УДК 681.518.5

В.В. Борисов, В.П. Зинченко, И.П. Муха

СИСТЕМА АВТОМАТИЗИРОВАННОГО УПРАВЛЕНИЯ ПРОЕКТНЫМИ ДАННЫМИ

Аннотация: Описаны концепция и основные принципы функционирования объектной системы управления данными "SPACE", которая была разработана для решения задачи синтеза конечно-элементных моделей планера самолета.

Ключевые слова: конечно-элементные модели, прочность конструкций, напряженно-деформированное состояние, СУБД.

Введение

В [1] показано, что одной из главных проблем, связанных с принципом декомпозиции, является проблема синтеза проектных моделей, которая решена только для узкого круга задач, например, синтеза 3D-моделей сборок. Не решена проблема синтеза структур и свойств конечно-элементных моделей (КЭМ), что не позволяет оптимизировать с помощью метода декомпозиции алгоритмы решения задач, связанных с анализом прочности конструкций сложных технических объектов (СТО). В результате, недопустимо увеличивается время создания КЭМ и анализа напряженно-деформированного состояния (НДС) конструкции СТО. В [2] изложены основные проблемы декомпозиции и синтеза КЭМ СТО. На основании анализа проблем показано, что задача синтеза КЭМ СТО может быть решена только путем использования произвольного множества специализированных алгоритмов, каждый из которых должен разрабатываться с учетом особенностей структур объединяемых КЭМ и путей передачи нагрузок. При этом перечень используемых алгоритмов определяется непосредственно в процессе проектирования, что возможно только с помощью информационных технологий (ИТ), обеспечивающих выбор алгоритмов вне процесса программирования. Показано, что обмен данными, необходимый для решения задачи синтеза КЭМ, может быть реализован только при использовании активных КЭМ, которые могут обмениваться данными непосредственно, без участия пользователей. В [3] исследованы современные системы управления базами данных (СУБД), в том числе системы управления проектными данными (PDM-системы). Показано, что ни одна из существующих PDM-систем не обеспечивает синтез КЭМ, поскольку формирование активных КЭМ возможно только при использовании объектно-ориентированных ИТ, которые реализуются с помощью объектных СУБД (ОСУБД). Анализ концептуальных моделей различных ОСУБД показал, что ни одна из них не обеспечивает формирование активных КЭМ, поскольку формируемые с их помощью объекты активизируются только внутри прикладных программ. Это исключает возможность выбора алгоритмов вне процесса прогаммирования, что не позволяет решить задачу синтеза структур и свойств КЭМ СТО.

Таким образом, задача повышения эффективности методов и средств автоматизированного управления проектными данными является безусловно актуальной.

В статье описываются концепция и основные принципы функционирования ОСУБД “SPACE”, которая была разработана для решения задачи синтеза КЭМ планера самолета. Концептуальная модель “SPACE” оптимизирована для работы с большим количеством алгоритмов. Поскольку в объектной базе данных (ОБД) количество экземпляров алгоритмов всегда соответствует количеству экземпляров структур данных, при разработке “SPACE” был использован принцип непосредственной адресации, в соответствии с которым вычисление исполнительных адресов производится один раз, при компиляции исходных текстов классов объектов. Это позволило отказаться от многократного вычисления испольнительных адресов в процессе анализа и обработки данных, которое традиционно используется в реляционных СУБД и большинстве ОСУБД, в результате чего существенно уменьшилось время поиска данных для задач, использующих большое количество алгортмов. Например, использование “SPACE” позволило уменьшить время формирования КЭМ кессона крыла с 2-х лет до 3-х месяцев.

Постановка задачи

Предложить средства и методы, повышающие эффективность управления данными при решении задач формирования КЭМ СТО, в которых используется большое количество специализированных алгоримов.

Оптимизация управления данными

В объектной модели данных количество экземпляров алгоритмов, используемых для анализа и обработки данных, всегда соответствует количеству экземпляров структур данных, что, в принципе, позволяет решать задачи, использующие большое количество алгоритмов.

Анализ концептуальных моделей собвременных ОБД показал, что все они формируются в виде хранилищ объектов, в которых функции ОСУБД ограничиваются обеспечением целостности данных, а также поиском объектов в ОБД [4]. Активизация методов объектов и изменение их свойств производятся только в процессе фукционирования программ, внутри которых создаются объекты (экземпляры классов). Такая концепция управления данными не позволяет выбирать алгоритмы вне процесса программирования, так как объекты являются частью прикладного программного обеспечения (ППО). Соответственно, исполнительные адреса данных (свойств) различных объектов зависят от структуры конкретного ППО и могут вычисляться только в процессе трансляции. Поэтому исполнительный адрес объекта в ОБД может быть вычислен только в процессе поиска, на основе данных, записанных в индексных таблицах (рис. 1).

Рассматривая объекты как части ППО, исследованные ОСУБД не обеспечивают автоматического обмена данными между моделями, храня-

 ISSN 1560-8956

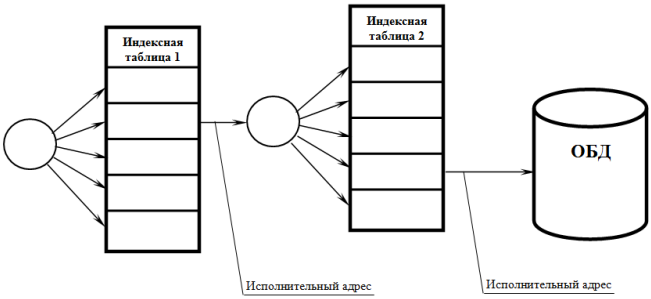

Рис. 1 – Поиск данных по сцепленному ключу

щимися в объектах, так как не возможен прямой обмен данными между объектами, находящимися в разных модулях ППО. Кроме этого структура ОБД не содержит связей между объектами. Связи между объектами описываются в исходных текстах ППО, только в процессе программирования, что не позволяет решать задачу синтеза КЭМ СТО.

Для решения этой проблемы предложено рассматривать объекты как активные структурные элементы, функционирующие вне ППО. При активизации объект загружается в оперативную память компьютера самостоятельно. Это позволяет компилировать класс объекта как отдельный программный код и вычислять исполнительные адреса структур данных на этапе компиляции, что дает возможность разработать унифицированную процедуру формирования объектов непосредственно в ОБД, без использования ППО. Все данные, необходимые для формирования объектов, определяются в процессе компиляции классов и хранятся в специальной базе данных (БД) классов, откуда считываются при формировании объектов. Предложенная концепция формирования ОБД реализована в ОСУБД “SPACE”, специально разработанной для решения задачи синтеза КЭМ. Унифицированная процедура формирования объектов с помощью ОСУБД, используемая в “SPACE”, позволяет формировать структуру ОБД вне процесса программирования. Максимальное количество ОБД (Space-ов), на которые может ссылаться одна ОСУБД “SPACE”, равно 65000. При формировании каждому объекту присваивается индивидуальный числовой код, который не меняется в течение всего времени существования объекта. Значение кода объекта соответствует номеру дескриптора объекта в глобальной таблице дескрипторов (GODT), в которой автоматически регистрируется каждый объект. Максимальное количество объектов в одном Space равно 4 000 000 000.

Для ускорения загрузки объектов в оперативную память (ОП) компьютера в ОБД “SPACE” используется система адресации, основанная на использовании внутриобъектных испольнительных адресов: $Addr = Seg : Offset$. Структура объекта соответствует структуре испольнительного адреса и включает в себя сегмент программных кодов “SPACE” ($Seg0$),

а так один и более сегментов данных ($Seg1, Seg2, \ldots, SegN$). Физически все сегменты хранятся в отдельных файлах. Для оптимизации процедуры определения испольнительных адресов объектов ОСУБД “SPACE” использует расширенный формат исполнительного адреса, который, кроме номера сегмента и смещения, содержит номер Space и индивидуальный код объекта в Space: $Addr = Space : Obj : Seg : Offset$.

В отличие от исследованных СУБД, формирование объектов в Space (за исключением первого или головного объекта) возможно, только с помощью уже существующих объектов. Для этого в состав процедур-членов объектов должна включаться функция `int $_cre_object ()`, входящая в состав ядра “SPACE”. При использовании указанной функции, каждый формируемый объект, кроме GODT, дополнительно регистрируется в локальной таблице дескрипоторов объектов (LODT) родительского объекта. То есть каждый объект в Space, за исключением головного, является дочерним объектом (субобъектом) по отношению к к-либо другому ранее созданному объекту. LODT служит для объединения элементов объекта и, кроме дескрипоторов субобъектов, содержит ссылку на локальную таблицу дескрипторов сегментов (LSDT) своего объекта. Таким образом структура объекта “SPACE” содержит две локальные таблицы дескрипторов – LODT и LSDT, один сегмент программных кодов и до 65500 сегментов данных.

Наличие в структуре каждого Space дополнительных локальных таблиц дескрипторов позволяет оптимизировать процедуру определения исполнительных адресов объектов в процессе их активизации. Все объекты в Space, за исключением головного, активизируются с помощью других объектов. Предусмотрено два основных метода активизации объекта.

В соответствии с первым методом исполнительный адрес объекта в Space определяется по его номеру в LODT родительского объекта (локальному номеру). В этом случае испольнительный адрес активизируемого объекта определяется по следующей схеме (рис. 2): на основании локального номера из LODT родительского объекта считывается ссылка на LODT активизируемого объекта, из которой считывается ссылка на соответствующую LSDT, из которой считываются ссылки на сегменты активизируемого объекта.

Для активизации субобъекта в состав процедур-членов родительского объекта должна включаться функция `int $_call_sub_object ()`.

В соответствии со вторым методом исполнительный адрес объекта в Space определяется по его индивидуальному коду. В этом случае исполнительный адрес LODT активизируемого объекта считывается из GODT. Для активизации объекта по индивидуальному коду в состав процедур-членов вызывающего объекта должна включаться функция `int $_call_object_by_number ()`.

В обоих методах не используется длительная процедура поиска по ключу, что актуально для БД проектов СТО, содержащих большое коли-

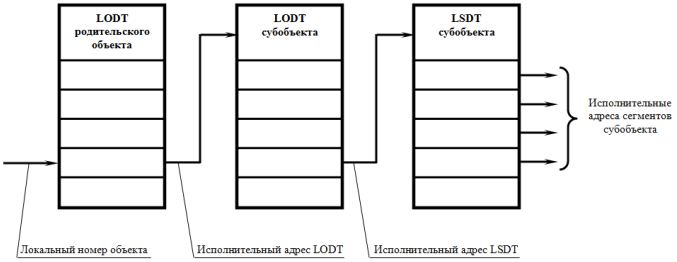

Рис. 2 – Определение исполнительного адреса субобъекта

чество объектов. При этом обеспечивается основная часть потребностей во взаимодействии моделей в проекте.

Кроме указанных методов “SPACE” дополнительно использует метод вычисления исполнительного адреса объекта на основе его наименования, которое записывается в GODT в процессе формирования объекта. В этом случае GODT используется в качестве индексной таблицы, а наименование активизируемого объекта – в качестве одиночного ключа. В процессе поиска дескриптора объекта производится последовательное считывание из GODT дескрипторов объектов и сравнение их с ключем. Данный метод является вспомогательным и используется для активизации сравнительно небольшого количества объектов общего пользования, к которым неудобно обращаться по локальному номеру или индивидуальному коду. Обычно такие объекты формируются в начале проектирования, что существенно сокращает время их поиска. При этом следует обеспечить уникальность наименования вызываемых объектов, поскольку “SPACE” допускает регистрацию в GODT двух и более объектов с одинаковыми наименованиями.

Оптимизация взаимодействия объектов

Для решения задачи синтеза КЭМ СТО должны быть решены две проблемы: автоматический обмен данными между моделями и синхронизация состояний различных КЭМ. Для решения этих проблем предложен метод пересечения объектов, в соответствии с которым два и более объектов могут ссылаться на одну и ту же область памяти ОБД (рис. 3).

Для реализации этого метода в “SPACE” предусмотрены особые, “абстрактные”, типы данных. В отличие от обычных типов данных, абстрактным типам данных выделяются индивидуальные дескрипторы в LSDT, но не выделяется память ОБД. При написании программных текстов абстрактные типы используются также, как обычные. Однако перед их первым использованием такие данные должны быть инициализированы. Процесс инициализации включает в себя две операции: запись в дескриптор абстрактного типа ссылки на реальную область памяти в ОБД и загрузку в оперативную память данных, на которые ссылается дескриптор. В отличие от обычных указателей, абстрактные

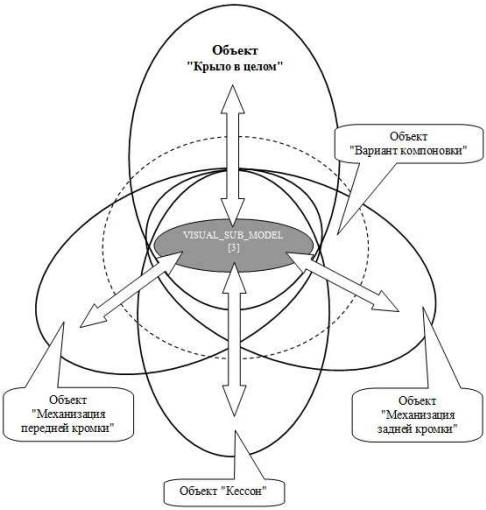

Рис. 3 – Принципиальная схема пересечения объектов “SPACE”

типы данных могут ссылаться на данные в любом объекте и в любом Space. Поэтому при их инициализации используется расширенный формат исполнительного адреса. Процесс инициализации осуществляется по следующей типовой схеме (рис.4). Объект, содержащий данные обычного типа, к которым должен быть обеспечен доступ со стороны другого объекта, (объект-донор) записывает адрес разделяемой области памяти ОБД в специальную область ОП (буфер), для чего используется функция `void $_get_addr ()`, входящая в состав ядра “SPACE”. Далее объект-донор активизирует объект, для которого обеспечивается доступ к разделяемой области памяти (объект-реципиент). Объект-реципиент считывает из буфера ссылку на разделяемую область памяти и записывает ее в дескриптор данных абстрактного типа, для чего используется функция `void $_set_index()`, также входящая в состав ядра “SPACE”. После записи ссылки в дескриптор данные автоматически загружаются в ОП. Для описания абстрактных типов данных используется ключевое слово “virtual”.

Абстрактный тип данных, чей формат совпадает с форматом данных, на который он ссылается, в сочетании с функциями активизации объектов, обеспечивает автоматический обмен данными между объектами.

Так как ссылки в дескрипторах абстрактных типов после инициализации сохраняются в ОБД постоянно, данные, на которые они ссылаются, загружаются в ОП при каждой активизации объекта-реципиента наравне с его собственными данными, что дает возможность синхронизировать состояния объекта-донора и объекта-реципиента.

 ISSN 1560-8956

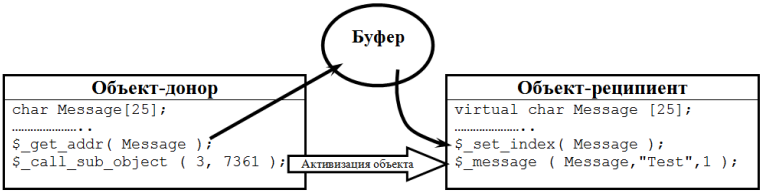

Рис. 4 – Схема инициализации абстрактных типов данных “SPACE”

Средство исполнения процедур

Команды, содержащиеся в сегменте программных кодов активизированного объекта, считываются и обрабатываются с помощью инерпретатора, входящего в состав ядра “SPACE”. Принципиальная схема интепретатора, спроектированного по типу MISC-процессора, приведена на рис.5.

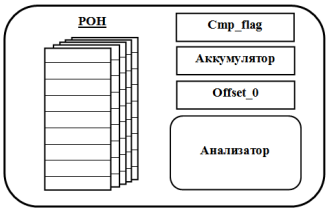

Рис. 5 – Принципиальная схема интерпретатора “SPACE”

С целью ускорения процесса интерпретации внутриобъектных процедур и исключения событий, связанных с синтаксическими ошибками, для интерпретатора “SPACE” была разработана система кодировки команд, подобная той, которая используется MISC-процессорами семейства i86 (рис. 6).

Длина команды интерпретатора составляет от 1 до 11 байтов, в зависимости от количества и размеров операндов. Структура команды включает код группы команд (0-й байт), код команды в группе (1-й байт) и операнды (2–10 байты). Для облегчения кодирования операций над различными типами операндов в состав регистров общего назначения (РОН) процессора входят регистры, соответствующие всем стандартным типам данных (таблица 1):

Всего предуспотрено по 100 регистров каждого типа. При вызове подпрограммы или другого объекта содержимое массива регистров не записывается в стек. В этом случае система создает новый экземпляр массива регистров, который уничтожается после окончания работы подпрограммы (объекта). Не предусмотрены сегментные регистры. Номер сегмента содержится в первых двух байтах адресного регистра $addr.

Рис. 6 – Типы команд интерпретатора

Таблица 1. Регистры интерпретатора “Space”.

Тип данных	**Имя регистра**
Однобайтное целое со знаком	$char
Однобайтное целое без знака	$uchar
Двухбайтное целое со знаком	$short
Двухбайтное целое без знака	$ushort
Четырехбайтное целое со знаком	$int
Четырехбайтное целое без знака	$uint
Четырехбайтное с плавающей точкой	$float
Восьмибайтное с плавающей точкой	$double
Адрес (6 байтов)	$addr

Аккумулятор также представляет собой массив регистров всех указанных типов и, в основном, используется для возврата данных из процедур. Значение регистра Offset_0 учитывается при вычислении физического адреса данных в ОП и служит корректировки исполнительных адресов, закодированных в командах обмена данными между РОН и ОП.

Средство описания и компиляции классов

БД классов имеет древовидную иерархическую структуру, состоящую из классов объектов ОСУБД “SPACE”, объединенных в группы как по функциональному признаку, так и по принципу наследования свойств. Для структурирования БД по функциональному приципу используются списки, которые представляют собой табличные структуры данных, содержащие ссылки на классы объектов и другие списки. Структура БД

 ISSN 1560-8956

классов “SPACE” формируется и поддерживается специальной программой Space Class Explorer (рис. 7), не входящей в состав ядра “SPACE”.

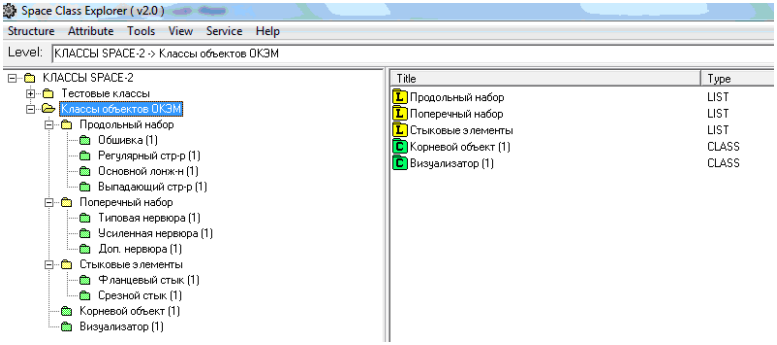

Рис. 7 – Общий вид Space Class Explorer

Классы, содержащиеся в БД классов, представляют собой сложные структуры данных и также могут использоваться как списки. Такое размещение классов используется для определения наследования классов. Класс, который регистрируется в другом классе, наследует его свойства и методы. Каждому классу присваивается индивидуальный код, который может быть определен с помощью диалога, активизируемого функцией `char $_select_class_dialog ()`, входящей в состав ядра “SPACE”.

Первичное кодирование, а также компиляция и отладка классов производятся с помощью программного модуля Space Class Builder (рис. 8), активизируемого из Space Class Explorer.

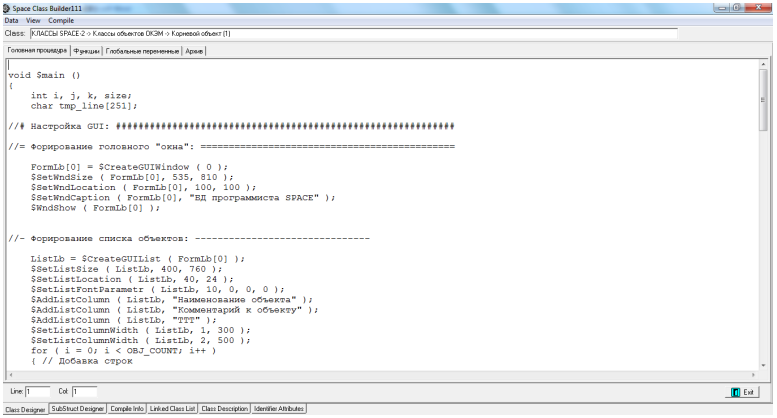

Рис. 8 – Общий вид Space Class Builder

В качестве языка первичного кодирования классов используется язык “SpaceC”, синтез которого, в основном, соответствует синтаксису языка

“Си”. Язык “Space C”, хотя и служит для описания классов объектов, не содержит инструкции “class”. Это связано с тем, что в “SPACE” отсутствует понятие “программа”, как таковая. Соответственно, нет необходимости описывать программные коды, не входящие в состав классов. Первичный код класса должен обязательно содержать описание функции `$main ()`, которая является аналогом функции main () языка “Си”.

Транслятор Space Class Builder является 3-хпроходным, поэтому один раз описанная переменная или функция, может быть использована на любом месте исходного текста. Это дает возможность отказаться от ключевых слов “#include” и “extern”, а также от прототипов процедур.

При формировании объекта ядро “SPACE” считывает данные из класса, чей индивидуальный код указан в числе аргументов при обращении к функции `$_cre_object ()`.

Средства поддержки графического пользовательского интерфейса

Графический пользовательский интерфейс “SPACE” (SPACE-GUI) поддерживается с помощью встроенных функций ядра ОСУБД. В отличие от обычных функций, описываемых разработчиками классов, встроенные функции не требуют к-либо описания.

Для настройки SPACE-GUI удобно использовать процедуру $main(), которая всегда выполняется первой:

```
void $main ()
{
  FormLb = $CreateGUIWindow ( 0 );
  $SetWndSize ( FormLb, 148, 432 );
  $SetWndLocation ( FormLb, 272, 321 );
  $SetWndCaption ( FormLb, ``Окно объекта SPACE'' );
  $WndShow ( FormLb );
  ButtonLb[0] = $CreateGUIButton ( FormLb );
  $SetButtonSize ( ButtonLb, 25, 185 );
  $SetButtonLocation ( ButtonLb, 72, 32 );
  $SetButtonCaption ( ButtonLb, ``Exit'' );
  $SetButtonClickEvent ( ButtonLb, ExitFunc );
  $ButtonVisible ( ButtonLb, 1 );
}
```

В результате выполнения приведенного кода формируется “окно” с кнопкой.

Если при активизации объекта было сформировано хотя бы одно “окно”, то окончание функции $main() не приводит к деактивации объекта. В этом случае деактивация текущего объекта и возврат управления в вызывающий объект происходит только либо при закрытии головного “окна”, либо при выполнении функции ядра “$exit ()”, которая закрывает все “окна” активного объекта. Головным “окном” считается “окно”, которое было сформировано первым. Если объект активизируется из другого

 ISSN 1560-8956

объекта, все “окна” активизирующего объекта автоматически блокируются. После окончания работы активизированного объекта, производится автоматическая разблокировка “окон” активизирующего объекта.

Выводы

Проведенный анализ методов и средств управления ОБД, реализованных в системе “SPACE”, показал, что ОСУБД “SPACE” является наиболее оптимальным средством решения задачи синтеза КЭМ СТО. В ней впервые предложена и реализована ИТ, рассматривающая объект как активный элемент структуры ОБД, что позволило решить проблему автоматизированного обмена данными между различными КЭМ.

Впервые предложен и реализован метод адресации в ОБД, использующий сегментную модель, что позволило отказаться от процедур вычисления исполнительных адресов по индексным таблицам, которые в проектных БД имеют недопустимую длительность. Включение в структуру ОБД таблиц дескрипторов объектов и сегментов позволил решить проблему ускоренного поиска объектов, за счет минимизации объема операций с индексными таблицами.

Впервые предложен и реализован принцип пересечения объектов, в соответствии с которым два и более объектов могут оперировать общими структурами данных, что позволило упростить обмен данными между КЭМ и обеспечить синхронизацию их состояния.

Литература

1. Зинченко В.П., Борисов В.В. Методы и алгоритмы автоматизированного проектирования сложных технических объектов // УСиМ, 2011. – Вып. 1. –С. 46–56.
2. Борисов В.В., Зинченко В.П. Анализ актуальных проблем информационной технологии декомпозиции и синтеза конечно–элементных моделей // Открытые информационные и компьютерные интегрированные технологии. – Харьков: Гос. Аэроком. Ун-т “ХАИ”, 2009. – Вып. 44. – С. 79 – 91.
3. Зинченко В.П., Борисов В.В., Конотоп Д.И.. Анализ средств и методов информационных технологий синтеза структур конечно-элементных моделей // Зб-к “Інформаційні системи, механіка та керування” ФАКС НТУУ “КПІ”, 2009. – Вып. 3. –С. 112 – 121.
4. http://www.cyberguru.ru/database/database-theory/object-oriented-db-overview.html. Объектно-ориентированные базы данных - основные концепции, организация и управление.

Отримано 04.12.2011 р.