###### Содержание

[Введение. 3](#_Toc534433390)

[Oracle Power Objects. 5](#_Toc534433391)

[Аспекты и компоненты приложения. 5](#_Toc534433392)

[Объектно-ориентированная разработка. 5](#_Toc534433393)

[Методы 6](#_Toc534433394)

[Внешний интерфейс: Наборы записей и связанные контейнеры 7](#_Toc534433395)

[Разработка в объектно-ориентированной среде. 8](#_Toc534433396)

[Прочие аспекты. 9](#_Toc534433397)

[Сервер: Сеансы и базы данных. 10](#_Toc534433398)

[Базы данных 10](#_Toc534433399)

[Базы данных Blaze 12](#_Toc534433400)

[ORACLE7 Server 13](#_Toc534433401)

[Базы данных SQL Server 14](#_Toc534433402)

[Объекты базы данных 15](#_Toc534433403)

[Среда разработки 17](#_Toc534433404)

[Структура приложения 19](#_Toc534433405)

[Обработка транзакций 19](#_Toc534433406)

[Подход к разработке, реализуемый в Oracle Power Objects 20](#_Toc534433407)

[Если начать с клиента 21](#_Toc534433408)

[Если вначале разрабатывается внешний интерфейс, следует ответить на следующие вопросы: 21](#_Toc534433409)

[Если начинать с сервера базы данных 22](#_Toc534433410)

[Начиная проектирование с сервера базы данных, необходимо ответить на следующие вопросы: 23](#_Toc534433411)

[Как проектировать пользовательские классы и библиотеки 24](#_Toc534433412)

[Заключение 26](#_Toc534433413)

[Список литературы 27](#_Toc534433414)

# Введение.

Мощные современные системы управления базами данных, такие как ORACLE 7, SQL Server и другие, применяются, как правило, для поддержания и обработки больших и очень больших баз данных, к информации которых одновременно обращается множество пользователей. В этих условиях, обеспечение каждого из этих пользователей или группы пользователей неким средством (приложением) для доступа к данным локальной или удаленной базы данных становится весьма сложной задачей, требующей существенных трудозатрат. Решить эту задачу можно либо путем привлечения значительных сил программистов и разработчиков, либо применением новых, высокопроизводительных технологий разработки. Объектно-ориентированное, визуальное проектирование – пример успешной реализации второго, интенсивного подхода.

Oracle Power Objects – высококачественный (что характерно для всех продуктов Oracle Corporation), оригинальный и надежный инструмент для разработки программного обеспечения доступа к базам данных. Он объединяет возможности приложений клиент/сервер с преимуществами объектно-ориентированной модели разработки.

Для построения приложения разработчик должен иметь в своем распоряжении некоторые конструктивные элементы – объекты. В Oracle Power Objects каждый компонент приложения, от непосредственно базы данных (или серверной части системы) до интерфейса клиента (или внешнего интерфейса) – это объект, который можно идентифицировать и управлять им, используя унифицированные технологии. Всем объектам, с которыми работает проектировщик, независимо от типа объекта, присущи стандартные свойства (характеристики, управляющие поведением объекта) и, в большинстве случаев, методы (программное описание действий объекта) – стандартные или пользовательские.

Такой объектно-ориентированный подход к разработке упрощает задачу создания приложения и более близко и точно, чем другие модели разработки, отражает представление проектировщика относительно компонентов приложения и их взаимосвязи. Обычно, при работе с определением объекта (формы, отчета, таблицы), разработчик задумывается о том, как управлять аспектами этого объекта. При этом все существенные характеристики объекта определяются значениями его свойств.

Объекты приложения Power Objects могут быть вполне «осязаемыми» для пользователя, как например, окна диалога, командные кнопки, текстовые поля и т.д. Объекты могут также представлять собой некоторые категории, которые не являются частью пользовательского интерфейса. Это, например, связи базы данных и сеансы пользователя. При построении приложения разработчик программирует действия объектов или задает реакцию этих объектов на некоторые события или состояния среды или приложения.

При построении приложения базы данных очень важно иметь возможность преодолеть ограничения, накладываемые на использование базы данных конкретной платформой. Перед передачей приложения к клиентам производственную среду полезно первоначально отладить и протестировать прототип приложения на локальной базе данных. Закончив отладку приложения, можно затем экспортировать определения всех таблиц, представлений, последовательностей и индексов на удаленный сервер, минимизируя тем самым время его простоя.

Oracle Power Objects позволяет работать с распространенными сетевыми базами данных, такими как SQL Server и ORACLE7. Кроме того, на уровне клиента Oracle Power Objects обеспечивает связь с Personal Oracle7, а также Blaze – персональной базой данных, поставляемым с Oracle Power Objects. Blaze полностью совместима по структуре и данным с ORACLE7.

В дополнение к этой гибкости при выборе платформы базы данных, Oracle Power Objects ещё более упрощает разработку приложений, автоматизируя многие аспекты доступа к базе данных посредством чрезвычайно развитых функций drag-and-drop. Например, положив мышью описание таблицы на форму, можно автоматически ассоциировать форму с этой таблицей, даже если это объект удаленной базы данных. Эта возможность позволяет разработчику, вместо написания сложного программного кода соединения внешнего интерфейса с сервером базы данных, сосредотачиваться непосредственно на проектировании приложения.

Для пользовательского программирования в Oracle Power Objects используется Oracle Basic, процедурный язык с поддержкой SQL, совместимый с Visual Basic for Application. Если разработчик ранее программировал приложения в Microsoft Visual Basic или Microsoft Access – он может быстро и легко перейти к написанию кода на Oracle Basic в Power Objects.

# Oracle Power Objects.

## Аспекты и компоненты приложения.

Процедуры разработки приложений разных типов ставят разработчика перед рядом сложных проблем и выборов. Принимаемые при этом решения во многом определяются тем, как приложение должно обращаться к данным – локально, к базе данных на том же РС, где расположен исполняемый файл приложения, или дистанционно, в системе сетевой базы данных, подобной ORACLE7 или SQL Server. Располагая инструментарием Oracle Power Objects помогает упростить и облегчить разработку приложения для среды клиент/сервер за счет применения следующих технологий:

* Объектно-ориентированная модель разработки позволяет использовать стандартную методологию при работе со всеми объектами – от таблиц и представлений на сервере базы данных до форм, текстовых полей и растровых объектов внешнего интерфейса.
* Развитые механизмы drag-and-drop помогают быстро связать формы и отчеты (объекты внешнего интерфейса) с таблицами и представлениями (объектами данных).
* Автоматизированная обработка транзакций устраняет необходимость написания для управления транзакциями больших объемов программного кода, а в ряде случаев позволяет вообще обойтись без программирования.

## Объектно-ориентированная разработка.

В общей объектно-ориентированной модели каждый компонент приложения предоставляется разработчику как объект. Объект имеет два различных набора характеристик:

* Свойства определяют представление, поведение и другие черты объекта. Цвет фона формы, строка соединения, используемая для открытия сеанса базы данных, и исходная таблица базы данных для отчета – все это свойства тех или иных объектов.

В период выполнения приложение может оценивать значение, назначенное свойству, и, когда требуется, назначать ему новое значение. Например, при нажатии командной кнопки в окне диалога, приложение проверяет, установлено ли свойство Enabled этой кнопки к значению True.Если это так, нажатие этой кнопки обусловит некоторый результат, иначе, нажать эту кнопку не удастся. В период выполнения, приложение может деактивировать командную кнопку, назначая свойству Enabled командной кнопки значение False.

* Методы – процедуры, которые выполняют некоторую обработку, связанную с объектом. Например, если нажатием командной кнопки требуется открыть форму, необходимо к телу метода Click() командной кнопки добавить соответствующий программный код. Код метода записывается на языке программирования Oracle Basic и добавляется к методу через окно редактирования в листе свойств.

Большинство стандартный методов (то есть, методов, которые являются частью умолчательного определения объекта) имеют связанную с ними умолчательную обработку. Например, метод формы OpenWindow() загружает форму в память и выводит ее на экран. Если стандартный метод имеет умолчательную обработку, к нему можно добавить некоторый код, который или заменяет умолчательную обработку, или расширяет ее. Как часть умолчательной обработки, один метод может вызывать другой. Это означает, что, если умолчательная обработка для одного метода прерывается, следующий метод в цепочке не будет вызван.

Разработчик может также создавать свои собственные пользовательские методы, которые затем добавляются объекту.

### Методы

Метод определяет поведение объекта при обращении к нему или при наступлении некоторого события. Например, при вызове формы ее метод Open Window() загружает форму в память и выводит ее на экран в интерфейсе приложения. В методах размещается программный код Oracle Basic, посредством которого настраивается работа приложения. Фактически, весь код Oracle Basic инкапсулируется в методах. В этой главе описывается, как вызывать объектные методы и как добавлять к ним код Oracle Dasic.

Методы – это программные процедуры – функции или подпрограммы. Они могут (но не обязательно должны) принимать один или больше параметров.

Oracle Power Objects включает набор предопределенных стандартных методов, большинство из которых включают умолчательную обработку, выполняемую при вызове метода. К объектам можно добавлять также пользовательские методы; пользовательские методы не имеют умолчательной обработки.

Стандартные и пользовательские методы можно настраивать, добавляя код Oracle Basic. Для стандартных методов можно или полностью переопределять умолчательную обработку, или вставлять пользовательский код до или после вызова умолчательной обработки.

Метод может быть инициирован одним из двух способов:

1. Через событие. Событие – действие, которое имеет место в интерфейсе приложени. События происходят, когда пользователь выполняет в приложении некоторые манипуляции (например, щелчок на элементе управления или закрытие приложения) или в ответ на системное функционирование.
2. Вызовом метода. Метод можно вызвать явно через код Oracle Basic.

Если метод – функция, он может возвращать значение, которое можно использовать где-либо в выражении или в операторе присваивания. Метод-функция используется, подобно встроенной функции Oracle Basic.

Например, метод GetRecordset() – функция. GetRecordset() возвращает значение типа данного Object, которое можно присваивать переменной. Переменную можно затем использовать везде, где требуется обращение к объекту набора записей.

Если метод – подпрограмма, его имя используется не подобно функции, в операторе присваивания или выражении, а в виде команды, то есть, единственный вызов в строке. Подпрограммы не возвращают значения; однако, подпрограмма может изменять переданный ей параметр.

В ходе выполнения, подпрограмме или функции передаются параметры. Эти параметры специфицируются в круглых скобках рядом с именем метода. Параметры функции могут быть переданы по ссылке (умолчание) или по значению.

При передаче параметра по ссылке, фактически, передается указатель на область памяти, занятой переменной, которая используется для специфицирования параметра, По ссылке могут быть переданы только переменные; другие типы выражений автоматически передаются по значению. Любые изменения, проведенные в параметре при выполнении функции, воздействуют на переменную в вызывающем методе.

С другой стороны, при передаче параметра по значению, передается только «абстрактное» текущее значение, присвоенное параметру, без ссылки непосредственно на сам параметр. В этом случае изменения, проведенные в параметре, не имеют никакого эффекта вне подпрограммы или функции. Все выражения, не содержащие переменных, автоматически передаются по значению. Переменные также могут быть переданы явно по значению, если в объявлении функции используется ключевое слово BYVL.

Чтобы создать пользовательскую функцию или подпрограмму, необходимо объявить ее и ассоциировать ее с объектом в приложении .Чаще всего функцию или подпрограмму добавляют к форме, классу или отчету, откуда она вызывается, но фактически, метод можно добавлять к любому объекту приложения.

Чтобы настроить поведение метода, к нему добавляется программный код. Код метода может быть добавлен к любому стандартному или пользовательскому методу Oracle Power Objects.

В любом методе Oracle Power Objects неявно объявляет набор переменных. Переменная объявляется для каждого параметра в объявлении метода и имеет тип данного, обозначенный в объявлении. Эти переменные можно использовать в коде, подобно любым другим значениям. Необходимо отметить, что присваивание значений этим переменным имеет смысл, только если параметр был передан по ссылке; параметр, переданный по значению, также может быть установлен, но по выполнении метода значение параметра исчезает.

Для методов-функций возвращаемое значение функции назначают, присвоив значение имени функции. Это имя, фактически, не является переменной; Это – просто соглашение для представления возвращаемого значения.

При вводе в метод программного кода Oracle Basic автоматически переопределяется любая умолчательная обработка, обычно выполняемая этим методом.

Эту установленную по умолчанию обработку можно, однако, выполнять, добавляя к коду метода синтаксис Inheried.*имя\_метода.*

Умолчательная обработка выполняется в точке кода метода, где она вызвана. Следовательно, ей может предшествовать или выполняться после ее пользовательский код Oracle Basic. Если вызываемый стандартный метод требует параметров, то их необходимо передать в вызове процедуры Inherited. Имя\_метода.

Умолчательная обработка для метода часто включает вызов в предопределенном порядке ряда других методов, В таких случаях, прежде, чем поток управления возвращается в вызывающую процедуру, сразу после оператора Inherited. Имя\_метода выполняются эти методы умолчательной обработки.

Любой код, который добавляется к методу, может нуждаться в корректировке. Процедуры не всегда сразу реализуют идею разработчика так, как он это предполагает при написании программного кода.

Отладка – процесс анализа, корректировки и тестирования кода, выполняемый с целью добиться требуемого порядка выполнения приложения и получения требуемых результатов. Когда выполнение сталкивается с проблемами, отладчик Oracle Power Objects показывает, где возникает эта проблема, и позволяет исследовать значения и пошагово пройти через код, чтобы локализовать ее.

## Внешний интерфейс: Наборы записей и связанные контейнеры

В клиентских системах с базой данных могут взаимодействовать два различных вида объектов: наборы записей и связанные контейнеры.

Набор записей (объект *recordset*) – набор строк, запрошенных из таблицы или представления, который хранится в памяти системы клиента. Обычно, для снижения информационной нагрузки сети и экономии памяти клиента, приложение хранит только подмножества всего набора записей. Когда требуется вывести на экран следующие записи или когда приложение должно выполнить некоторую агрегатную операцию, включающую все записи таблицы или представления, оно делает запрос большего количества строк.

Когда пользователь или приложение проводит изменения записей, изменения регистрируются сначала в наборе записей, но не в базе данных. В некоторый момент, приложение передает изменения в базу данных. В ходе этой процедуры приложение может устанавливать блокировки данных, предотвращая редактирования той же самой записи другими пользователями; оно может также отслеживать изменения, проведенные в записи с момента первоначального запроса.

Наборы записей не обязательно должны включать все столбцы из связанных таблиц или представлений. В большинстве случает, когда набор записей связывается с контейнером, он включает только те столбца, которые требуются для вывода на экран данных в текстовых полях, списках и других элементах управления контейнера.

Большинство наборов записей связывается с некоторым контейнером, хотя в Oracle Power Objects можно создавать также несвязанные наборы записей.

Связанный контейнер – объект интерфейса, который выводит на экран данные, полученные из таблицы или представления в результате запроса SQL. После выполнения запроса значения из столбцов набора записей отображаются в связанных элементах управления контейнера. В Oracle Power Objects. Связанным контейнером может быть форма, отчет, внедренная форма, пользовательский класс или окно репитера. Не все связываемые контейнеры должны быть связаны с источником записей: например, многие окна диалога – это формы, которые не имеют связанных таблиц или представлений. Однако, все связанные контейнеры имеют ассоциированные наборы записей, которые заполняются данными запросов базы данных при загрузке контейнера в память.

Внутри связанного контейнера располагаются связанные элементы управления, которые соединены со столбцами связанного набора записей. Например, в форме, предназначенной для работы с записями заказчиков, несколько текстовых полей выводят на экран имя заказчика, адрес, номер телефона и другую важную информацию, Информация в каждом текстовом поле происходит из различных столбцов набора записей.

Контейнер – лишь одно из многих типов объектов приложения. Определения объектов приложений сохраняются в файлах. РОА и, в конечном счете, компилируются в исполняемое приложение. Другие объекты приложения – элементы управления (например, командные кнопки), статические объекты (например, линии). OLE-объекты, импортируемые растровые рисунки и пользовательские классы.

## Разработка в объектно-ориентированной среде.

В дополнение к изменению характеристик отдельных объектов, необходимо также определить отношения между объектами. Например, при связывании формы с таблицей определяется отношение между объектом приложения и объектом базы данных.

Если данные, отбираемые из таблицы и представленные в форме, переупорядочиваются, отношение между этими двумя объектами переопределяется. Для командной кнопки, которая, будучи нажата, открывает форму, действует несколько иная зависимость: отношение между командной кнопкой и формой определяет код метода, который добавляется к методу Click() командной кнопки.

Каждый объект имеет большой интервал стандартных свойств и методов. Таким образом, характеристика, которую требуется получить, или действие, которое требуется выполнить, часто уже содержится в свойстве или методе. При необходимости, однако, можно также добавлять новые, пользовательские свойства и методы. Новое свойство может быть добавлено к объекту, когда разработчик решает, что для объекта требуется некоторая дополнительная характеристика (например, дополнительный уровень защиты при открытии формы). Новый метод добавляется, когда требуется определить новую задачу обработки, не охваченную стандартными методами (например, выполнение специфических бухгалтерских вычислений).

В сущности, объектно-ориентированная разработка лишь быстро фиксирует оформившееся представление разработчика относительно приложения. Зная набор средств того или иного инструментального продукта, перед выполнением фактической разработки программист рисует в своем воображении, какие объекты ему требуются, какие характеристики они должны иметь и как действие, которое выполняется над одним объектом (например, командной кнопкой) повлияет на другой (например, форму). Для работы с почти каждым объектом в Oracle Power Objects используется общая методология. Независимо от того, является объект сеансом базы данных или переключателем, для управления объектом разработчик определяет или изменяет одно из свойств или вызывает один из методов этого объекта.

## Прочие аспекты.

Помимо изложенного, объектно-ориентированное проектирование имеет три другие важные черты:

1. Модульность. В объектно-ориентированной среде разработки, объекты должны заключать в себе полное определение их характеристик. Никакие определения относительно методов и свойств объекта не должны располагаться где-либо вне данного объекта. Эта черта делает возможным копирование объекта в другие объекты – формы, отчеты, пользовательские классы, сеансы или приложения без потерь каких-либо показателей поведения, представления или других важных характеристик.
2. Иерархическая структура включения. В объектно-ориентированном приложении некоторые классы объектов могут содержать другие объекты. Например, форма может включать элементы управления, статические объекты и даже другие контейнеры (такие как окна репитеров). Объектно-ориентированные инструментальные средства разработки, позволяют определять, какие объекты будут содержать другие объекты, и выстраивать объектную иерархию сверху (приложение) до низу (например, командная кнопка внутри формы).
3. Возможность многократного использования. При проектировании объектов необходимо, насколько это возможно, предусматривать возможность их многократного использования. В Oracle Power Objects, пользовательские классы и растровые объекты – наиболее характерные виды объектов приложения многократного использования. Легко копировать между приложениями можно и другие объекты, например, формы и отчеты. В случае пользовательских классов, вначале определяется общий тип элемента управления, который затем тиражируется для любого приложения. Экземпляры класса могут быть точными копиями, но, в случае необходимости, они могут включать любые модификации. Возможности тиражирования пользовательского класса, практически, ничем не ограничены.

## Сервер: Сеансы и базы данных.

В Oracle Power Objects объект-сеанс создается на серверной части системы приложения базы данных. Сеансы обеспечивают доступ к объектам базы данных (таблицы, представления, последовательности и индексы), так и к данным, хранимым в самих таблицах и представлениях. В приложении можно использовать несколько сеансов; фактически, единственная форма, отчет или пользовательский класс могут обращаться к нескольким сеансам.

Посредством свойств и методов объекта-сеанса можно управлять режимом соединения с базой данных (открытое или закрытое). Устанавливая параметры логической структуры базы данных, определенной для сеанса, разработчик управляет диапазоном объектом базы данных, доступных через сеанс. Многие процессоры баз данных используют логические структуры (схемы пользователей) для ограничения доступа к объектам базы данных отдельным пользователям. Например, логическая структура администратора базы данных будет иметь доступ ко всем объектам базы данных, но рядовые пользователи должны иметь возможность редактировать данные только тех таблиц и представлений, которые нужны им для работы.

В Oracle Power Objects сеанс представляет собой отдельное окно, в котором символически показаны все объекты базы данных, доступные через этот сеанс. Дважды щелкнув на пиктограмме сеанса, можно активировать и деактивировать соединение с базой данных Когда соединение активно, окно Database Designer выводит все объекты базы данных, доступные в этом сеансе. Кроме того, при открытом соединении, во время разработки можно добавлять новые объекты базы данных и редактировать существующие.

### Базы данных

База данных – система, которая хранит и организует информацию. Oracle Power Objecs позволяет формировать приложения, которые обращаются к информации, сохраненной в реляционной базе данных, и коллекции объектов базы данных, включая таблицы, представления, индексы, последовательности и синонимы. Процессор базы данных координирует пользовательский доступ к информации базы данных.

В Oracle Power Objects пользователь взаимодействует с базой данных через сеанс базы данных – объект, который содержит информацию, необходимую для установления соединения с базой данных. В этой главе описываются объекты-сеансы базы данных и общие методы работы с ними.

Далее описываются типы баз данных, поддерживаемые Oracle Power Objects внутренние (базы данных Blaze) и внешние базы данных

Внутренние базы данных создаются и поддерживаются Oracle Power Objects. Базы данных Blaze компактны и эффективны, не требуют для выполнения большого количества системных ресурсов. Объекты базы данных Blaze хранятся на жестком диске в единственном файле.

Внешние базы данных создаются и поддерживаются процессором базы данных вне Oracle Power Objects, например, ORACLE7 Server.

Рассматриваемая версия Oracle Power Objects поддерживает следующие внешние базы данных

ORACLE7 Server. ORACLE7 Server – мощная, высокопроизводительная система управления реляционными базами данных Oracle Corp., которая эксплуатируется на широком разнообразии программно-аппаратных платформ.

Базы данных SQL Server. Надежные и совершенные базы данных SQL Server поставляются компаниями Microsoft и Sybase. Oracle Power Objects обеспечивает доступ к базам данных SQL Server через драйвер DBLIB.

Сеанс базы данных – объект, представляющий соединение между пользовательским приложением и базой данных. Сеанс базы данных, обеспечивает связь между внешним интерфейсом и серверной частью системы базы данных.

Каждый объект-сеанс базы данных сохраняется в собственном дисковом файле (в Windows этот файл имеет расширение POS). Объекты-сеансы базы данных представлены пиктограммой в окне Main рабочего стола Oracle Power Objects

Объект-сеанс может быть в одном из двух состояний: неактивный или активный.

Неактивный сеанс – не соединенный в данный момент с базой данных. Через неактивный сеанс нельзя вывести на экран объекты базы данных.

Активный сеанс – соединенный в данный момент с базой данных; он «содержит» все объекты базы данных, которые могут быть доступны через соединение. Активное состояние сеанса отображается пиктограммой включенного соединителя.

Пользователь обращается к объекту-сеансу так, как если бы он содержал объекты базы данных. Однако, объект-сеанс фактически не содержит никаких объектов. Вместо этого, он содержит информацию, необходимую для установления соединения базы данных – например, он может содержать имя пользователя, пароль и сетевой адрес учетной записи в базе данных сервера. Когда приложение запрашивает информацию из сеанса базы данных, он автоматически передает соответствующий запрос базе данных и обрабатывает результат.

Каждый объект-сеанс базы данных обеспечивает доступ к единственной логической структуре (схеме) или учетной записи пользователя в базе данных. Сеанс обеспечивает полный доступ к объектам и функциям учетной записи, хотя некоторые возможности базы данных доступны только через код SQL (например, в ORACLE7 Server необходимо писать пользовательский код SQL, чтобы создавать или обращаться к объекту-снимку). Во время разработки сеанс базы данных содержит пиктограммы, представляющие объекты базы данных, принадлежащие учетной записи. В окне сеанса присутствуют только наиболее общие типы объектов базы данных (таблицы, представления, индексы и последовательности). Сеанс не показывает объекты базы данных в других логических структурах или учетных записях, к которым пользователь имеет доступ.

Объекты-сеансы базы данных не зависят от объектов приложения, Эта независимость обеспечивает гибкость в конфигурировании соединения приложения с базами данных. Например, несколько приложений могут использовать один и тот же сеанс базы данных или единственное приложение может использовать одновременно много различных сеансов. Один сеанс базы данных можно также использовать для соединения в разное время с различными базами данных. Например, через сеанс можно в процессе разработки и отладки приложения соединяться с базой данных Blaze, а затем при развертывании законченного приложения в производственной среде через тот же сеанс подключиться к ORACLE7 Server.

При создании объекта-сеанса базы данных создается отдельный файл на жестком диске, содержащий информацию относительно того, как должно быть выполнено соединение с базой данных.

### Базы данных Blaze

База данных Blaze – компактная и эффективная реляционная база данных, которая создается и поддерживается Oracle Power Objects. Oracle Power Objects предоставляет все инструментальные средства, требуемые для создания и работы с базами данных Blaze.

Базы данных Blaze поддерживают многие из функций и возможностей больших реляционных баз данных, но требуют при работе значительно меньшего количества системных ресурсов. Они могут содержать основные объекты базы данных: таблицы, представления, индексы, последовательности и синонимы. Как и для большинства реляционных баз данных, обращение к ним осуществляется через SQL – язык программирования и реляционных обращений. Язык SQL, поддерживаемый Blaze – подмножество языка Oracle7 SQL с несколькими дополнительными чертами.

Каждая база данных Blaze сохраняется как единственный дисковый файл операционной системы (который в Windows имеет расширение BLZ). Этот файл содержит все объекты пользователя и данные, а также информацию, необходимую для соединений и разъединений базы данных в приложениях-клиентах. База данных Blaze может иметь одновременно только одно открытое соединение.

В отличие от некоторых других реляционных баз данных, Blaze не имеет никаких серверных выделенных структур памяти или процессов. Вся информация относительно базы данных, включая информацию по блокировкам данных и управлению транзакциями, хранится в файле базы данных. Все необходимые операции периода выполнения (т акие как анализ операторов SQL и чтение или запись данных) выполняются процессором базы данных Blaze, который встроен в Oracle Power Objects Designer и Oracle Power Objects Rum-time. Когда два различных приложения-клиента соединяются с одной базой данных Blaze, каждый клиент использует собственную копию процессора базы данных.

Базы данных Blaze идеальны для ситуаций умеренного доступа к данным, включая следующие применения:

* Локальные просмотровые таблицы. Если приложение обращается к внешним таблицам или представлениям, информация которых остается относительно статичной (например, список стран или категорий товаров), производительность системы базы данных можно повысить, сохраняя копии этих таблиц локально в базе данных Blaze. Часто обновляемые таблицы также могут быть сохранены локально, однако при этом необходимо организовать синхронизацию содержимого отдельных баз данных.
* Макетирование. При разработке приложения, которое предназначено для работы с большой корпоративной базой данных , для целей тестирования можно копировать определения объектов этой базы данных в базу данных Blaze. Это может ослабить нагрузку на расделяемые ресуры, повысить производительность разработки приложения и позволит работать, не соединяясь с сетью.
* Небольшие приложения. Для относительно небольших объемов (до 4 GB) данных, база данных Blaze формирует компактное, легко управляемое хранилище данных, которое поддерживает доступ одного пользователя.
* Разъединенный клиентский доступ. Базу данных Blaze можно использовать для хранения информации, разгруженной из большой системы базы данных; эту информацию пользователь может автономно исследовать и изменять в некритичном режиме.
* Базы данных только для чтения. База данных Blaze может поддерживать структуры информации на устройствах только для чтения, таких как CD-ROM.

Ниже приводятся ситуации доступа к данным, которые требуют дополнительных возможностей внешней базы данных:

* Большие требования к объему хранения данных. Базы данных Blaze ограничены максимальным доступным размером файла базовой операционной системы. Для Windows максимальный размер файла – 4GB.
* Параллельный доступ. Мощные внешние базы данных позволяют эффективно управлять большим количеством параллельных сеансов.
* Высоко-критичные приложения. Внешние базы данных обеспечивают расширенные возможности резервного копирования и восстановления данных – функции, которые являются жизненно важными для приложений повышенной критичности.
* Строгие требования защиты. Хотя базы данных Blaze поддерживают защиту с использованием пароля и шифрования данных, они не могут обеспечить такой же уровень защиты, как защищенная база данных, размещенная в защищенной операционной системе. Внешние базы данных также могут поддерживать такие дополнительные возможности защиты и управления, как ревизия, роли и защищенные логические структуры.
* Доступ из других платформ и инструментальных средств. В данный момент к базе данных Blaze можно обратиться только Oracle Power Objects. Если требуется использовать другие инструментальные средства доступа к базе данных или необходим доступ к базе данных из операционной системы, не поддерживаемой Oracle Power Objects, необходимо использовать внешнюю базу данных.

### ORACLE7 Server

ORACLE7 Server – система управления реляционными базами данных, доступная на большом количестве программно-аппаратных платформ. ORACLE7 Server обеспечивает качественные и эффективные решения по главным функциям базы данных, включая:

* Управление большими базами данных и пространствами
* Неограниченное (со стороны JRACLE7 Server) количество параллельных пользователей базы данных
* Высокая производительность обработки транзакций
* Высокая доступность
* Поддержка промышленных стандартов
* Управляемая защита
* Централизованно поддерживаемая целостность
* Поддержка среды клиент/сервер (распределенная обработка)
* Поддержка систем распределенных баз данных
* Мобильность
* Совместимость
* Стыкуемость

Oracle Power Objects обеспечивает полную поддержку функций и возможностей ORACLE7 Server. Однако, Oracle Power Objects не располагает инструментальными средствами для создания или управления базой данных ORACLE7 Server.

Как и большинству реляционных баз данных, обращение к ORACLE7 Server осуществляется посредством языка SQL. Ко многим функция ORACLE7 Server можно обратиться через Record Manager в Oracle Power Objects, а также с помощью свойств, методов и окон, ассоциированных с доступом к базе данных. Другие возможности ORACLE7 Server можно реализовать, выполняя через команду EXEC SQL пользовательские операторы SQL или PL/SQL.

### Базы данных SQL Server

База данных SQL Server – многопользовательская система управления реляционными базами данных, поставляемая фирмами Microsoft и Sybase. Как и ORACLE 7 Server, базы данных SQL Server эксплуатируются на широком диапазоне программно-аппаратных платформ и обеспечивают поддержку главных функций баз данных.

Типичная инсталляция SQL Server включает набор системных и пользовательских баз данных. Системные базы данных включают базы данных «master», «model» и «tempdb». Пользовательские базы данных создаются и поддерживаются, по мере необходимости, системным администратором SQL Server.

Oracle Power Objects в данный момент поддерживает любую базу данных SQL Server, которая может быть доступна через драйвер DBLIB. Некоторые функции баз данных Sybase System 10, включая поддержку курсов, через драйвер DBLIB не доступны.

Для таблиц, которые будут использоваться с Oracle Power Objects, необходмо всегда определять ограничения Primary Key.Oracle Power Objects использует значения первичного ключа для идентификации отдельных строк в операциях базы данных – например, при обновлении или удалении строк. Если приложение использует таблицы, которые не включают Primary Key, оно может вести себя непредсказуемо.

Драйвер DBLIB не включает поддержку нескольких параллельных курсоров. Следовательно, в каждый момент времени может быть обработан только один результирующий набор запроса – все результирующие строки запроса должны быть возвращены прежде, чем может быть выполнен второй запрос.

Устанавливая свойство RowFetchMode связанного контейнера, можно управлять порядком отбора результирующих строк. Когда свойство RowFetchMode установлено в «Fetch All Immediately», приложение будет вести себя идентично с базами данных всех типов. Однако, когда свойство RowFetchMode установлено в «Fetch as Needed» или «Fetch Count First», с базами данных SQL Server приложение может работать более медленно. Это происходит более медленно. Это происходит потому, что все не выбранные (не просмотренные пользователем) строки запроса должны быть довыбраны из базы данных, прежде чем можно будет вводить другие запросы.

Драйвер DBLIB непосредственно не предусматривает поддержку связанных переменных. Чтобы эмулировать поддержку связанной переменной, Oracle Power Objects автоматически заменяет упоминания связанной переменной в операторах EXEC SQL литеральными значениями данных.

Базы данных SQL Server не поддерживают объекты-последовательности. Следовательно, для генерирования уникальных значений таблиц (например, значений для столбца Primary Key) необходимо использовать альтернативную методику.

### Объекты базы данных

Объекты базы данных хранят и организуют информацию в реляционных базах данных. В Oracle Power Objects объекты базы данных представлены пиктограммами в окне сеанса базы данных.

Объекты базы данных, в отличие от объектов приложения, не создаются и не поддерживаются непосредственно Oracle Power Objects. Все процедуры, касающиеся объектов базы данных, выполняются процессором базы данных – компонентом базы данных, в которой хранятся объекты. Так как процессоры реляционных баз данных имеют различные возможности, доступные объектные типы и функции для разных баз данных различны.

Объекты базы данных формируют «сервер базы данных» приложения Oracle Power Objects. При разработке приложения объекты базы данных связываются с объектами приложения (такими, как формы и отчеты).Объекты приложения обеспечивают «окно» в объекты базы данных, предоставляя сохраненную информацию в полезном формате. Процесс соединения объектов базы данных с объектами приложения называется связыванием.

Объекты базы данных визуально содержатся внутри объекта-сеанса. В каждом окне сеанса представлены объекты, принадлежащие единственному пользователю базы данных.

В некоторых базах данных объекты каждого пользователя хранятся в отдельной логической структуре. Логическая структура – именованная коллекция объектов внутри базы данных. С каждым пользователем базы данных ассоциирована логическая структура того же имени. Например, пользователь NINA имеет логическую структуру NINA. Для баз данных, которые поддерживают логические структуры, каждый объект-сеанс базы данных обеспечивает доступ к единственной логической структуре пользователя.

Окно сеанса базы данных не обязательно показывает все объекты, доступные пользователю – в нем представлены только объекты, для которых пользователь является владельцем (объекты, созданные этим пользователем). В окне сеанса базы данных не показаны public синонимы или объекты, принадлежащие другим пользователям для доступа к которым текущий пользователь имеет привилегии.

Ниже приводятся типы объектов базы данных, присущие большинству баз данных, с которыми может взаимодействовать Oracle Power Objects:

1. Таблицы. Объекты базы данных, которые фактически хранят данные. Отдельная таблица чаще всего хранит информацию по конкретной теме (например, служащие компании или адреса заказчиков). Информация в таблице организована в строки и столбцы.
2. Представления. Настроенные обзоры данных из одной или больше таблиц. Представление – виртуальная таблица, которая позволяет связывать и объединять данные из несколько таблиц и представлений (называемых исходными таблицами). Представления, подобно таблицам, организованы в строки и столбцы; однако, представления непосредственно не содержат никаких данные – они создаются логически как результат определения в операторе SQL. Представления позволяют обрабатывать несколько таблиц и ли представлений как один объект базы данных.
3. Индексы. Обеспечивают быстрый доступ к отдельным строкам в таблице. Индексы хранят «указатели» на каждую строку в таблице в формате, оптимизированном для быстрой сортировки и поиска данные. Будучи создан, индекс автоматически поддерживается и используется базой данных всякий раз при обращении к индексированным столбцам.
4. Последовательности. Объекты, генерирующие ряд целых чисел, которые могут применяться для назначения уникальных идентификаторов строкам таблицы. Значения последовательности можно использовать, чтобы гарантировать, что столбец не содержит дублированных значений (например, столбец первичного ключа). Некоторые базы данных, такие какSQL Server, не поддерживают последовательности; для этих баз данных необходимы альтернативные методы формирования уникальных значений.
5. Синонимы. Псевдонимы объектов базы данных (таблиц, представлений и последовательностей). Синонимы могут обеспечивать public доступ к часто используемым объектам и могут скрывать расположение и владельца объекта.

Для работы с этими базовыми объектами базы данных Oracle Power Objects обеспечивает графические интерфейсы.

Внешние базы данных (такие, как ORACLE7 Server ) могут содержать ряд дополнительных объектов базы данных (такой как кластеры, пакеты, снимки и роли) которые часто используются для обеспечения дополнительных уровней защиты или повышения эффективности системы базы данных, Чтобы обратиться к этим объектам из Oracle Power Objects, необходимо выполнить команды SQL, используя команду Oracle Basic EXEC SQL или функцию SQLLOOKUP.

Объекты базы данных не имеют таких свойств и методов, какие имеются у объектов приложения, так как они не создаются посредством объектных механизмов Oracle Power Objects.У объектов базы данных имеются ассоциированные листы свойств, в которые, однако. Нельзя добавлять пользовательские свойства или методы. Большинство объектов базы данных имеет лишь свойство Name, которое предусмотрено для обращений разработчика. Свойство Name может быть изменено через лист свойств во время разработки, и при этом объект будет переименован в базе данных, но это свойство нельзя изменять посредством Oracle Basic в период выполнения.

Над объектами базы данных модно выполнять два общих типа операций: операции определения данных и операции манипулирования данными.

Операции определения данных манипулируют структурой объекта базы данных. Они включают создание, удаление и изменение структуры объектов базы данных. Обычно эти операции выполняются проектировщиком во время разработки.

Операции манипулирования данными управляют данными, сохраненными в объекте или доступными через объект. Они включают запросы, вставку, обновление и удаление строк данных. Операции манипулирования данными применяются, главным образом, к таблицам и представлениям, хотя, иногда они используются с другими объектами базы данных, такими как последовательности. Эти операции могут выполняться как разработчиком, так и пользователем в период выполнения.

При создании, удалении или изменении объекта базы данных из Oracle Power Objects, специфицированные изменения автоматически преобразуются в операторы SQL, которые затем передаются для выполнения процессору базы данных. Непосредственно Oracle Power Objects не выполняет никаких модификаций объектов и их данных.

Каждый тип операции имеет ассоциированный набор команд SQL: операции определения данных используют команды Языка Определения Данных (DDL), в то время как операции манипулирования данными используют команды Языка Манипулирования Данными (DML).

Типы операций, которые пользователь может выполнять с объектом базы данных, определяются привилегиями, которые он имеет по данному объекту. По умолчанию, владелец объекта(пользователь, который создал объект) имеет все привилегии по объекту. Для других пользователей, желающих обратиться к объекту, владелец должен предоставить соответствующие привилегия.

Доступные типы объектных привилегий для разных баз данных различны, что отражается в соответствующем синтаксисе SQL для представления или отмены привилегий. Базы данных Blaze не имеют объектных привилегий – все пользователи базы данных Blaze имеют привилегии для всех объектов в базе данных.

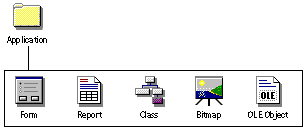
Предоставление или отмена привилегий выполняется через операторы SQL. Информация относительно предоставления и отмены привилегий приводится в документации по конкретной базе данных.

Имена объектов базы данных должны отвечать правилам именования объектов для базы данных, в которой они сохранены. Эти правила различны для разных баз данных.

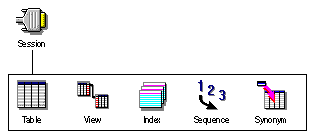
## Среда разработки

Среда разработки OPO внешне напоминает ставшую уже стандартной среду Visual Basic. Верхний уровень иерархии объектов OPO - это объект ы File, изображенных в виде иконок на основном окне среды разработки и которых может быть три типа :

**Приложения** (Application).



**Объекты Базы Данных** (Database session).

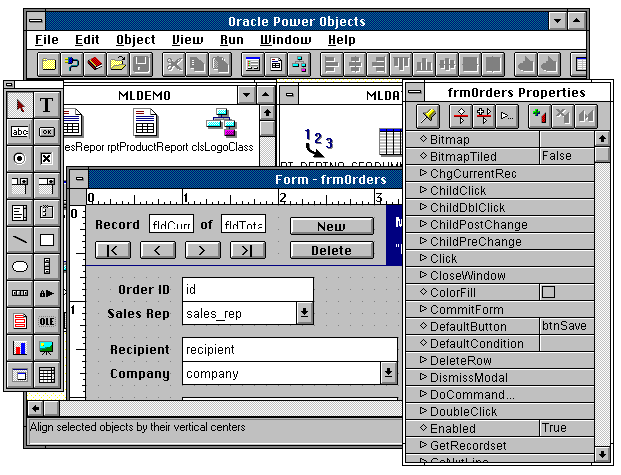


**Библиотеки** (Library).



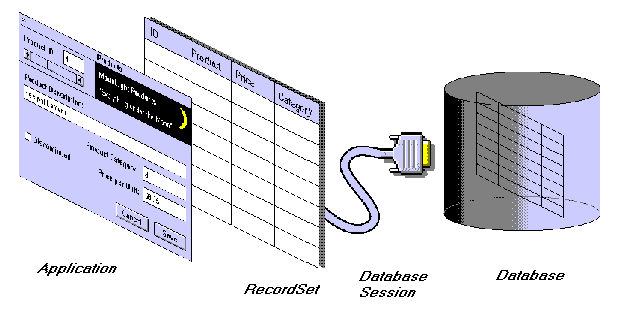
Каждая такая иконка может быть раскрыта в окно, содержащее объекты нижнего уровня иерархии. Копирование объектов между однотипными группами осуществляется простым переносом иконок .

При создании и редактировании объектов появляется инструментальная линейка и окно редактирования свойств и методов.



В процессе разработке можно запускать на выполнение как все приложение, так и отдельные экраны (формы) и отчеты. Большую помощь при отладке может оказать отладчик, функционально аналогичный отладчику в Visual Basic.   
Важно отметить возможность работать с объектами Базы Данных (таблицами, представлениями, индексами и т.д.) непосредственно из среды разработки OPO.   
Вместе с тем, представление объектов в виде иконок и раскрывающихся окон удобно только при весьма небольшом их количестве. Когда число используемых объектов превышает несколько десятков , становиться очень сложно ориентироваться среди них и разбираться среди множества раскрытых и мешающих друг другу окон. Реализованный в Forms 4.5 Навигатор Объектов [3] заметно более нагляден и удобен. Большое число иконок в окне, а число объектов (таблиц, представлений , индексов и последовательностей) в даже небольшой БД превышает несколько сотен, в Windows 3.1/3.11 вызывает истощение системных ресурсов и приводит к ее краху.

## Структура приложения

Приложение можно разделить на четыре структурные части :   


Application и RecordSet представляют обращенную к пользователю часть приложения (Front End), Session и Database являются частью выполняющегося на сервере приложения (Back End). В зависимости от задач, требований к сетевому трафику и характеристик аппаратуры необходимо распределять логику работу приложения между этими частями. Обработку транзакций и поддержку целостности данных можно осуществлять во всех структурных частях приложения.   
Основной частью (и видимой пользователю) частью приложения является *форма* или *экран* (Form). При создания экранов можно применять все стандартные элементы Windows приложений - поля и метки, различного рода кнопки, списки и т.д. Экраны могут содержать вложенные экраны.   
Удобным является механизм *запроса по форме* (Query By Form). Для отображаемого в данный момент экрана можно вызвать экран для ввода запроса, представляющий собой копию первого экрана, в поля которого можно ввести условия и критерии выборки. В результате на первом экране будет информация в соответствии с заданными условиями. При этом экран запроса тоже остается видимым.

## Обработка транзакций

Любая транзакция должна пройти через несколько уровней, каждый из которых играет свою роль в управлении обработкой транзакций.

1. Контейнеры. Контейнер – элемент интерфейса, через который пользователь инициирует запросы и изменения набора записей. После добавления, удаления или изменения записи пользователь перед закрытием контейнера должен фиксировать или аннулировать транзакцию (обычно, нажимая кнопку, предназначенную для этой цели – чаще всего, это кнопки ОК и Отмена (Cancel)). Кроме того, здесь определяют устанавливаемые на уровне клиента бизнес-правила, ограничивающие транзакции, которые может инициировать пользователь. Если система должна сообщить пользователю относительно результатов предпринятой транзакции, информация появляется здесь же, в интерфейсе пользователя.
2. Наборы записей. Прежде, чем пользователь сможет провести изменения в наборе записей, приложение выполняет проверку ссылочной и объектной целостности данных. Кроме того, приложение регулирует объем информационной нагрузки сети, при необходимости выборочно выполняя запрос строк базы данных, чтобы заполнить набор записей.
3. Сеансы. После прохождения уровней приложения и набора записей транзакция должна затем пройти через сеанс. Чтобы фиксировать или аннулировать транзакции, ассоциированные с данным сеансом, для объекта-сеанса могут вызываться несколько методов.
4. Базы данных. В конечном счете, инициированная пользователем транзакция достигает непосредственно базы данных, где процессор базы данных проверяет, завершена ли транзакция. Процессор базы данных может установить на сервере бизнес-правила, контролируя, не нарушает ли переданная транзакция правила защиты, и выполнять другие важные функции управления транзакциями. Если транзакция – запрос, процессор базы данных после этого передает требуемую информацию клиенту.

Следовательно, при управлении транзакциями внутри приложения клиент/сервер Oracle Power Objects предоставляет возможности установления бизнес-правил, регулирования информационной нагрузки сети и поддержания целостности данных. Например, отдельное бизнес-правило может устанавливаться как ограничение базы данных или как часть приложения-клиента.

## Подход к разработке, реализуемый в Oracle Power Objects

Хороший стиль проектирования требует принятия важных решений в самом начале, с тем, чтобы исключить необходимость повторения значительных объемов работы.

В зависимости от вида разрабатываемого приложения и используемого инструментария, разработка начинается с сервера базы данных или с внешнего интерфейса. Средства разработки Oracle Power Objects позволяют сначала разработать интерфейс пользователя, построив формы, отчеты и другие компоненты приложения-клиента. После того, как интерфейс пользователя завершен, при необходимости, можно сформировать под него объекты базы данных и затем соединить компоненты интерфейса с их связанными таблицами и представлениями, Обычно, если проектировщик работает с приложением, в котором для соединения компонентов внешнего интерфейса с сервером базы данных требуется значительный объем программирования, он откладывает написание этих процедур, пока не будет удовлетворен интерфейсом пользователя и пока не будут определены все объекты базы данных. В противном случае, при переопределении объектов базы данных или компонентов интерфейса ему придется тратить много времени на редактирования и переделки кода программы.

С другой стороны, высококачественные инструментальные средства клиент/сервер часто требуют, чтобы их проектирование было начато с построения объектов базы данных, а затем уже был сформирован внешний интерфейс. Учитывая как сложность и важность правильной организации объектов базы данных, так и трудность управления отношениями среди них, при разработке этих видов приложений сервер базы данных должен иметь приоритет.

В реальной жизни, разработчики часто переключаются между сервером базы данных и внешним интерфейсом, вместо исключительного проектирования первым того или другого, (Однако, обычно, все равно один раздел приложения базы данных сначала получает акцент, даже если впоследствии ему не уделяется исключительное внимание). Если проектировщик имеет ясную совокупную картину требуемых объектов и их отношений, Oracle Power Objects помогает существенно упростить этот вид инкрементной разработки.

### Если начать с клиента

Обычно, с разработки внешнего интерфейса начинают, когда…

* Необходимо применить в приложении интерфейса клиента какие- либо производственные стандарты, например, для финансовых расчетов.
* Необходимо сосредоточить усилия на обеспечении правильного выполнения приложением последовательности действий некоторого процесса.
* Используются уже имеющиеся объекты базы данных, для которых необходимо просто сформировать интерфейс. Однако, при этом, все же, необходимо проверить, не требуются ли в связи с этим в базе данных некоторые небольшие модификации. Например, может быть принято решение о том, что инициирование отдельного бизнес-правила можно перенести с сервера на клиент, устраняя тем самым потребность в хранимой процедуре сервера, разработанной для этой задачи.
* Приложение не предназначено для интенсивного доступа к базе данных, так что можно уделить больше времени интерфейсу пользователя.

В этих случаях Oracle Power Objects позволяет по отдельности проектировать и тестировать различные формы и отчеты, которые составляют интерфейс пользователя. Следовательно, разработка приложения, которая начинается с внешнего интерфейса, может выполняться пошагово или фронтом: разработчик может или проектировать компоненты приложения последовательно или работать с несколькими компонентами одновременно. На этом этапе, модификации объектов не составляют проблем и выполняются относительно просто, так как доступ к базе данных еще не полностью реализован. После создания форм, отчетов, классов и других объектов приложения решаются вопросы навигации между ними и добавляется программный код, в котором устанавливаются бизнес-правила и выполняются задачи обработки данных (вычисления).

Объекты интерфейса и их связанные таблицы или представления можно также разрабатывать параллельно, так как таблицы и представления структурно часто дублируют формы и отчеты, в которых выводятся их записи.

### Если вначале разрабатывается внешний интерфейс, следует ответить на следующие вопросы:

* Какими будут главные формы, которые пользователь увидит на экране? Они будут, вероятно первыми объектами, которые должны быть спроектированы, наряду с их связанными таблицами или представлениями.
* Какая модель последовательности действий будет заложена в приложение? Иными словами, следует тщательно продумать, как легко пользователь сможет вводить данные, осуществлять навигацию между формами и выполнять другие операции внутри приложения. Кроме того, необходимо оценить, как приложение организует работу пользователя и задает ли оно разумный темп при решении задач.
* Какие объекты должны быть определены вне приложения Oracle Power Objects и затем импортированы? Например, если планируется добавлять растровые образы или другие OLE-объекты, возможно, вначале придется разработать некоторые из этих ресурсов приложения.
* Где лучше и как лучше установить ограничения? Например, если требуется гарантировать, чтобы транзакция, введенная в приложении регистрации заказов, не содержала значений, превышающих некоторый объем, возможно, будет лучше установить это ограничение на клиенте через код Oracle Basic, а не на сервере через триггер. Для этого потребуется добавить необходимый код Oracle Basic, который будет прерывать транзакцию до ее фиксации, если она превышает установленный объем заказа.
* Как лучше представить документ? В данном случае, «документ» - отдельный объект, такой как заказ на приобретение, который заполгяеься в приложении. Можно поместить каждое поле, которое будет содержать данные, относящиеся к заказу, на одной форме. Однако, для удобочитаемости, возможно, лучше будет разбить «мега-форму» на несколько меньших форм.
* Какие компоненты интерфейса будут повторяться в приложении? Если имеются объекты, неоднократно появляющиеся в приложении, вероятно, их следует проектировать как пользовательские классы, сохраненные или в приложении или библиотеке, Экземпляры пользовательских классов мажно добавлять к формам и отчетам, вместо того, чтобы много раз генерировать одни и те же объекты. Например, если проектируется пользовательский набор средств управления для просмотра базы данных, следует создать их как класс, чтобы экземпляры одного класса легко могли наследовать изменения в исходном классе.

Главный недостаток первоначальной разработки внешнего интерфейса заключается в том, что проектирование базы данных – обычно, одна из наиболее важных задач при разработке программного обеспечения, но в данном случае к интерфейсу пользователя приковано основное внимание разработчика. То, что хорошо смотрится в экранной форме, может быть трудно выразимо с помощью таблицы или представления или даже нескольких связанных таблиц.

Основные функции Oracle Power Objects, которые позволяют начать разработку с внешнего интерфейса, включают:

* Инструментальные средства GUI (графический пользовательский интерфейс) для быстрого создания новых объектов внешнего интерфейса.
* Объекты приложений многократного использования, созданные как пользовательские классы или объекты библиотек.
* Возможность всесторонне тестировать отдельные компоненты интерфейса пользователя, такие как главная форма приложения, до перехода к созданию других объектов приложения.
* Отладчик периода выполнения, который может запрашивать свойства, тестировать код методов и выполнять другие важные проверки.

### Если начинать с сервера базы данных

При этом подходе, прежде, чем приступить к формированию интерфейса, проектировщик начинает с разработки модели данных (разработка всех таблиц и представлений). Необходимость начать с объектов базы данных может быть обусловлена несколькими причинами:

* Приложение будет использовать большое количество таблиц и представлений сложной структуры. Проектировщики, которые работают с реляционными базами данных, знают, как важно иметь ясное представление относительно объектов базы данных и их отношений. Этим в дальнейшем предотвращаются проблемы, вызванные сколько-нибудь значительными корректировками этих объектов. Если в одной из таблиц отношения один-к-многим изменяется тип данного для ключевого значения, это может разрушить отношение между главной и подчиненными таблицами.
* На сервере будет устанавливаться много ограничений. В таких случаях, до разработки интерфейса пользователя имеет смысл определить объекты базы данных, а также триггеры и хранимые процедуры, которые из защищают. Затем уже можно переходить к проектированию клиента, где эти ограничения будут использоваться. Примером использования ограничений может быть генерирование на сервере кодов ошибок и передача их в читабельном виде пользователю.
* Интерфейс пользователя – лишь окно в базу данных. В случаях, когда поля формы – простое отображение таблиц и представлений сервера, проектированию интерфейса можно уделить меньше времени.
* Защита сервера – приоритетная задача.
* Для повышения производительности требуется вначале произвести на компонентах сервера специальные процедуры (например, индексирование или нормализацию таблиц).
* Одни и те же таблицы и представления используются несколькими различными внешними интерфейсами.
* Приложение использует сложные отношения один-к-многим или вычисляемые значения. В таких случаях требуется тщательно спроектировать таблицы и представления, чтобы отношения один-к-многим могли быть легко представлены внутри приложения. Кроме того, правильно построенная модель данных сэкономит время при работе приложения за счет уменьшения сложности уравнений, оперирующих данными.

Если проектирование начинается с сервера, имеется возможность сформировать эффективную модель данных, отражающую информацию из реальной жизни. Объектом реальной жизни может быть любой объект (например, данные служащего, транзакция бухгалтерской книги, позиция инвентарной ведомости и т.д.), который требуется описать в одной или больше таблиц.

### Начиная проектирование с сервера базы данных, необходимо ответить на следующие вопросы:

* Какие требуются объекты базы данных? Иными словами, что будет представлять собой модель данных?
* Как следует оптимизировать структуру данных с точки зрения повышения производительности их обработки?
* Какие таблицы или представления будут основными? Почти в каждой модели данных некоторые таблицы более важны, нежели другие. Следовательно, необходимо рассмотреть весьма вероятное событие, когда в сети клиент/сервер к этой таблице попытается обратиться много пользователей. Кроме того, необходимо предусмотреть меры защиты важных данных от разрушительных изменений (например, модификаций ключевых значений в отношении один-к-многим), а также от несанкционированного доступа. Для реализации этих мер имеется широкий диапазон средств – от определения пользовательских логических структур, ограничивающих доступ к объектам базы данных, до написания триггеров, которые при некоторых условиях предотвращают проведение изменений в базе данных.
* Какие бизнес-правила целесообразно установить на сервере? Здесь необходимо балансировать между нежелательностью перегрузки сервера работой по обслуживанию каждого бизнес-правила и необходимостью установки на сервере важных ограничений, которые должны гарантировать целостность и согласованность данных приложений всех клиентов.

## Как проектировать пользовательские классы и библиотеки

Прежде, чем приступать к разработке приложения, необходимо рассмотреть возможность неоднократного использования в ходе разработки некоторых объектов или наборов объектов. Вот некоторые примеры таких объектов:

* Эмблема компании, которая присутствует на многих формах и отчетах.
* Набор средств управления, используемых для навигации по записям.
* Набор элементов управления, используемых для фильтрования и сортировки записей.
* Группа переключателей, используемых для обеспечения стандартного набора опций в нескольких формах
* Ряд текстовых полей, выводящих в нескольких формах информацию заказчика, продавца или компании.

Создавая эти объекты как пользовательские классы, сохраненные в приложении или в библиотеке, разработчик получает следующие преимущества:

* После того, как объекты созданы, добавление их к форме может выполняться простой операцией drag-and-drop.
* В Oracle Power Objects экземпляры пользовательского класса или объекты библиотеки наследуют свойства и методы определения исходного класса. Точно также экземпляры наследуют изменения, проводимые в этих определениях. Следовательно, изменение часто используемых объектов приложения выполняется намного быстрее, если они создаются как экземпляры. При этом, чтобы провести изменение по всем экземплярам, требуется изменить лишь определение пользовательского класса или объекта библиотеки.

Решение по созданию объекта приложения многократного использования как пользовательского класса принимается по результатам ответа на вопрос: этот объект будет использоваться один раз только в данном приложении, или, возможно, он потребуется в нескольких приложениях или несколько раз в одном приложении?

* Если предполагается использовать объект в нескольких приложениях, целесообразно создать его как объект библиотеки (или как объект-библиотеку). Когда потребуется изменить объект, он выделяется в собственную библиотеку, подобную библиотеке динамической компоновке (DLL) в среде Windows.
* Если планируется использовать объект только в одном приложении, можно создать его как пользовательский класс, При хранении пользовательского класса в приложении уменьшается количество поддерживаемых объектов-файлов.
* Oracle Power Objects позволяет создавать как объекты библиотеки пользовательские классы и растровые рисунки. Пользовательские классы – наиболее применяемый тип объектов библиотеки, так как они обеспечивают все функциональные возможности как контейнеров, так и элементов управления.

# Заключение

Суммируя вышеизложенное:

* База данных содержит объекты, предназначенные для хранения и организации информации.
* Сеанс обеспечивает доступ к базе данных, а также к коллекции объектов базы данных, определенных логической структурой (схемой) пользователя базы данных.
* Приложения внешнего интерфейса выполняют запросы базы данных, заполняя информацией наборы записей.

Наборы записей обычно связываются с объектами-контейнерами, которые непосредственно предоставляют пользователю доступ к данным, хранимым в базе данных.

Несмотря на то, что Oracle Power Objects обеспечивает обширный набор средств для разработки любых видов приложений, этот инструмент ориентирован и предназначен, в первую очередь, для формирования приложений базы данных в среде клиент/сервер. Используя один и тот же внешний интерфейс в системе клиента, можно выбрать любую систему управления базами данных по критериям защиты, эффективности, масштабируемости и другим важным функциям. Кроме того, Oracle Power Objects позволяет устанавливать и поддерживать специфические требования к данным средствами как приложения-клиента, так и сервера базы данных.

OPO - новый продукт на рынке средств разработки приложений клиент/сервер. У него нет за спиной богатой истории развития и как чисто инструментального языка (Visual Basic, Delphi) и как инструмента создания приложений для системы клиент/сервер (Forms). Первую версию скорее можно воспринимать как заявку на место на рынке, как демонстрацию новых идей. Если будет улучшен пользовательский интерфейс Среды разработки, введена полная поддержка OLE и особенно OLE Automation, будут созданы драйверы для других источников данных, при разработке можно будет использовать репозитарий приложения, то OPO действительно может стать весьма интересным продуктом.

# Список литературы

1. Г.Анзер: «Oracle Power Objects», Мосва, АБФ, 1997
2. www.oraclub.trecom.tomsk.su/articles/mir-ora/program/23.htm