест объекно-ориентированных баз данных

- **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 ООСУБД, структуризуются в терминах конструкций модели данных.
- *Постоянное хранилище объектов.* Логическая организация хранилища данных любой ООСУБД, совместимой со стандартном ODMG , должна основываться на модели данных ODMG .

Архитектура СУОБД

- *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 т.д. также сохраняются в иерархии типов и являются единообразно доступными из программного кода разработчика приложения. Ассортимент инструментальных средств и библиотек в стандарте не определяется.
- *Язык определения данных (ODL).* Схемы баз данных описываются в терминах языка ODL, в котором конструкции модели данных конкретизируются в форме языка определения. 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. Кроме того, ODL является *виртуальным* языком в том смысле, что в стандарте ODMG не требуется его реализация в программных продуктах ООСУБД.

Архитектура СУОБД

- *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 (language binding).* Дело в том, что в самих языках программирования, естественно, 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.

Введение в объектную модель ODMG

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

На разработку модели повлиял тот факт, что она предназначена для поддержки работы с базами данных, так что особо важной является эффективность доступа к данным.

Модель ODMG подстраивается под специфику систем баз данных следующим образом:

- ✔ 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).

Введение в объектную модель ODMG

Объекты и литералы

Первое отличие объекта от значения: наличие у объекта уникального идентификатора – OID, Object Identifier (объекты обладают свойством идентифицируемости). Недостаток: накладные расходы, требуемые для обращения к объекту по его идентификатору с целью получения доступа к базовым значениям данных, которые могут весьма сильно замедлить работу приложений.

В модели ODMG допускается описание всех данных в терминах объектов и использование традиционного вида значений, которые в модели называются *литеральными значениями*.

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

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

Введение в объектную модель ODMG

Объекты и литералы

Второе отличие объектов и литералов – понятие ***изменчивости*** *(mutability)*.

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). Тем самым, 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***.

Введение в объектную модель ODMG

Связи

- ✔ 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.

Введение в объектную модель ODMG

Связи, пример

Например, 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. Возникает проблема, если у объекта-компании имеется свойство, которое затрагивает множество служащих этой компании (например, employees – множество, включающее все объекты служащих данной компании). Эти два свойства являются несвязными, и поддержка их согласованности может вызывать значительную программистскую проблему.

Если worksFor определяется как связь, должно быть явно указано, что инверсией является свойство employees объекта-компании, а при определении employees должна быть указана инверсия worksFor. После этого система баз данных должна поддерживать согласованность связанных данных, что позволяет сократить объем работы программистов приложений и повысить надежность их программ. Если в объекте-компании свойство employees не требуется, то свойство объекта-служащего employees может быть атрибутом.

Введение в объектную модель ODMG

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.

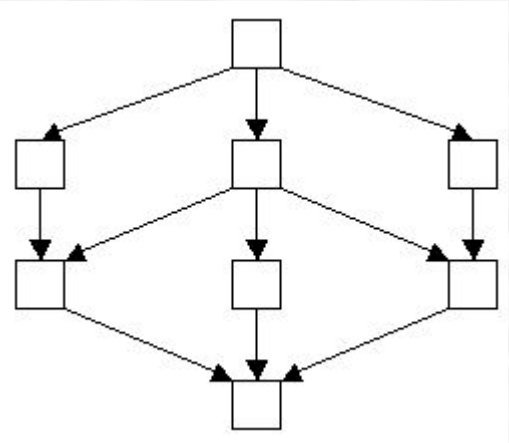
Литеральные типы:

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.

Введение в объектную модель ODMG

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.



Введение в объектную модель ODMG

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.

Введение в объектную модель ODMG

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.

Введение в объектную модель ODMG

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.

Введение в объектную модель ODMG

Объектный тип. Определение

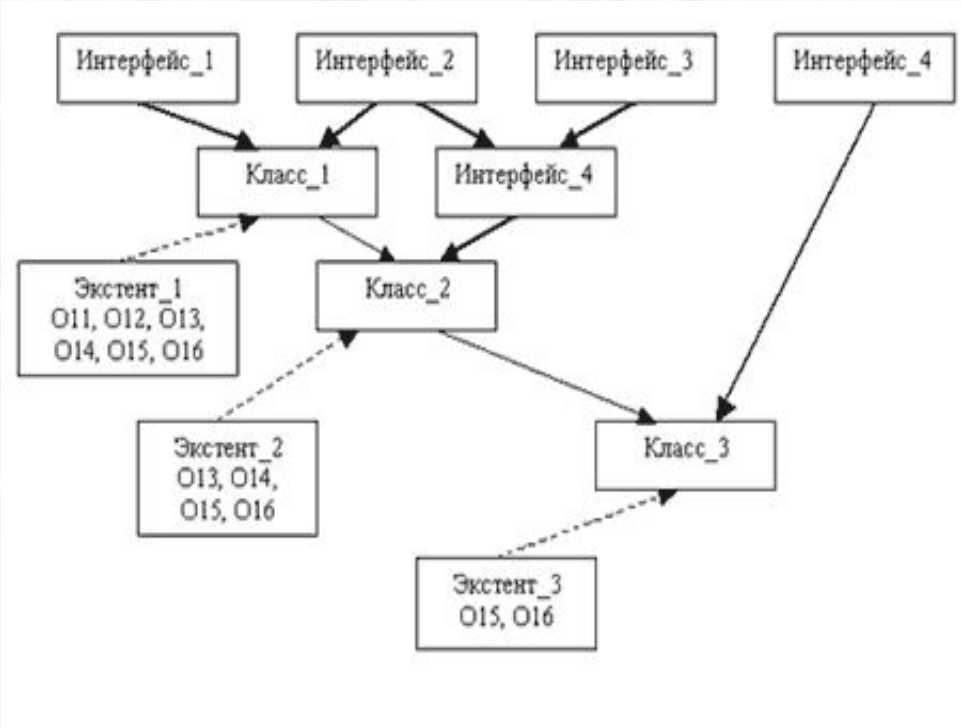
- Допускается создание объектов только тех объектных типов, которые определены как классы.
- Объектные типы, определенные как интерфейсы, могут использоваться только для порождения новых объектных типов (интерфейсов и классов) на основании (вообще говоря) множественного наследования.
- Классы могут определяться 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.

Введение в объектную модель ODMG

Объектный тип. 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*, означающая, что 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.



Введение в объектную модель ODMG

Объектный тип. 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
        ... }
```

Введение в объектную модель ODMG

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 является функциональным языком, допускающим неограниченную компонируемость операций, т.к. результат любого запроса обладает типом, принадлежащим к ODMG.

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.

Введение в объектную модель ODMG

Краткая характеристика языка запросов OQL

“OQL-запрос является функцией, вырабатывающей объект, тип которого может быть выведен на основе операций, которые участвуют в выражении запроса”.

Пример 1. Найти даты рождения служащих, зарплата которых превышает 30000 р.

```
SELECT DISTINCT E.EMP_BDATE
        FROM EMPLOYEES E
        WHERE E.EMP_SAL > 30000.00
```

Пример 2. Найти имена и даты рождения служащих с зарплатой выше 30000 руб.

```
SELECT DISTINCT STRUCT (name: E.EMP_NAME, bdate: E.EMP_BDATE)
        FROM EMPLOYEES E
       WHERE E.EMP_SAL > 30000.00
```

Пример 3. Получить номера менеджеров отделов и тех сотрудников их отделов, зарплата которых превышает 30000 руб.

```
SELECT DISTINCT STRUCT ( mng: D.DEPT_MNG,
                         DHS: ( SELECT   E   FROM D.CONSISTS_OF AS E
              WHERE E.EMP_SAL > 30000.00 ) )
     FROM  DEPARTMENTS  D
```

Введение в объектную модель ODMG

Объекты как результаты запросов

Введем следующие определения объектных типов:

```
typedef Set <interval> ages;
class persinfo {
        attribute string <20> n;
        attribute date b; };
typedef Bag <persinfo > info;
```

Тогда можно сформулировать следующие запросы.

Пример 4:

```
ages ( SELECT DISTINCT E.age
            FROM EMPLOYEES E
            WHERE E.EMP_SAL > 20000.00 )
```

Окончательным результатом этого запроса является объект-множество, включающий литеральные значения типа interval.

Пример 5:

```
info ( SELECT persinfo (n: E.EMP_NAME, b: E.EMP_BDATE)
          FROM EMPLOYEES E
          WHERE E.EMP_SAL > 20000.00 )
```

Введение в объектную модель ODMG

Объекты как результаты запросов

В совокупности результатом допустимых в OQL выражений запросов могут являться:

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

Неопределенные значения

- Результатом выборки свойства объекта с пустым идентификатором является специальное значение UNDEFINED.
- Результатом операции is_undefined(undefined) является логическое значение *true*. Результатом операции is_defined(undefined) является значение *false*.
- Если при вычислении предиката раздела WHERE OQL-запроса получается значение UNDEFINED, то оно трактуется как *false*.
- UNDEFINED является допустимым элементом любой явно или неявно конструируемой коллекции или агрегатной функции COUNT.

Результатом любой другой операции, включающей хотя бы один операнд со значением UNDEFINED, является UNDEFINED.

Введение в объектную модель ODMG

Путевые выражения

Для навигации между подобъектами сложных объектов и по связям между разными объектами в OQL поддерживается специальный вид выражений, называемых *путевыми (path expression)*. Эти выражения определяются в так называемой *точечной* нотации (вместо операции “.” можно использовать и операцию “->”).

Пример. Пусть в классах DEPT и EMP , кроме связи CONSISTS_OF – WORKS определена еще и связь “один-к-одному” MANAGED_BY – MANAGER_OF :

```
class DEPT { ...
      relationship EMP managed_by
      inverse EMP :: manager_of
      ... }
class EMP { ...
      relationship DEPT manager_of
      inverse DEPT :: managed_by
      ... }
```

Введение в объектную модель ODMG

Путевые выражения

Пример 6. Для получения объекта типа EMP , соответствующего менеджеру отдела main_department достаточно написать выражение

main_department.managed_by

Если же требуется получить имя менеджера отдела main_department, то можно воспользоваться путевым выражением

main_department.managed_by.EMP_NAME

Пример 7. Получить имена всех сотрудников отдела main_department.

Кажется, что запрос очень похож на предыдущий, и что для него годится выражение

main_department.consists_of.EMP_NAME

Но в данном случае имеется связь “один-ко-многим”, и хотя у каждого служащего имеется только одно имя, такие путевые выражения в OQL не допускаются. Правильной формулировкой запроса является следующая:

```
SELECT E.EMP_NAME
        FROM main_department.consists_of AS E
```

Введение в объектную модель ODMG

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 СУБД

GemStone

- ООСУБД GemStone была одной из первых коммерчески доступных ООСУБД.
- Система была разработана компанией Servio-Logic совместно с OGI.
- Разработчики GemStone опирались на язык Smalltalk, потом добавили C и C ++.
- И серверная, и клиентская части системы могут работать под управлением всех основных ветвей ОС UNIX и всех развитых вариантов Windows.
- В настоящее время продукт поддерживается, развивается и распространяется компанией GemStone Systems Inc. (http://www.gemstone.com/).
- 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. Важность этого подхода состоит в том, что, если в приложении используется много данных, то код приложения целесообразно расположить на стороне хранилища данных, а если в приложении производится много изменений над небольшим объемом данных, то имеет смысл разместить код приложения на стороне пользователя. Архитектура позволяет уменьшить объем сетевого трафика без перегрузки сервера, что повышает скорость обработки данных.

Объектно-ориентированные СУБД

GemStone

- Для управления мультидоступом используется механизм транзакций.
- Механизм основан на так называемом *оптимистическом* подходе, при котором каждая сессия работает с собственной локальной копией хранилища объектов, и слияние произведенных в сессиях изменений хранилища происходит при завершении транзакции. Если при завершении транзакции обнаруживается, что произведенные в ней изменения конфликтуют с изменениями других ранее зафиксированных транзакций, то фиксация транзакции не производится, транзакция не завершается, и решение проблемы возлагается на пользователя.
- Автоматическая блокировка объектов не производится, но пользователь может явно запросить блокировку, что повышает шансы на успешную фиксацию транзакции.
- Для обеспечения безопасности данных поддерживается механизм авторизации доступа на уровне владельца объекта и его группы пользователей, т.е. может быть ограничен доступ к некоторым объектам или некоторым методам объектов.
- К каждому объекту приписывается *авторизационный* объект, содержащий данные о том, какие пользователи и в каком режиме (чтения или изменения) имеют доступ к объекту.

Объектно-ориентированные СУБД

GemStone

- Для восстановления базы данных после сбоев аппаратуры используются механизмы репликации, резервного копирования и журнализации.
- Любой авторизованный пользователь может запросить выполнения полного или частичного копирования, а также поддержку реплицирования областей хранилища данных.
- Восстановление базы данных после сбоя системы или дисковых устройств начинается с использования последней по времени резервной копии. После этого при помощи данных, сохраненных в журнальных файлах, хранилище объектов приводится к состоянию, соответствующему последней до момента сбоя зафиксированной транзакции.
- В системе поддерживается целостность по ссылкам между всеми объектами, поскольку все ссылки основываются на использовании объектных идентификаторов, и объекты, для которых существуют ссылки от других объектов, не могут быть удалены. Для повышения скорости доступа к часто используемым коллекциям обеспечивается средство построения индексов.

Объектно-ориентированные СУБД

GemStone

- Объекты делаются стабильными (т.е. сохраняются в базе данных) путем использования своего рода стабильного *корня*, называемого *коннектором*.
- Все объекты, прямо или косвенно достижимые по объектным ссылкам от коннектора, являются стабильными.
- В GemStone для каждого класса, в котором существует хотя бы один стабильный объект, поддерживается эквивалентная серверная версия класса. Другими словами, один вариант класса служит классом в контексте программирования, а другой – в контексте базы данных. Такие пары поддерживаются автоматически: если создается класс в смысле Smalltalk , и некоторый объект этого класса становится стабильным, то автоматически создается серверный класс этого объекта (класс в смысле GemStone).
- Создание коннектора приводит к появлению экземпляра класса GemStone, эквивалентного классу объекта, который должен быть сделан стабильным. Аналогично, любой объект, достижимый от коннектора, автоматически становится стабильным.

Объектно-ориентированные СУБД

GemStone

- В GemStone поддерживается динамическая *сборка мусора (garbage collection)*. Процесс-“мусорщик” автоматически освобождает память, занимаемую объектами, на которые отсутствуют ссылки.
- В среде GemStone можно использовать различные реализации Smaltalk , а также языки C и C ++. Классы и объекты можно создавать с использованием любого из этих языков, и объекты, созданные на одном языке можно использовать в приложениях, написанных на любом другом языке. Подключения к реляционным системам (например, Oracle или IBM DB 2) производятся через шлюзы. Для синхронизации состояния локальной (управляемой GemStone) и внешних копий данных обеспечивается автоматическая модификация данных. В зависимости среды и требований к уровню синхронизованности эти обновления выполняются немедленно или же в пакетном режиме.
- GemStone можно также использовать для управления данными, соответствующими стандартам OLE и CORBA . Для работы с данными в реляционном стиле поддерживаются стандарты SQL и ODBC .

Объектно-ориентированные СУБД

ITASCA

- Распределенная ООСУБД ITASCA основана на результатах проекта Orion , выполнявшегося в MCC. Разработка серии из трех прототипов завершилась выпуском системы, основанной на архитектуре “много клиентов-много серверов”. Поддерживается компанией IBEX Corp. (http://www.ibex.ch).
- В распределенной архитектуре ITASCA частные и совместно используемые базы данных разнесены по узлам локальной UNIX -ориентированной сети.
- Каждое значение данных хранится в одном узле, но централизованное управление отсутствует; все серверы автономны.
- На каждом сервере поддерживаются кэш страниц и кэш объектов, и каждый сервер множество клиентов с обеспечением мультидоступа на основе блокировок. На клиентах поддерживается только кэш объектов.
- Для управления мультидоступом в ITASCA используется двухфазный протокол синхронизационных блокировок с сериализацией транзакций и обнаружением тупиков. Также поддерживаются долгие транзакции на основе перемещения объекта из совместно используемой базы данных в частную базу данных (*check-out*). Для обеспечения совместной работы допускается участие нескольких пользователей в одной долгой транзакции.

Объектно-ориентированные СУБД

ITASCA

- Для всей распределенной базы данных поддерживается единая схема с использование подсхем для частных фрагментов базы данных. Модель данных включает следующие аспекты:
  - ✔ множественное наследование;
  - ✔ представление классов в виде объектов;
  - ✔ наличие свойств и операций классов;
  - ✔ наличие свойств и операций классов;
  - ✔ поддержка ограничений целостности;
  - ✔ возможность перегрузки операций.
- В любое время могут добавляться новые данные, классы, свойства и операции.
- Для обеспечения контроля над распространением таких операций как удаление объекта имеется возможность определения составных объектов.
- Для поддержки мультимедийных приложений имеется возможность использования линейных массивных объектов, которые предназначены прежде всего для хранения последовательных данных, таких как текст или аудиоданные.

Объектно-ориентированные СУБД

ITASCA

- В пространственных массивных объектах имеются два измерения, и они подходят, например, для хранения изображений.
- Восстановление базы данных после сбоев производится на основе журнала, предназначенного для аннулирования результата выполненных операций (*undo log*). Это позволяет в процессе восстановления устранить эффект всех транзакций, не завершившихся к моменту сбоя.
- Фиксация транзакции заключается в том, что на сервере все объекты, измененные транзакцией, перемещаются из буфера объектов в буфер страниц.
- Имеется возможность создания классов, поддерживающих оповещение. Имеются две формы оповещения – пассивная и активная. Пассивное оповещение состоит в том, что сохраняется информация о модификации или удалении экземпляров класса. Приложение может обратиться классу с запросом данных о таких событиях. Активное оповещение приводит к вызову некоторой операции при выполнении операций модификации, удаления, создания версии, перемещения объекта из общей базы данных в частную базу данных (*check-out*) или наоборот (*check-in*).

Объектно-ориентированные СУБД

ITASCA

- Поддерживается механизм индексирования, основанный на использовании техники B+-деревьев. Можно создавать индексы для одного класса и одного свойства или для нескольких свойств нескольких классов.
- Допускается создание временной, рабочей или “выпускной” версии объекта.
- Для динамического или статического связывания разных версий поддерживается иерархия происхождения версий. При использовании динамического связывания версий иерархия автоматически модифицируется при создании новых версий.
- Безопасность данных обеспечивается на основе механизма авторизации доступа, в котором конкретная привилегия (доступ по чтению, доступ по записи или создание) предоставляется роли, за которой может стоять один пользователь или группа пользователей.
- Привилегии могут быть подсоединены к базам данных, классам, экстентам, объектам, операциям и свойствам.
- Имеется авторизация по умолчанию, которая подразумевается для любой роли и может быть дополнена явной авторизацией, положительной (с добавлением привилегий) или отрицательной (с изъятием привилегии).

Объектно-ориентированные СУБД

ITASCA

- При использовании C ++ стабильность достигается путем доступа к библиотеке классов, поддерживающих стабильность. В CLOS (Common Lisp Object System) обеспечивается метакласс стабильности. Стабильные объекты должны быть экземплярами классов, являющихся экземплярами этого метакласса. Кроме того, можно указать, что некоторые свойства стабильного класса являются недолговечными.
- В ITASCA поддерживаются C , C ++, Smalltalk , CLOS. Акцент делается на возможности динамического изменения схемы без остановки действия системы и без потребности в массовой повторной компиляции и редактирования связей.
- Доступ к программам на каждом из языков производится через функциональный API. В случае использования C++ автоматически создается файл заголовков, который сливается с исходными файлами программного кода при генерации приложения.
- Собственный механизм запросов ITASCA позволяет запрашивать данные в частной базе данных, общей базе данных или сразу в обеих базах данных. Для повышения производительности применяются оптимизация запросов и методы распараллеливания.