Краткое практическое руководство разработчика информационных систем на базе СУБД Oracle

Власов Андрей Игоревич, к.т.н., кафедра "Конструирование и технология производства электронной аппаратуры" МГТУ им.Н.Э.Баумана, iu4.bmstu.ru. Лыткин Сергей Леонидович, выпускник факультета Вычислительной математики и кибернетики МГУ. Яковлев Владимир Леонидович, кафедра "Автоматизированные информационные системы" МГТУ им.Н.Э.Баумана, iu5.bmstu.ru.

Краткое практическое руководство разработчика информационных систем на базе СУБД Oracle: Библиотечка журнала "Информационные технологии" - М.: изд-во Машиностроение, 2000. - 120 с. ил.

В руководстве приведены краткие сведения о теории реляционных баз данных, практическая методика создания, эксплуатации и администрирования информационных систем на основе СУБД Oracle версий 7.x, 8.x. Рассмотрены основные элементы языка PL/SQL и словаря данных СУБД Oracle версий 7.x, 8.x. Основные принципы работы с синтаксическими конструкциями языка и элементами словаря данных рассмотрены на конкретных практических примерах. Руководство рассчитано на разработчиков прикладных систем под СУБД Oracle версий 7.x, 8.x., студентов, аспирантов и преподавателей, специализирующихся в разработке информационных систем и баз данных архитектур клиент/сервер и интернет/интранет.

Изложенные в пособии материалы являются обобщением материалов лекционных курсов: "Автоматизированные банковские системы: проектирование и эксплуатация" и "Разработка автоматизированных систем управления конструкторско-технологическим проектированием на основе СУБД Oracle", читаемых в МГТУ им.Н.Э.Баумана.

СОДЕРЖАНИЕ

Предисловие

Бурная информатизация общества, автоматизация технологических процессов, широкое использования вычислительной техники, средств связи и телекоммуникаций ставит перед современным менеджером, инженером и служащим целый комплекс взаимосвязанных задач по повышению эффективности бизнес - процессов принятия и выполнения решений.

На сегодня без использования современных автоматизированных информационных управляющих систем трудно представить себе ни учебный процесс в школе, институте, университете, ни эффективную работу практически в любой фирме, на предприятии, в банке или в госучреждении. И практически везде информационная система представляет собой интегрированную систему, ядро которой составляет база данных.

На сегодня издано огромное число различных монографий и учебников, описывающих те или иные вопросы проектирования информационных систем, теорию и практику использования СУБД и т.п. При этом успех любого издания определяется удачным сочетанием необходимого количества теоретических сведений и практических вопросов. Лишь только при таком подходе возможно дать читателю ответы на вопросы "Для чего?" и "Как?" создавать ту или иную информационную систему.

Изложенные в пособии материалы являются обобщением материалов лекционных курсов: "Автоматизированные банковские системы: проектирование и эксплуатация" кафедры "Автоматизированные информационные системы" (ИУ-5, http://iu5.bmstu.ru) и "Разработка автоматизированных систем управления конструкторско-технологическим проектированием на основе СУБД Oracle" кафедры "Конструирование и технология производства ЭА" (ИУ-4, http://iu4.bmstu.ru), читаемых в МГТУ им.Н.Э.Баумана.

Надеемся, что изложенные в данном руководстве базовые концептуальные подходы к построению информационных систем, результаты исследований типовых архитектурных решений, подкрепленные практическими примерами и краткое практическое руководство по инструментальным средствам создания современных информационных систем управления на базе СУБД Oracle, окажется полезным, как студенту, аспиранту и преподавателю, специализирующимся в области автоматизированных систем управления, так и специалисту-разработчику корпоративных и банковских информационных систем.

Введение в проектирование информационных систем

1.1. Методы проектирования информационных систем

Индустрия разработки автоматизированных информационных систем управления родилась в 50-х - 60-х годах и к концу века приобрела вполне законченные формы. Материалы данного руководства являются обобщением цикла лекций по Автоматизированным Банковским Системам (АБС) и Автоматизированным системам управления конструкторско-технологическим проектированием (АСУ КТП), читаемым в МГТУ им.Н.Э.Баумана. Не смотря на имеющиеся различия в реализации функциональных модулей данных систем, общие подходы к их разработки во многом схожи, что позволило нам объединить вопросы их проектирования в рамках одного издания.

На рынке автоматизированных систем для крупных корпораций и финансово-промышленных групп на сегодня можно выделить два основных субъекта: это ранок автоматизированных банковских систем (АБС) и рынок корпоративных информационных систем промышленных предприятий. Не смотря на сильную взаимосвязь этих двух рынков систем автоматизации, предлагаемые на них решения пока еще не достаточно интегрированы между собой, чего следует ожидать в недалеком будущем.

В дальнейшем под Автоматизированной Банковской Системой (АБС) будем понимать комплекс аппаратно-программных средств реализующих мультивалютную информационную систему, обеспечивающую современные финансовые и управленческие технологии в режиме реального времени при транзакционной обработке данных.

Под Автоматизированной Информационной Системой промышленного предприятия (АСУ КТП) будем понимать комплекс аппаратно-программных средств реализующих мультикомпонентную информационную систему, обеспечивающую современное управление процессами принятия решений, проектирования, производства и сбыта в режиме реального времени при транзакционной обработке данных.

Как вы видите, оба определения достаточно схожи. На сегодня существования нескольких методов построения автоматизированных информационных систем (АИС), среди которых можно выделить следующие:

1.1.1 Метод "снизу-вверх".

Менталитет российских программистов сформировался именно в крупных вычислительных центрах (ВЦ), основной целью которых было не создание тиражируемых продуктов, а обслуживание сотрудников конкретного учреждения. Этот подход во многом сохранялся и при автоматизации и сегодня. В условиях постоянно изменяющихся законодательства, правил ведения производственной, финансово-хозяйственной деятельности и бухгалтерского учета руководителю удобно иметь рядом посредника между спущенной сверху новой инструкцией и компьютером. С другой стороны, программистов, зараженных "вирусом самодеятельности", оказалось предостаточно, тем более что за такую работу предлагалось вполне приличное вознаграждение.

Создавая свои отделы и управления автоматизации, предприятия и банки пытались обустроиться своими силами. Однако периодическое "перетряхивание" инструкций, сложности, связанные с разными представлениями пользователей

об одних и тех же данных, непрерывная работа программистов по удовлетворению все новых и новых пожеланий отдельных работников и как следствие - недовольство руководителей своими программистами несколько остудило пыл как тех, так и других. Итак, первый подход сводился к проектированию **"снизу-вверх"**. В этом случае, при наличии квалифицированного штата программистов, вполне сносно были автоматизированы отдельные, важные с точки зрения руководства рабочие места. Общая же картина "автоматизированного предприятия" просматривалась недостаточно хорошо, особенно в перспективе.

1.1.2. Метод "сверху-вниз".

Быстрый рост числа акционерных и частных предприятий и банков позволил некоторым компаниям увидеть здесь будущий рынок и инвестировать средства в создание программного аппарата для этого растущего рынка. Из всего спектра проблем разработчики выделили наиболее заметные: автоматизацию ведения бухгалтерского аналитического учета и технологических процессов (для банков это в основном - расчетно-кассовое обслуживание, для промышленных предприятий - автоматизация процессов проектирования и производства, имеется в виду не конкретных станков и т.п., а информационных потоков). Учитывая тот факт, что ядром АИС безусловно является аппарат, обеспечивающий автоматизированное ведение аналитического учета, большинство фирм начали с детальной проработки данной проблемы. Системы были спроектированы "сверху", т.е. в предположении что одна программа должна удовлетворять потребности всех пользователей.

Сама идея использования "одной программы для всех" резко ограничила возможности разработчиков в структуре информационных множеств базы данных, использовании вариантов экранных форм, алгоритмов расчета и, следовательно, лишила возможности принципиально расширить круг решаемых задач - автоматизировать повседневную деятельность каждого работника. Заложенные "сверху" жесткие рамки ("общие для всех") ограничивали возможности таких систем по ведению глубокого, часто специфического аналитического и производственно - технологического учета. Работники проводили эту работу вручную, а результаты вводили в компьютер. При этом интерфейс каждого рабочего места не мог быть определен функциями, возложенными на пользователя, и принятой технологией работы. Стало очевидно, что для успешной реализации задачи полной автоматизации банка следует изменить идеологию построения АИС.

1.1.3. Принципы "дуализма" и многокомпонентности.

Развитие банковских структур и промышленных предприятий, увеличение числа филиалов, рост количества клиентов, необходимость повышения качества обслуживания предъявляли к автоматизированным системам новые требования. Новый подход к проектированию АИС заключается в сбалансированном сочетании двух предыдущих. В первую очередь это относилось к идеологии построения ядра системы: "Автоматизированная бухгалтерия - аналитический учет".

Для банковских структур это дало: с одной стороны, в ядре системы сохранялась возможность работы "от лицевого счета", с автоматическим формированием соответствующих бухгалтерских проводок, с другой стороны, отменялись жесткие требования работы только с лицевыми счетами. Появилась возможность ведения бухгалтерского учета по балансовым счетам любого порядка без углубления до уровня лицевых счетов клиентов. При этом ведение

аналитического учета по лицевым счетам клиентов опускалось на уровень специализированного программного обеспечения (СПО), установленного на рабочих местах банковских работников (контролеров, кредитных бухгалтеров, инспекторов и т. д.). Таким образом, принципиальное отличие нового подхода к созданию АБС заключается в идее распределения плана счетов по уровням экспертизы. При этом и сам справочник плана счетов с соответствующими описаниями, и информационное множество клиентов проектировались по принципу распределенной базы данных. Результатом этого явилось:

- формирование всех необходимых бухгалтерских проводок, уже агрегированных по балансовым счетам, и автоматическая их передача в базу данных "Автоматизированной бухгалтерии";
- реализация специфических требований каждого банковского работника, в том числе по формированию произвольных отчетов и справок, мемориальных ордеров, операционных дневников; выполнение любых вспомогательных и технологических расчетов и пр.

С использованием гибкой системы настроек СПО (компонентов АБС) появилась реальная возможность адаптации программного аппарата к практически любым условиям и различным требованиям инструктивных материалов и правилам работы, принятым либо в вышестоящей организации, либо в данном банковском учреждении. Кроме того, при многокомпонентной схеме организации АБС при проведении модернизации одного из компонентов центральная часть (ядро) АБС и другие ее компоненты не затрагивались, что значительно повышало надежность, продолжительность жизни автоматизированной системы и обеспечивало наиболее полное выполнение требуемых функций.

Двойственный подход к формированию ежедневного баланса лег в основу т.н. "принципа дуализма" - одного из важных принципов построения современных банковских систем. Реализация принципа дуализма неизбежно требовала построения АБС нового поколения в виде программных модулей, органически связанных между собой, но в то же время способных работать и автономно.

Задача проектирования АИС промышленных предприятий более сложна, т.к. характер обрабатываемой информации еще более разнороден и сложно формализуем. Однако и здесь можно выделить основную модель работы - это работа "от кода проекта". В общем случае код проекта представляет собой аналог (функциональный) лицевого счета, он имеет определенную разрядность, порядок (т.е. конкретная группа цифро-буквенного обозначения характеризует деталь, сборочную единицу, изделие и их уровень взаимосвязи). Причем конкретная часть кода характеризует технологические, конструкторские, финансовые и др. документы. Все это регламентируется соответствующими ГОСТами (аналог инструкций ЦБ для банков), поэтому может быть формализовано. При этом модульный подход к реализации АИС в этом случае еще более важен.

Двойственный подход к формированию ежедневного производственного плана лег в основу т.н. "принципа дуализма" для АИС промышленных предприятий. Реализация принципа дуализма неизбежно также требовала построения АИС предприятий нового поколения в виде программных модулей, органически связанных между собой, но в то же время способных работать и автономно.

Такая многокомпонентная система обеспечивала соблюдение основополагающего принципа построения автоматизированных информационных систем - отсутствия дублирования ввода исходных данных. Информация по операциям, проведенным с применением одного из компонентов системы, могла быть использована любым

другим ее компонентом. Модульность построения АИС нового поколения и принцип одноразового ввода дают возможность гибко варьировать конфигурацией этих систем. Так, в банках, имеющих разветвленную филиальную сеть и не передающих данные в режиме реального времени, установка всего СПО во всех филиалах не всегда экономически оправдано. В этих случаях возможна эксплуатация в филиалах ПО общего назначения, предназначенного для первичного ввода информации и последующей автоматизированной обработки данных в СПО, установленном в головном офисе банка. Такая структура дает возможность органически включить в АБС нового поколения компонент для создания хранилища данных, разделяя системы оперативного действия и системы поддержки принятия решения.

Кроме того, одно из достоинств **принципа многокомпонентности**, являющегося базовым при создании АИС нового поколения, состоит в возможности их поэтапного внедрения. На первом этапе внедрения устанавливаются (или заменяются уже устаревшие) компоненты системы на те рабочие места, которые нуждаются в обновлении ПО. На втором этапе происходит развитие системы с подсоединением новых компонентов и отработкой межкомпонентных связей. Возможность применения такой методики внедрения обеспечивает ее достаточно простое тиражирование и адаптацию к местным условиям. ***Таким образом, автоматизированная информационная система нового поколения - это многокомпонентная система с распределенной базой данных по уровням экспертизы.***

Что же заставляет банки разрабатывать предприятия и банки свои АИС собственными силами [1,2]:

- **Во-первых**, это конечно относительно низкая стоимость таких разработок (по сравнению с покупными). Как правило, к существующим подразделениям департамента информатизации, таким как: управление эксплуатации, управление эксплуатации вычислительной сети и средств связи, экспертно-аналитическое управление (постановка задач), добавляется лишь новая структура: управление развития и разработки АИС, что, как правило, не влечет за собой больших финансовых затрат.
- **Во-вторых**, собственная разработка - это максимальная ориентация на реализацию бизнес - процессов предприятия или банка, его уникальных финансовых и управленческих технологий, складывающихся годами.
- **В третьих**, это позволяет обеспечивать значительно более высокий уровень безопасности и независимости от внешних факторов.
- **В четвертых**, оперативная реакция на изменения правил игры на рынке.

Вместе с тем при собственной разработке необходимо решить целый комплекс организационно-технических задач, которые позволили бы избежать ошибочных решений [1,2]:

- **Во-первых,** правильный выбор архитектуры построения вычислительно-коммуникационной сети и ориентация на профессиональные СУБД. По экспертным оценкам собственные разработки АИС в 53% базируются на СУБД Oracle, около 15% на Informix, 22% - другие СУБД.
- **Во-вторых**, использование при разработке современного инструментария (CASE средства, эффективные средства разработки: Delphi, Designer2000, Developer2000, SQL-Stations и т.п.).
- **В третьих**, мультизадачная инфраструктура разработки проекта, когда конкретный модуль АИС ведет группа разработчиков с взаимосвязанным перечнем задач, построенная на принципах полной взаимозаменяемости,

т.е. функционирование данного модуля АИС и его развитие не связано с одним конкретным разработчиком.

- **В четвертых,** применение эффективных организационно-технических средств по управлению проектом и контролю версий АИС.

Только при соблюдении этих основных положений можно рассчитывать, что собственная разработка окажется конкурентной и эффективной. В противном же случае можно столкнуться с эффектом "неоправданных ожиданий" - это в лучшем случае, а в крайнем случае вообще задуматься о смене АИС. При этом, смена АИС может вызвать как непосредственно смену клиентских модулей и табличной структуры БД, так и потребовать замены серверного и клиентского аппаратного и общесистемного программного обеспечения, включая СУБД, а это дело не дешевое. Поэтому очень важно при выборе варианта реализации АИС сразу решить вопрос о возможностях экспорта/импорта данных в создаваемой системе. При правильном решении данного вопроса смена АИС, если в ней все-таки возникнет необходимость, произойдем практически безболезненно для функциональных подразделений.

В отличие от банковских структур крупные отечественные промышленные предприятия сейчас только подходят к осознанию явной необходимости внедрения и развития корпоративных информационных систем как одной из основных компонент стратегического развития бизнеса. В связи с этим в недалеком будущем можно ожидать расширение рынка корпоративных информационных систем и в последующем его значительно роста. Учитывая тесную интеграцию финансовых и промышленных структур можно полагать, что основой построения корпоративных систем финансово-промышленных групп будут являться, используемые в их финансовых учреждениях, АБС.

1.2. Ориентация на профессиональные СУБД - "За" и "Против"

По материалам периодической печати [1,2] можно судить, что 1998 год стал годом перехода к внедрению АБС четвертого поколения, основой которых, в свою очередь, является ориентация на профессиональные СУБД. Что же это дает и зачем все это нужно:

1. Оптимизированный многопользовательский режим работы с развитой системой транзакционной обработки, что обеспечивает многочисленным пользователям возможность работы с базой данных, не мешая друг другу.
2. Надежные средства защиты информации (учитывая стандартную трехзвенную архитектуру защиты на уровне сети - на уровне сервера БД - на уровне клиентской ОС).
3. Эффективные инструменты для разграничения доступа к БД.
4. Поддержка широкого диапазона аппаратно - программных платформ.
5. Реализация распределенной обработки данных.
6. Возможность построения гетерогенных и распределенных сетей.
7. Развитые средства управления, контроля, мониторинга и администрирования сервера БД.
8. Поддержка таких эффективных инструментариев, как: словари данных, триггеры, функции, процедуры, пакеты и т.п.

Все выше перечисленное обусловило широкое распространение решений на базе профессиональных СУБД в крупных коммерческих банках и промышленных корпорациях. По экспертным оценкам по числу установок лидируют СУБД Oracle, Informix, Sybase. Несмотря на это в большинстве средних и малых банках и предприятиях по-прежнему, ориентируются на решения на базе АИС

третьего и даже второго поколения. Какие же основные "мнимые" стереотипы пока не позволяют этим структурам ориентироваться на использования профессиональных СУБД при построении своих АИС [1,2]:

- **"ПРОТИВ"** - Относительно высокая дороговизна профессиональных СУБД
- **"ЗА"** - Как правило, поставщиками практически всех профессиональных СУБД сейчас предлагаются масштабируемые решения, т.е. например, Enterprise Database - для крупных систем и WorkGroup Database - для средних и малых систем, причем цена последних сравнима с ценами на локальные СУБД.
- **"ПРОТИВ"** - Профессиональные СУБД предъявляют высокие требования к аппаратной платформе.
- **"ЗА"** - С резким ростом производительности Intel-ориентированных аппаратных платформ большинство производителей профессиональных СУБД выпустила свои версии и под Intel-сервера, в том числе и под ОС LINUX, а учитывая что LINUX при всей своей мощности UNIX системы практичсеки беспланая ОС, то и решение на ее основе как правило не повлечет больших финансовых затрат. Это позволяет при построении системы ориентироваться не только на высокопроизводительные многокластерные RISC сервера, но и использовать серверные Intel-платформы.
- **"ПРОТИВ"** - Профессиональные АИС сложны и дороги в администрировании.
- **"ЗА"** - Как правило, сложность администрирования зависит от конкретной АИС. Кроме этого, при эксплуатации АИС в многопрофильном банке или предприятии на UNIX платформе снимает многие проблемы, возникающие на местах, за счет широких возможностей удаленного администрирования из центра.
- **"ПРОТИВ"** - Разработки АИС на промышленной платформе слишком дороги.
- **"ЗА"** - Проектирование современных интегрированных систем - процесс трудоемкий, требующей высокой квалификации разработчиков. Все это находит отражение в цене и объективно делает АИС нового поколения более дорогими, но все же сравнимыми по стоимости с их предшественниками.
- **"ПРОТИВ"** - Внедрение систем на профессиональной платформе процесс затяжной и дорогостоящий.
- **"ЗА"** - Затяжка внедрения, как правило, обусловлена либо недостатком опыта фирмы поставщика по установке таких систем, либо недостаточной готовностью самого внедряемого продукта. Ориентировочный срок установки типовой АИС четвертого поколения под СУБД Oracle при отлаженном технологическом процессе составляет несколько недель.
- **"ПРОТИВ"** - Сопровождение систем на базе профессиональной платформы неоправданно дорого, а качественные характеристики такой АИС оставляют желать лучшего.
- **"ЗА"** - Во многом это предубеждение сложилось на основании опыта эксплуатации АИС зарубежного производства. Можно указать целый ряд случаев, когда зарубежные фирмы поставщики либо отказывались своевременно вносить изменения, обусловленные новыми инструкциями ЦБ, либо требовали за эти изменения неоправданно крупные суммы. Однако это совсем не относится к отечественным системам нового поколения, изначально рассчитанным на изменчивое российское законодательство.

Выводы: Анализ рынка показывает, что на сегодня современная АИС должна представлять собой интегрированный комплекс аппаратно-программных средств реализующих мультипредметную информационную систему, обеспечивающую современные финансовые, управленческие, проектирующие, производственные и сбытовые технологии в режиме реального времени при транзакционной

обработке данных. Если задуматься, то это достаточно закономерно. Персональные СУБД (Clipper, Clarion, FoxPro) совершенно не приспособлены для создания интегрированных систем, работающих с общей базой. В принципе эти СУБД вообще не поддерживают понятие "база данных", работая на уровне индивидуальных таблиц-файлов.

Широко распространенные сегодня системы на базе Btrieve все же трудно назвать масштабируемыми, а саму Btrieve - профессиональной СУБД, пригодной для построения корпоративной информационной системы. Btrieve-системы унаследовали свою архитектуру и большую часть кода от своих предшественников, разработанных на Clipper и Clarion, что во многом объясняет столь большую популярность Btrieve среди фирм, разрабатывавших ранее под эти платформы. Действительно, механически перенести Clarion систему под использование менеджера записей Btrieve относительно несложно, а вот для использования в качестве СУБД Oracle придется существенно изменить архитектуру системы.

В чем основные отличительные особенности корпоративных СУБД. Во-первых, они были изначально направлены на создание интегрированных, многопользовательских систем, имея в своем распоряжении развитые словари данных, что значительно повышает роль системного анализа и моделирования при проектировании системы. Во-вторых, средства разработки для данных СУБД оптимизированы для коллективной разработки сложных систем в рамках единой продуманной стратегической линии. Все это обуславливает неуклонно растущее количество успешных внедрений систем на базе профессиональных СУБД.

1.3. Этапы разработки автоматизированных информационных систем.

Итак, мы выбрали метод, которым будем руководствоваться при проектировании автоматизированной информационной системы. Теперь нам необходимо спланировать комплекс работ по созданию нашей системы в соответствии с типовыми этапами разработки АИС, краткая характеристика которых приведена в табл.1., а последовательность трансформации бизнес модели в объекты базы данных на рис.1.

Таблица 1.Этапы проектирования АИС и их характеристики.

№	Наименование этапа	Основные характеристики
1	Разработка и анализ бизнес - модели	Определяются основные задачи АИС, проводится декомпозиция задач по модулям и определяются функции с помощью которых решаются эти задачи. Описание функций осуществляется на языке производственных (описание процессов предметной области), функциональных (описание форм обрабатываемых документов) и технических требований (аппаратное, программное, лингвистическое обеспечение АИС). **Метод решения**: Функциональное моделирование. **Результат**: 1.Концептуальная модель АИС, состоящая из описания предметной области, ресурсов и потоков данных, перечень требований и ограничений к технической

		реализации АИС. 2.Аппаратно-технический состав создаваемой АИС.
2	Формализация бизнес - модели, разработка логической модели бизнес -процессов.	Разработанная концептуальная модель формализуется, т.е. воплощается в виде логической модели АИС. **Метод решения**: Разработка диаграммы "сущность-связь" (ER (Entity-Reationship) - CASE- диаграммы). **Результат:** Разработанное информационное обеспечение АИС: схемы и структуры данных для всех уровней модульности АИС, документация по логической структуре АИС, сгенерированные скрипты для создания объектов БД.
3	Выбор лингвистического обеспечения, разработка программного обеспечения АИС.	Разработка АИС: выбирается лингвистическое обеспечение (среда разработки - инструментарий), проводится разработка программного и методического обеспечения. Разработанная на втором этапе логическая схема воплощается в реальные объекты, при этом логические схемы реализуются в виде объектов базы данных, а функциональные схемы - в пользовательские формы и приложения. **Метод решения:** Разработка программного кода с использованием выбранного инструментария. **Результат**: Работоспособная АИС.
4	Тестирование и отладка АИС	На данном этапе осуществляется корректировка информационного, аппаратного, программного обеспечения, проводится разработка методического обеспечения (документации разработчика, пользователя) и т.п. **Результат**: Оптимальный состав и эффективное функционирование АИС. Комплект документации: разработчика, администратора, пользователя.
5	Эксплуатация и контроль версий	Особенность АИС созданных по архитектуре клиент сервер является их многоуровневость и многомодульность, поэтому при их эксплуатации и развитии на первое место выходят вопросы контроля версий, т.е. добавление новых и развитие старых модулей с выводом из эксплуатации старых. Например, если ежедневный контроль версий не ведется, то в как показала практика, БД АИС за год эксплуатации может насчитывать более 1000 таблиц, из которых эффективно использоваться будет лишь 20-30%. **Результат**: Наращиваемость и безизбыточный состав гибкой, масштабируемой АИС

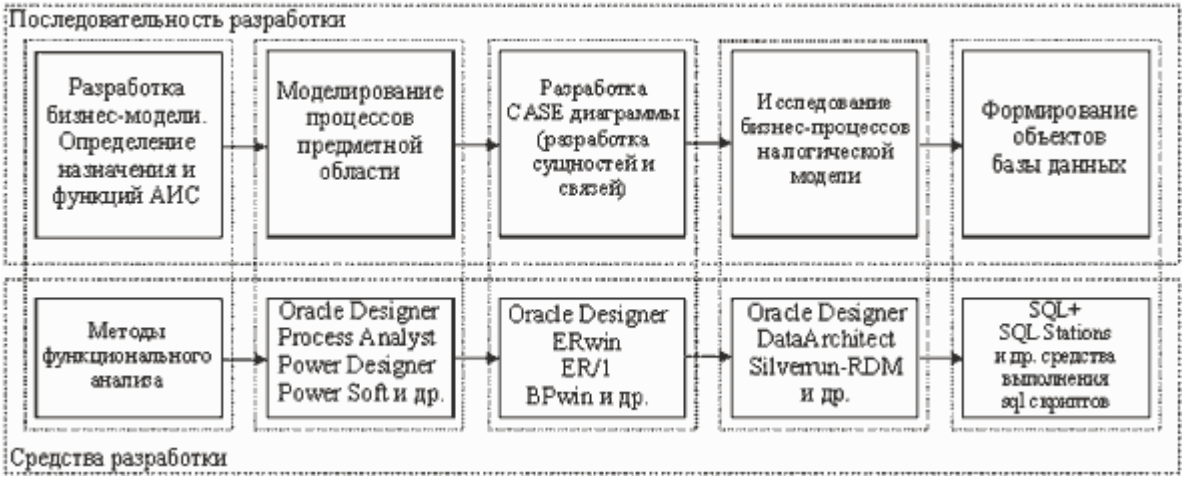


Рис.1. Последовательность трансформации бизнес-модели в объекты БД и приложения.

1.3.1. Разработка и анализ бизнес-модели

При построении эффективной автоматизированной системы первым этапом является исследование и формализация бизнес-процессов деятельности банка или предприятия. Т.е. описание системы ведения делопроизводства с целью эффективного использования информации для достижения поставленных задач и решения проблем, стоящих перед организацией. Организация работы с документами (будь то платежные или конструкторско-технологические документы) является важной составной частью процессов управления и принятия управленческих решений, существенно влияющей на оперативность и качество управления. Процесс принятия управленческого решения состоит из:

- Получения информации;
- Переработка информации;
- Анализа, подготовки и принятия решения.

Все эти этапы самым тесным образом связаны с документационным обеспечением процессов управления, проектирования и производства. Если на предприятии отсутствует четкая организация работы с документами, то, как следствие этого, закономерно появление документов низкого качества, как в оформлении, так и в полноте и ценности содержащейся в них информации, увеличение сроков их обработки. Это приводит к ухудшению качества управления и увеличению сроков принятия решений и числу неверных решений. С ростом масштабов предприятия и численности его сотрудников вопрос об эффективности документационного обеспечения управления становится все более актуальным. Основные проблемы, возникающие при этом, выглядят примерно так:

- руководство теряет целостную картину происходящего;
- структурные подразделения, не имея информации о деятельности друг друга, перестают слаженно осуществлять свою деятельность. Неизбежно падает качество обслуживания клиентов и способность организации поддерживать внешние контакты;
- это приводит к падению производительности и вызывает ощущение недостатка в ресурсах: людских, технических, коммуникационных и т.д.;
- приходится расширять штат, вкладывать деньги в оборудование новых рабочих мест, помещения, коммуникации, обучение новых сотрудников;

- для производственных предприятий увеличение штата может повлечь изменение технологии производства, что потребует дополнительных инвестиций;
- оказывается, что штат увеличен, производительность упала, производство требует инвестиций, соответственно возникает потребность в увеличении оборотного капитала, что может потребовать новых кредитов и уменьшить плановую прибыль.

В итоге предприятие перестает расти интенсивно и дальнейшее расширение происходит чисто экстенсивным путем за счет ранее созданной прибыли.

Почему же сегодня, когда для организации документооборота (в дальнейшем под этим термином мы будем понимать документооборот любых документов: конструкторских, технологических, финансовых, организационных и т.п.) предлагается множество самых различных средств автоматизации, документооборот часто организован плохо, даже на относительно небольших предприятиях? Ответ, независимо от степени автоматизации предприятия и его типа, может быть один - ***отсутствие или игнорирование модели организации документооборота неизбежно приведет к тому, что старые проблемы останутся нерешенными. При этом, если по состоянию делопроизводства в организации был "ручной" хаос, то результатом автоматизации будет "компьютерный" хаос***.

Когда на Западе, а теперь и в России схлынула первая волна увлечения системами автоматизации документооборота, оказалось, ***что без должной оценки возможностей пользователя, исследования бизнес - процессов его предприятия трудно ожидать эффекта от внедрения как систем документооборота класса docflow так и, тем более, workflow.*** При этом совсем неважно, как планируется или уже реализован документооборот: вручную или путем автоматизации с помощью мощных западных либо отечественных пакетов - всегда на первом месте должна быть четкая стратегия, направленная на упорядочение бизнес - процессов. Иначе говоря, прежде чем что-то делать, неплохо было бы ответить на вопрос, кому и почему выгодно выполнять те или иные процессы, имеющие место на предприятии. ***Проводя в жизнь программу модернизации делопроизводства, важно представлять, какого уровня уже достигло предприятие и какое место ему отводится в модельном пространстве системы документооборота.***

1.3.1.1. Основные понятия электронного документооборота

Документ.

Пример: Вы написали заявление на отпуск и передали его в отдел кадров. Так появился документ. Что же превратило чистый лист бумаги в документ? Во-первых, информация, представленная в виде текста. Во-вторых, текст в форме заявления. И в-третьих, бумагу готовили с расчетом на последующую деятельность сотрудников отдела кадров.

Теперь предположим, что вы обратились в отдел кадров с устным заявлением об отпуске. Можно ли назвать документом эту процедуру? Устная беседа не была зафиксирована физически, она не поддается точному воспроизведению и, следовательно, документом не является.

Итак, документ - это совокупность трех составляющих [3]:

- Физическая регистрация информации.

- Форма представления информации
- Активизация определенной деятельности.

Именно некоторая деятельность и превращает информацию в документ. Но документ перестает существовать, если в дальнейшем не подразумевает процедуры обработки. При этом форма документа тесно связана с характером дальнейшей деятельности, она порождает необходимость документов. Так родилась бюрократия- неизбежный спутник цивилизации.

Документ [4] - слабоструктурированная совокупность блоков или объектов информации, понятная человеку. В общем случае обойтись без документов пока нельзя. Сам по себе документ, независимо от того, обычная ли это бумага или электронный бланк, проблем корпорации не решает - первичны бизнес-процессы и четкий контроль за выполнением проекта.

Бюрократическая технология - это технология взаимодействия людей, служб и подразделений внутри и вне организации. Не будет технологии - возникнет анархия. Если работник не знает что ему надо делать, он делает то, что считает нужным, а не то, что требует тот или иной бизнес-процесс предприятия. Сама бюрократия неизбежна, опасность представляет отрыв реальных целей предприятия от работы текущей системы документооборота.

Собственно документооборот может быть двух типов:

- универсальный - автоматизирующий существующие информационные потоки слабоструктурированной информации. Справедливо было бы его называть аморфным или беспорядочным документооборотом;
- операционный - ориентированный на работу с документами, содержащими операционную атрибутику, вместе с которой ведется слабоструктурированная информация.

Кроме собственно документов важен еще регламент работы с ними. Любой опытный менеджер может подтвердить, что работа не по регламенту порой отнимает намного больше времени, чем собственно производственная деятельность. Дублирование документов, их потеря, навязчивый способ их распространения, а также запутанный порядок их прохождения могут существенно усложнить работу, повысив вероятность допущения ошибки вследствие, например, потери нужной информации.

Итак, документ занимает определенное место в процессе некоторой деятельности на границе разделяемых функций исполнения. Поэтому правильно рассматривать документ как инструмент распределения функций между работниками [3].

1.3.1.2. Преимущества электронного документооборота

К основным преимуществам электронного документооборота можно отнести следующие:

- Полный контроль за перемещением и эволюцией документа, регламентация доступа и способ работы пользователей с различными документами и их отдельными частями.
- Уменьшение расходов на управление за счет высвобождения (на 90% и более) людских ресурсов, занятых различными видами обработки бумажных документов, снижение бюрократической волокиты за счет

маршрутизированного перемещения документов и жесткого контроля за порядком и сроками прохождения документов.
- Быстрое создание новых документов из уже существующих.
- Поддержка одновременной работы многих пользователей с одним и тем же документом, предотвращение его потери или порчи.
- Сокращение времени поиска нужных документов.

Использование АИС может рассматриваться в качестве базы для общего совершенствования управления предприятием. При этом управление предприятием реализует следующие основные функции:

- обслуживание клиентов;
- разработка продукции;
- учет и контроль за деятельностью предприятия;
- финансовое обеспечение деятельности предприятия и т.д.

1.3.1.3. Модели информационного пространства предприятия.

Комплексная автоматизация этих функции требует создания единого информационного пространства предприятия, в котором сотрудники и руководство могут осуществлять свою деятельность, руководствуясь едиными правилами представления и обработки информации в документном и бездокументном виде.

Для этого в рамках предприятия требуется создать единую информационную систему по управлению информацией или единую систему управления документами, включающую возможности:

- удаленной работы, когда члены одного коллектива могут работать в разных комнатах здания или в разных зданиях;
- доступа к информации, когда разные пользователи должны иметь доступ к одним и тем же данным без потерь в производительности и независимо от своего местоположения в сети;
- средств коммуникации, например: электронная почта, факс, печать документов;
- сохранения целостности данных в общей базе данных;
- полнотекстового и реквизитного поиска информации;
- открытость системы, когда пользователи должны иметь доступ к привычным средствам создания документов и к уже существующим документам, созданным в других системах;
- защищенность информации;
- удобства настройки на конкретные задачи пользователей;
- масштабируемости системы для поддержки роста организаций и защиты вложенных инвестиций и т.д.

Начальным этапом создания такой системы является построение модели предметной области или другими словами модели документооборота для конкретного бизнеса и позиционировать в ней свое предприятие.

Определенные ранее направления автоматизации документооборота: поддержка фактографической информации, возможность работы с полнотекстовыми документами, поддержка регламента прохождения документов, определяют трехмерное пространство свойств, где по некоторой траектории движется любой программный продукт данного класса, проходя различные стадии в своем развитии (рис.2.).

Первая ось (F) характеризует уровень организации хранения фактографической информации, которая привязана к специфике конкретного рода деятельности компании или организации. Например: при закупке материальных ценностей происходит оформление товарно-сопроводительных документов (накладных, приемо-передаточных актов, приходных складских ордеров и т.д.), регистрируемых в качестве операционных документов, атрибутика которых очень важна для принятия управленческих решений. Информация из операционных документов используется при сложной аналитической и синтетической обработке, и, в частном случае, может быть получена пользователем через систему отчетов.

Рис.2. Трехмерное пространство свойств.

Вторая ось (D) - полнотекстовые документы, отражает необходимость организации взаимодействия: формирование и передача товаров, услуг или информации как внутри корпорации, так и вне ее. В этих документах наряду с фактографической информацией содержится слабо структурированная информация, не подлежащая автоматизированной аналитической обработке, такая, например, как форс-мажерные факторы и порядок предъявления претензий при нарушении условий договора. Все взаимоотношения между субъектами бизнеса сопровождаются документами, которые становятся осязаемым отражением результата взаимодействия.

Третья ось (R) вносит в пространство документооборота третье измерение - регламент процессов прохождения документов, а именно: описание того какие процедуры, когда и как должны выполняться. Основа для позиционирования относительно данной оси - набор формальных признаков (атрибутов) и перечень выполнения операций.

Точка в пространстве (F, D, R) определяет состояние системы документооборота и имеет координаты (f,d,r), где f,d и r принадлежат множествам F,D и R, соответственно. Положение этой точки зависит от уровня развития и стадии внедрения системы документооборота на предприятии, а также от его специфики и самих масштабов бизнеса.

Представив модель документооборота именно таким образом, можно, например, зная текущее положение дел с организацией делопроизводства на каждом конкретном предприятии, четко представить, в каких направлениях нужно двигаться дальше, чего недостает в текущий момент и каким образом органично использовать уже существующие системы автоматизации. Например, в одном из московских банков был накоплен большой массив фактографических данных, для обработки которых использовалась современная СУБД, развернутая на мощных, отказоустойчивых серверах - все, казалось бы, должно быть

отлично. Однако при работе с внутренними документами наблюдалось дублирование информации: возникали ситуации, когда "никто вроде бы и не виноват", а банк время от времени лишается выгодных клиентов. Причина в том, что точка, отражающая положение системы документооборота для этой организации, имела достаточно большие координаты по оси "F" и, возможно, по оси "D", однако значение координаты по оси "R" было близко к нулю. Конкретным решением в этом случае может быть рассмотрение вопроса о внедрении системы управления регламентом. При этом не надо пока заботится о СУБД (ось "F") или электронных архивах (ось "D") - речь идет только об изменении значения координат по оси "R".

В общем случае, как уже отмечалось, процесс автоматизации делопроизводства на предприятии можно представить в виде кривой в трехмерном пространстве координат F,D,R. Причем, чем круче эта кривая, тем быстрее идет процесс модернизации, а чем больше значения всех трех координат - тем выше уровень автоматизации на корпорации и, как следствие, тем меньше у нее проблем с организацией своей собственной деятельности.

Работоспособна ли данная модель для задания пространства развития неавтоматизированной системы управления документооборотом? Да. Единственно, что в этом случае решается задача не облегчения рутинного труда по перемещению документов, их поиску и регистрации, а упорядочения всей системы документооборота. Новое качество, с которым сегодня ассоциируется возросший интерес к системам электронного документооборота, связано с использованием инструментальных систем, предназначенных для хранения, регистрации, поиска документов, а также для управления регламентом. Чаще ошибочно под новым качеством сегодня понимается простое внедрение отличной от ранее используемой технологии работы с документами, например локальной сети вместо дискет, переносимых с одного компьютера на другой. Вряд ли в этом случае уместно говорить о новом качестве управления предприятием. Кстати, уже упомянутый пример ручной работы режимных служб "почтовых ящиков", прекрасно вписывается в предложенную модель документооборота, а точка, отражающая его состояние, будет иметь координаты (1,1,1) - все равномерно, единственное, что отсутствует - компьютеризация.

Эволюция модели.

Рассмотренная модель документооборота не является застывшим образованием, данным нам в ощущениях - прежде чем сформировалось современное представление о контурах этой модели, она претерпела три основные фазы своей эволюции, две из которых представлены на рис.3., а третья на рис.2.

Фаза первая - фактографическая. Начало любой деятельности знаменуется обычно периодом накопления первичной информации, имеющей жесткую структуру и атрибутику. Условно эту фазу можно представить в виде одной единственной оси.

Рис.3. Эволюция бизнес - моделей документооборота.

Точка на этой оси - это текущее состояние системы документооборота организации. Движение по оси вверх характеризует накопление фактографической информации и начиная с определенного момента которого можно отметить второй этап первой фазы - возникновение понятия "операция". Документ теперь представляется как некоторый привязанный к бизнес - процессам предприятия агрегат из имеющихся характеристик (атрибутов). На этом этапе начинается процесс возникновения неравенства между ранее равноправными документами, в частности, документ-основание, а дальнейшее движение по оси приобретает все более операционный оттенок. После возникновения привязки к конкретным бизнес - процессам дальнейшая эволюция документооборота в одномерном пространстве уже невозможна - необходим новый качественный скачок к новой фазе.

Фаза вторая - полнотекстовая. Расширение организации и увеличение круга решаемых задач требуют использования полнотекстовых документов, включающих уже не только тексты, но и любые другие способы представления: графики, таблицы, видео и т.п виды конструкторско-технологической документации. Возникает новая ось - полнотекстовые или, лучше, мультимедийные документы, а точка в новом, уже двумерном, пространстве характеризует систему документооборота предприятия, где кроме фактографической базы документов имеются уже хранилища и архивы информации.

Хранилища позволяют накапливать документы в различных форматах, предполагают наличие их структуризации и возможностей поиска. Если на предприятии уже используется автоматизация, то хранилище - это не что иное, как электронный архив. Движение по оси "полнотекстовые документы" предполагает наращивание атрибутивных возможностей: разграничение доступа, расширение средств поиска, иерархию хранения, классификация. Здесь же возникают такие понятия как электронная подпись, шифрование и т.п.

На данной оси также имеются свои этапы - с определенного момента развития хранилища можно уже говорить не об индивидуальном, а о корпоративном архиве, обслуживающем деятельность рабочих групп. Точка на плоскости эволюции, достигнутой во второй фазе, характеризует систему документооборота, позволяющую отображать фактографическую информацию в виде полнотекстовых документов, имеющих необходимое количество атрибутов. Доступ к этим документам может быть осуществлен по маршруту любого уровня сложности с соблюдением различных уровней конфиденциальности. Если, например, говорить о точке "А", то соответствующее ей состояние системы

документооборота позволяет осуществлять синхронизацию работы различных рабочих групп сотрудников корпорации, расположенных на различных площадках. Система для этой точки предполагает также структурирование информации по уровням управления и наличие средств репликации данных. Однако, как только речь пошла о корпорации, двумерного пространства для соответствующей ей системы документооборота опять становится недостаточно - необходим новый скачок к очередной фазе.

Фаза третья - регламентирующая. Нормальный документооборот в масштабах корпорации невозможен без решения вопросов согласования или соблюдения регламента работы. Если ранее, на второй фазе (плоскость) негласно присутствовал лишь один, простейший регламент (нулевая точка) - каждый сотрудник имел доступ к архиву или его части, либо в папку каждому работнику помещалось индивидуальное задание (иначе говоря, было известно только, что документ существует), то сейчас этого недостаточно. Требуется уже интегральная оценка. Необходим, например, контроль за тем, как работник выполнил задание, или как продвигается документ в условиях нелинейного процесса своего согласования (например согласования пакета конструкторско-технологической документации на сборочную единицу).

Третья ось в пространстве документооборота предприятия, как и две другие имеет свое деление на этапы. Первоначальный этап движения по оси характеризуется наличием упрощенного регламента, отображаемого появлением атрибутов, отвечающих за регламент, например: "оплатить до", "действителен для". Количественное накопление атрибутов и расширение возможностей по управлению регламента сопровождается постепенным переходом ко второму этапу, отличительная черта которого - появление системы, специально предназначенной для отслеживания процесса соблюдения регламента. При дальнейшем движении вдоль этой оси можно говорить о появлении единой системы управления проектом. Теперь документ в системе "документооборота" становится вторичным - первична цель бизнеса, сам процесс реализации бизнес - процедур, оставляющий после себя документы.

Оси "F" и "D" определяют специфику деятельности организации, регламентируемую положением третьей координаты (R) пространства модели документооборота. При этом модель не зависит от технологии обработки документов, принятой на предприятии - все решает только цель деятельности, будь то государственная организация, торговая компания и промышленная фирма. В общем случае можно выделить три типа организаций:

- Банк и торговая компания: приобретение, наценка, продажа, получение прибыли - главный объект деятельности;
- бюджетная организация: основная деятельность - формирование документов;
- промышленное предприятие: закупка сырья, переработка, создание нового продукта, реализация, получение прибыли. Цель деятельности - операция.

Если задачей организации является формирование документов, например мэрия, суд или министерство, то ее позиция в модели будет занимать достаточно высокое положение относительно осей "F" и "D". Кстати, сегодня наибольшей популярностью пользуются именно приложения, ориентированные на автоматизацию деятельности государственных и правительственных административных структур - основная цель которых и состоит в подготовке документов. Однако если рассматривать деятельность коммерческого банка или

фирмы задача которой - производство операций, материальных ценностей , то здесь уже все три координаты должны иметь сбалансированные значения.

Рассмотрим в качестве примера основные шаги при разработки функциональной модели типовой АБС. Среди архитектур реализации АБС на сегодня выделят АБС пяти поколений: первые были предназначены для решения задач автоматизации бухгалтерских операций, АБС второго и третьего поколений также реализовывали элементы автоматизированного документооброта, АБС четвертого поколения строились по классической схеме трехзвенной клиент-серверной архитектуры и имели модульную структуру, реализующую концепции docflow. АБС пятого поколения являются информационными системы реилизующими полнофункциональную модель workflow - автоматизации бизнесс-процессов (рис.4,5).

Рис.4. Поколения АБС и их характеристики.

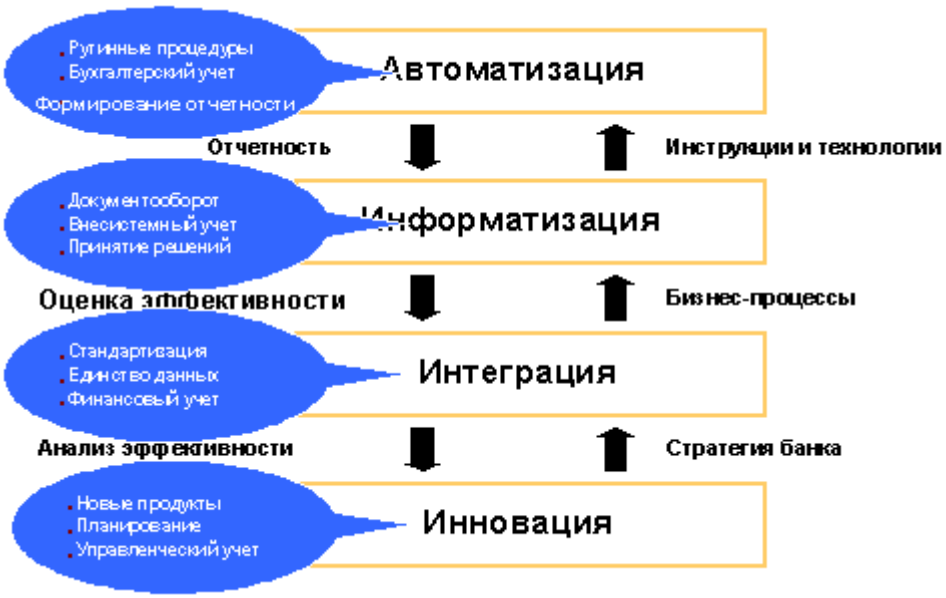


Рис.5.приницпы реализации интеллектуальной инфраструктуры банка.

Схема функциональных бизнес-потоков типовой АБС четвертого поколения представлена на рис.6., а функциональная схема технологических потоков операций на рис.7.

Рис.6. Функциональная схема типовой АБС 4-го поколения.

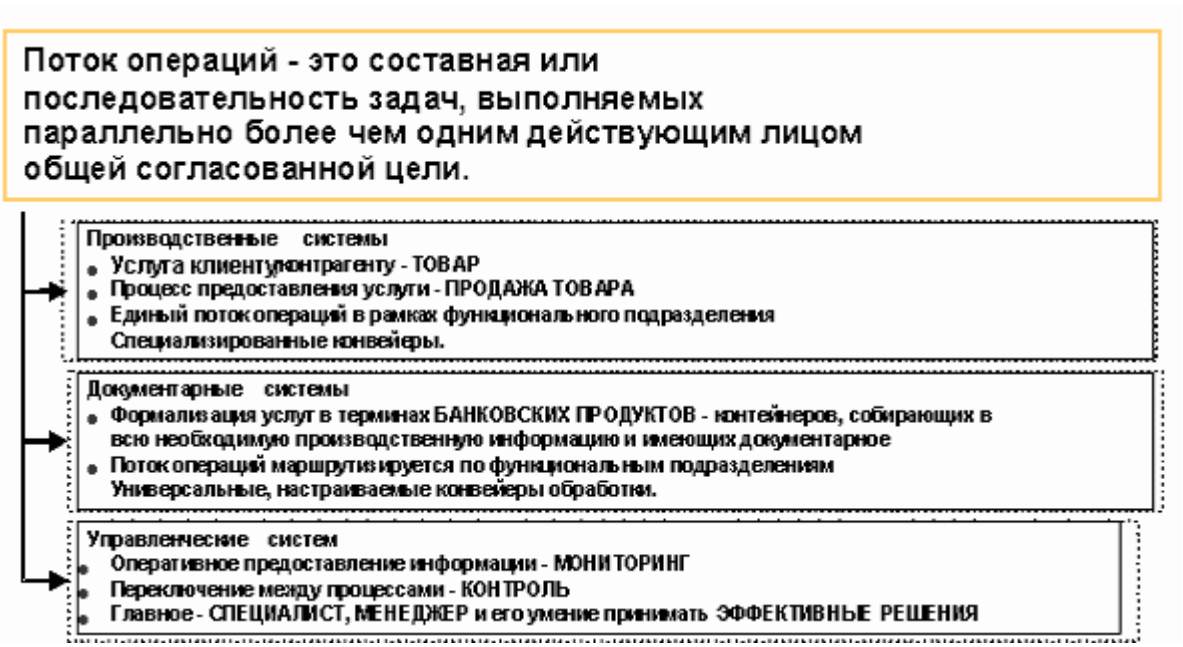


Рис.7. Функциональная схема технологических потоков операций.

Назначения большинства модулей АБС (рис.6) достаточны понятны, едисктвенное следует уточнить, что под интернет-банкингом понимает автоматизированная система реализующая технологии busines-to-customers

(клиентское обслуживание, включая электронные платежы физических лиц) и bisnes-to-bisnes (обеспечение работы дистрибьюторско-дилерской сети корпораций в режиме on-line коммерции).

Разработав функциональную схему бизнес-потоков, структурировав ее в соотвествии с модульным принципом построения любой системы управления и выделив базовые технологические потоки операций на каждом из уровне модульности (итог: перечень документов и операций обрабатываемых на конкретном рабочем месте, маршруты их движения, контроля и управления), можно приступать к следующему этапу проектирования информационной системы.

1.3.1.4. Выводы.

Координаты точки, характеризующей сбалансированную систему документооборота (бизнес - модель функционирования предприятия), должны иметь ненулевые значения, а в идеале быть примерно одинаковы - соответствовать друг другу. Главное не автоматизация как таковая, а оптимизация потоков документов и интегральность - даже самая прекрасная программа автоматизации документооборота окажется напрасным вложением средств, если модель одномерная или плоская.

Нет большого смысла говорить о жизненном цикле документов без связи с основными бизнес-процессами предприятия. Система автоматизации документооборота, функционирующая в отрыве от всех слоев, будет мертва, мало того, она может нанести вред: запутать и без того неуправляемые бизнес - процессы, отвлечь персонал от выполнения основной работы ради поддержания системы автоматизации документооборота, по сути дела, ничего не автоматизирующего. И, как следствие, раздувание штатного расписания и дискредитация самой идеи автоматизации делопроизводства. Знакомая ситуация - все работники заняты, работа кипит, однако если рассмотреть две разные фирмы имеющие равный доход, занимающиеся одним бизнесом, то штат сотрудников у одной из них будет вдвое больше, чем у другой. При создании или внедрении АИС необходимо всегда помнить, что компьютер или комплект программных средств типа workflow - это только голый инструмент, неумелое использование которого чаще всего влечет за собой только вред, а не долгожданное облегчение и освобождение от внутрикорпоративных проблем по управлению.

Технологии создания распределенных информационных систем

Построение современных распределенных информационных систем сегодня на прямую связано с реаляционными и объектно-ориентированными СУБД, которые в последнее время утвердились как основные средства для обработки данных в информационных системах различного масштаба - от больших приложений обработки транзакций в банковских системах до персональных систем на PC. В настоящее время существует множество систем управления базами данных (СУБД) и других программ выполняющих сходные функции. Инструментальные средства Oracle - одни из лучших и наиболее мощных имеющихся инструментов разработки профессионального класса.

2.1 Базы данных и их сравнительные характеристики.

2.1.1 Классификация моделей построения баз данных

В зависимости от архитектуры СУБД делятся на *локальные и распределенные* СУБД. Все части локальной СУБД размещаются на одном компьютере, а распределенной на нескольких. За несколько десятилетий последовательно появлялись системы (СУБД), основанные на трех базовых моделях данных: иерархической, сетевой и реляционной. Основные определения теории баз знаний и баз данных представлены в таблице 1.

Табл.1. Основные определения

№	Термин	Определение
1	База данных (БД)	Базами данных называют электронные хранилища информации, доступ к которым осуществляется с помощью одного или нескольких компьютеров.
2	Системы управления базами данных (СУБД)	это программные средства для создания, наполнения, обновления и удаления баз данных.
3	База знаний	Базы знаний это хранилища знаний, представленных в формализованном виде.
4	Система управления базами знаний СУБЗ	это программные средства для создания, наполнения, обновления и удаления баз знаний

	Дополнения к таблице 1.
Виды знаний:	
Процедурные	Знания, отвечающие на вопрос "Как решать поставленную задачу?"
Декларативные	Знания, не содержащие в явном виде процедуры решения задач.
Каузальные	Знания о причинно-следственных связях между объектами предметной области
Неточные	Знания отличающиеся неполнотой или противоречивостью.

Парадигмы решения задач *В СУБД* *В СУБЗ*	 Данные + Алгоритм = Программа решения задачи Знания + Стратегия вывода = Решение проблемы.
Модели знаний *Продукционная* *Фреймовая* *Семантическая сеть*	 Знания представленные в формате "ЕСЛИ-ТО" Знания представленные в виде набора взаимосвязанный фреймов. Граф, вершины которого соответствуют объектам или понятиям, а дуги определяют отношения между вершинами.
Фрейм *Фрейм прототип* *Конкретный фрейм*	Структурированное описание объекта предметной области состоящее из наименования объекта (имя фрейма), атрибутов объекта (свойств, характеристик) - слоты фрейма. Это фрейм у которого значения слотов не определены. Это фрейм прототип с конкретными значениями.
Enterprise JavaBeans.	Стандарт для создания средствами языка Java пригодных для многократного использования компонентов, из которых формируются прикладные программы. Компоненты Enterprise JavaBeans облегчают разработку программ, обеспечивающих доступ к хранимой в базе данных информации.
Распараллеливание обработки запроса ***(Intraquery parallelism).***	Использование нескольких ЦП для обработки одного запроса.
Параллельная обработка запросов (interquery parallelism)	подразумевает параллельную обработку нескольких запросов (на разных ЦП).
Уровень изоляции (Isolation level).	Установочный параметр БД, определяющий, в какой степени одновременно обратившиеся к базе данных пользователи могут оказывать влияние на работу друг друга. Как правило, используются три уровня изоляции: *завершение чтения* (read committed), характеризуется большим количеством одновременно обслуживаемых пользователей и низким уровнем изоляции каждого из них); *в установленном порядке* (serializable), небольшое число одновременно обслуживаемых пользователей, высокая степень изоляции *и повторяющееся чтение* (repeatable read), сочетание двух первых уровней.
Технология COM	COM - Component Object Model - Компонентная модель объектов, предложена корпорацией Микрософт.
*Технология **CORBA***	CORBA - Common Object Require Broker Architecture - архитектура с брокером требуемых общих объектов,

	разработана независимой группой OMG.
JDBC (Java Database Connectivity).	Интерфейс взаимодействия с базами данных на языке Java. Этот стандарт, разработанный фирмой Sun Microsystems, определяет способы доступа Java-приложений к данным БД.
ODBC (Open Database Connectivity).	Открытый интерфейс взаимодействия с базами данных. Предложенный корпорацией Microsoft стандарт, регулирующий доступ Windows -приложений к базам данных. Стандарт ODBC постепенно заменяется спецификацией OLE DB.
OLAP (Online analytical processing).	Оперативный анализ данных. Этот метод обработки применяется с целью ускорения обработки запросов и предусматривает предварительный расчет часто запрашиваемых данных (например, сумм или значений счетчика).
OLE DB (Object Linking and Embedding Database).	OLE для баз данных. Новый стандарт Microsoft, регулирующий доступ приложений к базам данных. Имеет расширения для серверов OLAP и предусматривает применение специальных средств обработки мультимедийных данных.
	Дополнения к табл.2
Операция соединения (Join).	Процесс, позволяющий объединять данные из двух таблиц посредством сопоставления содержимого двух аналогичных столбцов.
SQL (Structured query language).	Язык структурированных запросов, язык S0L. Является принятым в отрасли стандартом для выполнения операций вставки, обновления, удаления и выборки данных из реляционных БД.
Хранимая процедура (Stored procedure).	Программа, которая выполняется внутри базы данных и может предпринимать сложные действия на основе информации, задаваемой пользователем. Поскольку хранимые процедуры выполняются непосредственно на сервере базы данных, обеспечивается более высокое быстродействие, нежели при выполнении тех же операций средствами клиента БД.
Транзакция (Transaction).	Совокупность операций базы данных, выполнение которых не может быть прервано. Для того чтобы изменения, внесенные в БД в ходе выполнения любой из входящих в транзакцию операций, были зафиксированы в базе данных, все операции должны завершиться успешно. Все базы данных, представленные в нашем обзоре, позволяют использовать транзакции, тогда как БД для настольных систем, например Visual dBase фирмы Inprise или Microsoft Access, не предусматривают применения механизма транзакций.
Триггер (Trigger).	Программа базы данных, вызываемая всякий раз при вставке, изменении или удалении строки таблицы. Триггеры обеспечивают проверку любых изменений на корректность, прежде чем эти изменения будут приняты

2.1.1.1 Иерархическая модель

Первые иерархические и сетевые СУБД были созданы в начале 60-х годов. Причиной послужила необходимость управления миллионами записей (связанных друг с другом иерархическим образом), например при информационной поддержке лунного проекта Аполлон. Среди реализуемых на практике СУБД этого типа преобладает система *IMS* (Information Management System компании IBM) (На данный момент это самая распространенная СУБД из всех данного типа). Применяются и другие иерархические системы: *TDMS (*Time-Shared Date Management System) компании Development Corporation; *Mark IV* Multi - Access Retrieval System компании Control Data Corporation; *System - 2000* разработки SAS-Institute.

Отношения в иерархической модели данных организованы в виде совокупностей деревьев, где *дерево* - структура данных, в которой тип сегмента потомка связан только с одним типом сегмента предка. Графически: *Предок* - точка на конце стрелки, а *Потомок* - точка на острие стрелки. В базах данных определено, что точки - это типы записей, а стрелки представляют отношения один - к - одному или один - ко - многим.

К ограничения иерархической модели данных можно отнести:

1. Отсутствует явное разделение логических и физических характеристик модели;
2. Для представления неиерархических отношений данных требуются дополнительные манипуляции;
3. Непредвиденные запросы могут требовать реорганизации базы данных.

2.1.1.2 Сетевая модель.

Сети - естественный способ представления отношений между объектами. Они широко применяются в математике, исследованиях операций, химии, физике, социологии и других областях знаний. Сети обычно могут быть представлены математической структурой, которая называется *направленным графом.* Направленный граф имеет простую структуру. Он состоит из точек или *узлов,* соединенных стрелками или ребрами. В контексте моделей данных узлы можно представлять как типы записей данных, а ребра представляют отношения один-к -одному или один-ко-многим. Структура графа делает возможными простые представления иерархических отношений (таких, как генеалогические данные) .

Сетевая модель данных - это представление данных сетевыми структурами типов записей и связанных отношениями мощности один-к-одному или один-ко-многим. В конце 60-х конференция по языкам систем данных (Conference on Data Systems Languages, CODASYL) поручила подгруппе, названной Database Task Group (DTBG), разработать стандарты систем управления базами данных. На DTBG оказывала сильное влияние архитектура, использованная в одной из самых первых СУБД, Iategrated Data Store (IDS), созданной ранее компанией General Electric.Это привело к тому, что была рекомендована сетевая модель.

Документы Database Task Group (DTBG) (группа для разработки стандартов систем управления базами данных) от 1971 года остается основной формулировкой сетевой модели, на него ссылаются как на модель CODASYL DTBG. Она послужила основой для разработки сетевых систем управления

базами данных нескольких производителей. IDS (Honeywell) и IDMS (Computer Associates) - две наиболее известных коммерческих реализации. В сетевой модели существует две основные структуры данных: типы записей и наборы:

- *Тип записей.* Совокупность логически связанных элементов данных.
- *Набор.* В модели DTBG отношение один-ко-многим между двумя типами записей.
- *Простая сеть.* Структура данных, в которой все бинарные отношения имеют мощность один-ко-многим.
- *Сложная сеть.* Структура данных, в которой одно или несколько бинарных отношений имеют мощность многие-ко-многим.
- *Тип записи связи*. Формальная запись, созданная для того, чтобы преобразовать сложную сеть в эквивалентную ей простую сеть.

В модели DBTG возможны только простые сети, в которых все отношения имеют мощность один-к-одному или один-ко-многим. Сложные сети, включающие одно или несколько отношений многие-ко-многим, не могут быть напрямую реализованы в модели DBTG. Следствием возможности создания искусственных формальных записей является необходимость дополнительного объема памяти и обработки, однако при этом модель данных имеет простую сетевую форму и удовлетворяет требованиям DBTG.

2.1.1.3 Реляционная модель.

В 1970-1971 годах Е.Ф.Кодд опубликовал две статьи, в которых ввел реляционную модель данных и реляционные языки обработки данных - реляционную алгебру и реляционное исчисление.

- *Реляционная алгебра* Процедурный язык обработки реляционных таблиц.
- *Реляционное исчисление* Непроцедурный язык создания запросов.

Все существующие к тому времени подходы к связыванию записей из разных файлов использовали физические указатели или адреса на диске. В своей работе Кодд продемонстрировал, что такие базы данных существенно ограничивают число типов манипуляций данными. Более того, они очень чувствительны к изменениям в физическом окружении. Когда в компьютерной системе устанавливался новый накопитель или изменялись адреса хранения данных, требовалось дополнительное преобразование файлов. Если к формату записи в файле добавлялись новые поля, то физические адреса всех записей файла изменялись. То есть такие базы данных не позволяли манипулировать данными так, как это позволяла бы логическая структура. Все эти проблемы преодолела **реляционная модель, основанная на логических отношениях данных.**

Существует два подхода к проектированию реляционной базы данных.

- *Первый подход* заключается в том, что на этапе концептуального проектирования создается не концептуальная модель данных, а непосредственно реляционная схема базы данных, состоящая из определений реляционных таблиц, подвергающихся нормализации.
- *Второй подход* основан на механическом преобразовании функциональной модели, созданной ранее, в нормализованную реляционную модель. Этот подход чаще всего используется при проектировании больших, сложных схем баз данных, необходимых для корпоративных информационных систем.

Табл.1. Основные определения реляционных СУБД

№	Термин	Определение
1	*Реляционная модель данных*	Организует и представляет данные в виде таблиц или реляций.
2	*Реляционная база данных (РБД, RDBMS).*	База данных, построенная на реляционной модели.
2	*Реляция (таблица-элементарная информационная единица)*	Двумерная таблица, содержащая строки и столбцы данных.
4	*Степень реляции.*	Количество атрибутов реляции. При том необходимо помнить, что никакие два атрибута реляции не могут иметь одинаковых имен.
	Кортежи	Строки реляции (таблицы), соответствуют объекта, конкретному событию или явлению.
	Атрибуты	Столбцы таблицы, характеризующие признаки, параметры объекта, события, явления.
	Область атрибута	Набор всех возможных значений, которые могут принимать атрибуты. Если в процессе работы возникает ситуация, что атрибут неприменим или значения одного или нескольких атрибутовстроки пока неизвестны, то строка запишется в базуданных с пустыми значениямиэтих атрибутов (NULL строка).
	Пустое значение	Значение, приписываемое атрибуту в кортеже, если атрибут неприменим или его значение неизвестно
	Ключ	Любой набор атрибутов, однозначно определяющий каждый кортеж реляционной таблицы.
	Ключ реляции	Ключ также можно описать как минимальное множество атрибутов, однозначно определяющих (или функционально определяющих)каждое значение атрибута в кортеже.
	Составной ключ	Ключ содержащий два или более атрибута.
	Потенциальный ключ	В любой данной реляционной таблице может оказаться более одного набора атрибутов. Обычно в качестве *первичного ключа* выбирают потенциальный ключ, которым проще всего пользоваться при повседневной работе по вводу данных.
	Первичный ключ.	Поле или набор полей, однозначно идентифицирующий запись.
	Внешний ключ.	Набор атрибутов одной таблицы, являющийся ключом другой (или той же самой) таблицы; используется для определения логических связей между таблицами. Атрибуты внешнего ключа не обязательно должны иметь те же имена, что и атрибуты ключа, которым они соответствуют.
	Рекурсивный внешний ключ.	Внешний ключ, ссылающийся на свою собственную реляционную таблицу.
	Родительская реляция (таблица)	Таблица, поля которой входят в другую таблицу.

	Дочерняя реляция (таблица)	Таблица, поля которой используют информацию из полей другой таблицы, являющейся по отношению к данной родительской.
	Отношение один-к-одному	Когда одной записи в родительской таблицы соответствует одна запись в дочерней таблице
	Отношение один-ко-многим	Когда одной записи в родительской таблицы соответствует несколько записей в дочерней таблице
	Отношение многие-ко-многим	Когда многим записям в родительской таблицы соответствуют несколько записей в дочерней таблице
	Рекурсивное отношение.	Отношение, связывающее объектное множество с ним самим.
	***View** (Представления)*	Информационная единица РБД (по структуре аналогичная таблице), записи которой сформированы в результате выполнения запросов к другим таблицам.
	Ссылочная целлостность	Адекватное воспроизведение записей в ссылочных полях таблиц.
	Триггер	Средство обеспечения ссылочной целостности на основе механизма каскадных изменений.
	Индекс	Механизмы быстрого доступа к хранящимся в таблицах данных путем их предварительной сортировки.
	Транзакция	Такое воздействие на СУБД, которое переводит ее из одного целостного состояния в другое.

2.1.1.3.1. Ограничительные условия, поддерживающие целостность базы данных

Как следует из определения ссылочной целостности при наличии в ссылочных полях двух таблиц различного представления данных происходит нарушение ссылочной целостности, такое нарушение делает информацию в базе данных недостоверной. Чтобы предотвратить потерю ссылочной целостности, используется механизм каскадных изменений (который чаще врего реализуется специальными объектами СУБД - триггерами). Данный механизм состоит в следующей последовательсности действий:

- при изменении поля связи в записи родительской таблицы следует синхронно изменить значения полей связи в соответствующих записях дочерней таблицы;
- при удалении записи в родительской таблицы следует удалить соответствующие записи и в дочерней таблице.

2.1.1.3.2 Процесс нормализации

Нормализация - процесс приведения реляционных таблиц к стандартному виду. В базе данных могут присутствовать такие проблемы как:

- *Избыточность данных.*Повторение данных в базе данных.
- *Аномалия обновления.* Противоречивость данных, вызванная их избыточностью и частичным обновлением.
- *Аномалия удаления.* Непреднамеренная потеря данных, вызванная удалением других данных.
- *Аномалия ввода.* Невозможность ввести данные в таблицу, вызванная отсутствием других данных.

Для решения этих проблем применяют р*азбиение таблиц* - разделение таблицы на несколько таблиц. Для того чтобы это сделать пользуются нормальными формами или правилами структурирования таблиц.

Первая нормальная форма

Реляционная таблица находится в *первой нормальной форме (1НФ),* если значения в таблице являются атомарными для каждого атрибута таблицы, т.е. такими значениями, которые не являются множеством значений или повторяющейся группой. В определении Кодда реляционной модели уже заложено, что реляционные таблицы находились в 1НФ,

Вторая нормальная форма.

Реляционная таблица находится во *второй нормальной форме (2НФ),* если никакие неключевые атрибуты не являются функционально зависимыми лишь от части ключа. Таким образом, 2НФ может оказаться нарушена только в том случае, когда ключ составной.

*Функциональная зависимость.*Значение атрибута в кортеже однозначно определяет значение другого атрибута в кортеже.

Более формально можно определить функциональную зависимость следующим образом: если А и В - атрибуты в таблице В, то запись ФЗ (функциональную зависимость): А - " В обозначает, что если два кортежа в таблице В имеют одно и то же значение атрибута А, то они имеют одно и то же значение атрибута В. Это определение такжеприменимо,если А и В - множества столбцов, а не просто отдельные столбцы.

Атрибут в левой части ФЗ называется *детерминантом,* так как его значение определяет значение атрибута в правой части. Ключ таблицы является *детерминантом,* так как его значение однозначно определяет значение каждого атрибута таблицы.

Процесс разбиения на две 2НФ-таблицы состоит из следующих шагов:

1. Создается новая таблица, атрибутами которой будут атрибуты исходной таблицы, входящие в противоречащую правилу ФЗ. Детерминант ФЗ становится ключом новой таблицы.
2. Атрибут, стоящий в правой части ФЗ, исключается из исходной таблицы.
3. Если более одной ФЗ нарушают 2НФ, то шаги 1 и 2 повторяются для каждой такой ФЗ.
4. Если один и тот же детерминант входит в несколько ФЗ, то все функционально зависящие от него атрибуты помещаются в качестве неключевых атрибутов в таблицу, ключом которой будет детерминант.

Третья нормальная форма

Реляционная таблица имеет *третью нормальную форму* (ЗНФ), если для любой ФЗ: Х - У Х является ключом. Заметим, что любая таблица, удовлетворяющая ЗНФ, также удовлетворяет и 2НФ. Однако обратное неверно.

Критерийнормальной формы Бойса-Кодда (НФБК) утверждает, что таблица удовлетворяет ЗНФ, если в ней нет транзитивных зависимостей. Транзитивная зависимость возникает, если неключевой атрибут функционально зависит от одного или более неключевых атрибутов. То есть этот критерий учитывает следующие два случая:

1. Неключевой атрибут зависит от ключевого атрибута, входящего в составной ключ (критерий нарушения 2НФ).
2. Ключевой атрибут, входящий в составной ключ, зависит от неключевого атрибута.

Таким образом, если таблица удовлетворяет НФБК, то она также удовлетворяет ЗНФ в смысле транзитивных зависимостей и 2НФ.

Четвертая нормальная форма

Таблица имеет *четвертую нормальную форму (4НФ),* если она имеет ЗНФ и не содержит многозначных зависимостей. Поскольку проблема многозначных зависимостей возникает в связи с многозначными атрибутами, то мы можем решить проблему, поместив каждый многозначный атрибут в свою собственную таблицу вместе с ключом, от которого атрибут зависит.

Пятая нормальная форма.

Пятая нормальная форма (5НФ) была предложена для того, чтобы исключить аномалии, связанные с особым типом ограничительных условий, называемых *совместными зависимостями*. Эти зависимости имеют в основном теоретический интерес и сомнительную практическую ценность. Следовательно, пятая нормальная форма в действительности не имеет практического применения.

Нормальная форма область/ключ.

Таблица имеет *нормальную форму область/ключ (НФОК)*, если любое ограничительное условие в таблице является следствием определений областей и ключей. Однако не был дан общий метод приведения таблицы к НФОК.

В качестве примера, рассмотрим структуру реляционной базы данных, описывающей "отношения" пациентов и докторов в произвольной клинике (область приложения примера выбрана из-за того, что в сертификационных тестах Oracle аналогичные примеры встречаются очень часто). Пусть существует некая клиника, основные характеристики которой описываются в таблице CLINICS, в данной клинике работают доктора, основные характеристики которых описывает таблица DOCTORS. Данные пациентов клиники хранятся в таблице PATIENTS. Взаимосвязи между таблицами представлены на рис.10. (Для упрощения предполагается, что у доктора может быть несколько пациентов, которые не являются пациентами других докторов, для реализации реальной картины, когда один пациент может относиться к нескольким разным докторам, между таблицами DOCTORS и PATIENTS необходимо включить дополнительную связывающую таблицу).

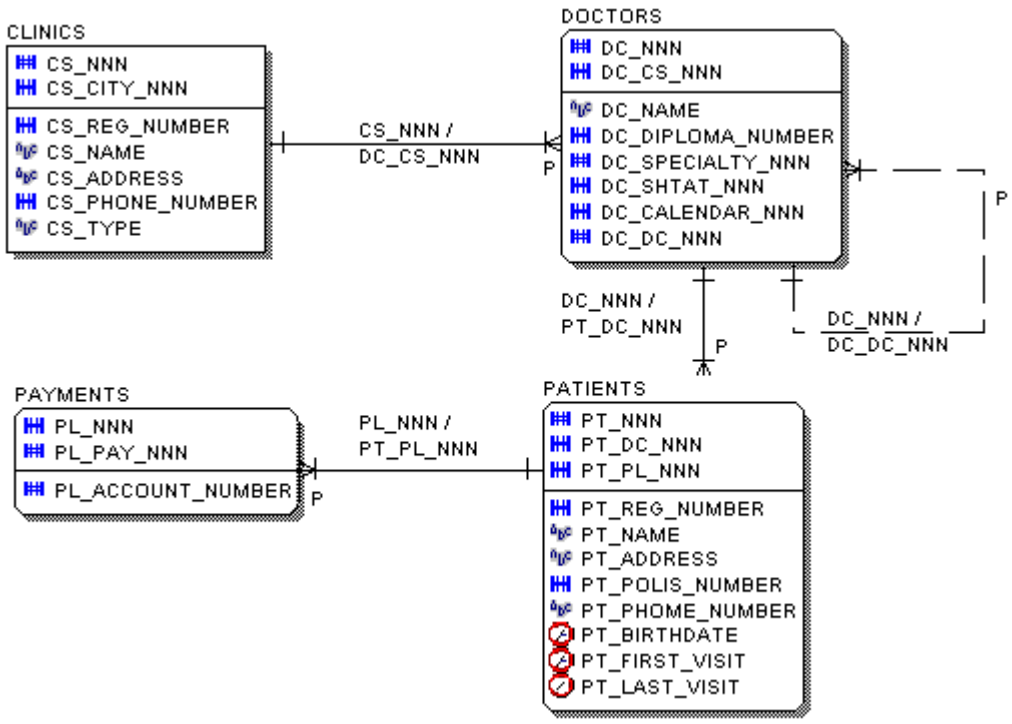


Рис.10. Диаграмма, иллюстрирующая отношения таблиц АИС.

№	Наименование столбца	Описание
Таблица CLINICS		
1	CS_NNN	Индекс
2	CS_REG_NUMBER	Регистрационный номер

3	CS_CITY_NNN	Ссылка на справочник городов и регионов
4	CS_NAME	Наименование клиники
5	CS_ADDRESS	Адрес клиники
6	CS_PHONE_NUMBER	Номер телефона
7	CS_TYPE	Профиль клиники
Таблица DOCTORS		
1	DC_NNN	Индекс
2	DC_NAME	Ф.И.О. доктора
3	DC_CS_NNN	Ссылка на таблицу CLINICS
4	DC_DIPLOM_NUMBER	Номер диплома
5	DC_SPECIALTY_NNN	Ссылка на справочник специальностей
6	DC_SHTAT_NNN	Ссылка на штатное расписание
7	DC_CALENDAR_NNN	Ссылка на расписание приема
Таблица PATIENTS		
1	PT_NNN	Индекс
2	PT_REG_NUMBER	Регистрационный номер
3	PT_NAME	Ф.И.О. пациента
4	PT_ADDRESS	Адрес пациента
5	PT_POLIS_NUMBER	Номер полиса
6	PT_PHONE_NUMBER	Номер телефона
7	PT_BIRTHDATE	Дата рождения
8	PT_FIRST_VISIT	Дата первого визита
9	PT_LAST_VISIT	Дата последнего визита
10	PL_PT_NNN	Ссылка на таблицу платежей
Таблица PAYMENTS		
	PL_NNN	Индекс
	PL_ACCOUNT_NUMBER	Номер расчетного счета
	PL_PAY_NNN	Ссылка на таблицу расчетов

Представленная структура, конечно, не обладает функциональной полнотой с точки зрения проектирования АИС клиники, с ее помощью мы лишь рассмотрим различные типы отношений в реляционных СУБД.

Перед тем, как перейти к рассмотрению вопросов стандартизации и целостности данных в РСУБД несколько рекомендаций по выбору наименований таблиц и полей. Внимательно взглянув на описание таблиц можно заметить, что генерация наименований таблиц и столбцов подчиняется некоторой синтаксической конструкции, которая в общем виде может быть представлена следующим образом:

Для таблиц:

<Псевдоним АИС>_<Псевдоним модуля АИС>_:_<Псевдоним подмодуля>_<Имя таблицы>

Например, если бы мы разрабатывали АИС клиники с сокращенным названием CSL, то все таблицы входящие в данную систему было бы целесообразно называть

CSL_<имя модуля>_<имя таблицы>.

Для столбцов:

<Псевдоним таблицы>_<имя столбца>.

Например, как показано на рис.10. Регистрационный номер пациента храниться в поле PT_REG_NUMBER, таблицы PATIENTS, имеющий псевдоним PT.

Конечно, использование этих не хитрых правил не является обязательным, но позволяет значительно облегчить читаемость разработанной информационной структуру. Предположите, как было бы все, если бы поля таблиц назывались P111, P112 и т.п., а ведь такие вещи встречаются практически очень часто, например в FoxPro 2.6.

Перейдем к рассмотрению вопросов стандартизации и обеспечения ссылочной целостности реляционных таблиц.

Преобразование отношений

Поля таблиц могут находиться между собой в одном из следующих отношений:

один-к-одному, один-ко-многим, многие-ко-многим и рекурсивных, определения которых приведены в табл.1. Рассмотрим преобразование отношений на примере АИС "ДОКТОР-ПАЦИЕНТ" (рис.10).

Отношение один-к-одному представляет собой такое отношение, при котором каждой записи в таблице А соответствует единственная запись в таблице В (рис.11). Применение такого типа отношений встречается крайне редко и предназначено в основном для функционального разделения информации на несколько таблиц, т.е. когда не хотят, чтобы таблица БД "распухала" от второстепенной информации. На рис.10 представлено, как используя отношение один-к-одному таблица PATIENTS преобразована в две таблицы: PATIENTS_REG и PATIENTS_KART (на рисунке показаны только основные атрибуты таблиц). Также необходимо принимать во внимание, что БД использующие такие отношения не могут быть полностью нормализованы.

PATIENTS_REG

PT_REG_NNN	PT_REG_NUMBER
1	CS112/99-07
2	CS102/98-01
3	CS098/96-56
4	CS546/99-76
5	CS234/98-02

PATIENTS_KART

PT_NNN	PT_NAME	PT_BIRTHDATE
1	ИВАНОВ А.И.	01.02.1987
2	ПЕТРОВ А.А.	30.10.1945
3	СИДОРОВ В.Н.	03.12.1965
4	КРАСНОВ Б.Б.	10.11.1999
5	ВАСИН Н.Н.	15.05.1976

Рис.11. Отношение один к одному.

Отношение один-ко-многим можно без преувеличения назвать основным типом отношений использующемся при проектировании современных БД, так как позволяет представлять иерархические структуры данных. Под данным

отношение понимается такое отношение, когда одной записи в родительской таблице соответствуют записи в дочерней таблице (причем число соответствующих записей выражается рядом натуральных чисел 0,1,2,:N и т.п.) (рис.12). Отношения один-ко-многим могут быть жесткими и нежесткими. Для жестких отношений должно выполнять требование, что каждой записи в родительской таблице должна соответствовать хотя бы одна запись в дочерней таблице.

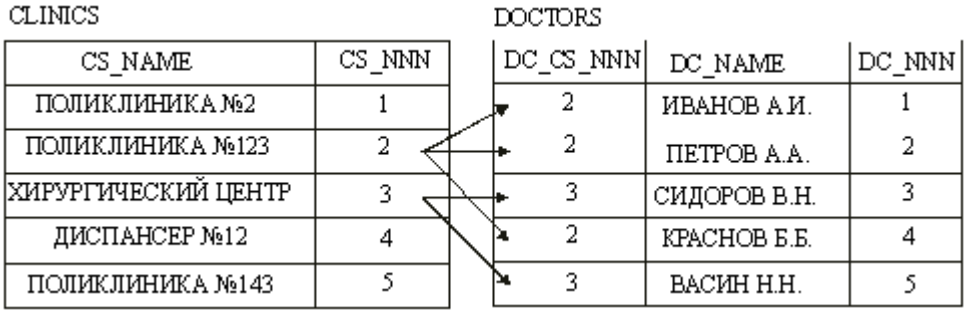


Рис.12. Отношение один ко многим.

Отношение многие-ко-многим представляет собой отношение при котором записям родительской таблицы соответствуют записи дочерней таблицы, а ряду записей дочерней таблицы соответствуют записи в родительской таблицы (рис.13). Использование такого типа отношений крайне ограничено, не только из-за того, что некоторые БД его вообще не поддерживают на уровне индексов и ссылочной целостности, но и потому, что практически любое отношение многие-ко-многим может быть заменено одним или более отношением один-ко-многим (посмотрите на пример на рис.13. и так не когда не делайте).

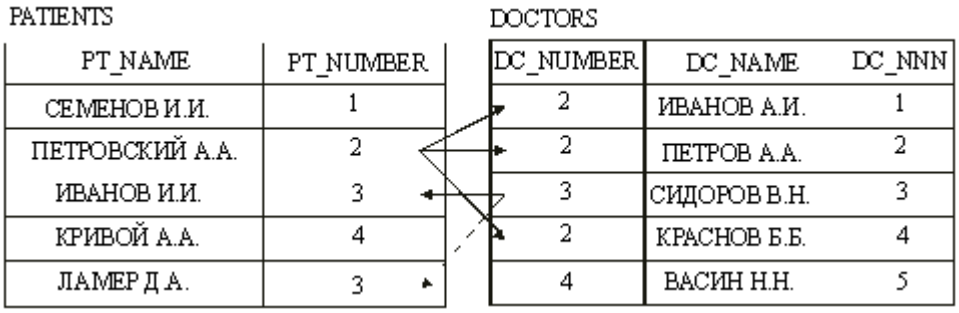


Рис.13. Отношение многие ко многим.

Другим важным типом отношений - является рекурсивное отношение, т.е. такое отношение которое описывает связи между записями внутри одной таблицы БД, т.е. оно связывает объектное множество с ним самим. Пример рекурсивных отношений показан на рис.14., который иллюстрирует, что доктора Петров А.А. и Васин Н.Н. находятся в зависимости от доктора Сидорова В.Н.. В зависимости от функционального назначения этого отношения оно может иллюстрировать, например, что они являются пациентами доктора Сидорова В.Н., или Сидорова В.Н. является по отношению к ним начальником и т.п. Данный тип отношений позволят реализовать древовидную структуру функциональных отношений, например, структуру организации.

DOCTORS

DC_DC_NNN	DC_NAME	DC_NNN
2	ИВАНОВ А.И.	1
3	ПЕТРОВ А.А.	2
2	СИДОРОВ В.Н.	3
2	КРАСНОВ Б.Б.	4
3	ВАСИН Н.Н.	5

Рис.14. Отношение многие ко многим.

Учитывая требования ссылочной целостности и нормализации на основе применения рассмотренных выше типов отношений осуществляется преобразование функциональной модели бизнес - процессов и реаляционную модель. Итогом этапа является диаграмма "Сущность-связь" (часто называемая CASE диаграмма, ER-диаграама, рис.10).

Замечания

Например:

1)Что такое традиционная база данных ?
Бывают сетевые, иерархические и реляционные БД. Последние, в свою очередь делятся на СУБД для решения задач оперативного управления транзакциями (OLTP) и системы принятия решений (DSS).
Почему "традиционная база данных"- это база в разделямых файлах? 2) "Ведь в реляционной безе данных проблемы синхронизации данных не возникает вовсе" - очень опасно так говорить. Если USER читает не с начала,- то он может спутать это с синхронизацией транзакций,- а это проблема ключевая в рел-ых СУБД.
3)Подсистемы RDBMS очень схожи с соответствующими подсистемами ОС и сильно интегрированы с предоставляемыми базовой ОС сервисными функциями - я бы так
не говорил.
4) , и администратор базы данных должен будет восстанавливать часть или всю
БД, используя файлы резервных копий (если их сделали!) - кого-копии или администраторов, когда они не делают копии?
5)SYS или SYSTEM, с парролем: master или manager - а также Amum13:). (на самом деле - CHANGE_ON_INSTALL и MANAGER)
Что касаемо, сосбтвенно, описания конкретно Oracle, - то вроде ничего странного я там не видел.

2.1.1.3.3 Преобразование функциональной модели в реляционную.

В разделе 1.3. нами были рассмотрены основные этапы разработки автоматизированной информационной системы, в разделе 1.3.1 мы разработали функциональную модель АИС, теперь, после того как мы рассмотрели основные оложения терии баз данных, пришло время заняться непосредственно формализацией выделенных бизнес-процессов, операций и т.п. Результатом первого этапа проектирования АИС является функциональная модель системы содержащая множество объектов (процессов, операций), их атрибутов.

Объектное множество с атрибутами может быть преобразовано в реляционную таблицу с именем объектного множества в качестве имени таблицы и атрибутами объектного множества в качестве атрибутов таблицы. Если некоторый набор этих атрибутов может быть использован в качестве ключа таблицы, то он выбирается ключом таблицы. В противном случае мы добавляем к таблице атрибут, значения которого будут однозначно определять объекты-элементы исходного объектного множества, и который, таким образом, может служить ключом таблицы.

Преобразование отношений

Поля таблиц могут находиться между собой в обном из следующих отношений: один-к-одному, один-ко-многим, многие-ко-многим и рекурсивных, определения которых представлены в табл.1. Прежде чем рассмотреть реализацию и преобразование отношений более подробно, обсудим реторический вопрос о правилах именования таблиц и столбцов. Как мы уже ранее отмечали, что практически любая АИС имеет модульную структуру и соответственно, в каждый модель входит определенное число таблиц. Пусть имеется модуль "Операционный День", условно назовем его OPDAY, тогда удобно, что все таблицы данного модуля наименовались следующим образовам OPDAY_CUSTOMERS (ТАБЛИЦА КЛИЕНТОВ), OPDAY_ACCOUNT (таблица счетов) и т.п. При наменовании столбцов таблицы желательно придерживаться следующего подхода: <краткое наименование таблицы>_<наименование столбца>. Например, для таблицы OPDAY_CUSTOMERS наименование столбцов удобно реализовать следующим образом CUST_NNN (порядковый номер записи), CUST_FIO (фио клиента), CUST_ACCOUNT_NNN (ссылка на таблицу счетов) и т.п. Практически в каждой организации, занимающейся разработкой АИС существуют свои нормы к наименованию модулей, таблиц, столбцов и объектов базы данных, однако общие принципы во многом схожи с приведенным в данных примерах. Теперь рассмотрим основные принципы преобразования отношений:

Отношение один-к-одному.

Рассмотрим пример установки отношений клиентов и счетов в АБС (см. рис.6).

Рис.8. Отношение один к одному.

Отношение ИМЕЕТ ТЕКУЩИЙ СЧЕТ представляет собой связь один-к-одному. Это означает, что клиент имеет не более одного текущего счета и каждым текущим счетом пользуется только один клиент. Если мы решим, что ключами являются №-КЛИЕНТА для CUSTOMER (КЛИЕНТ) и №-ТЕКУЩЕГО-СЧЕТА для ACCOUNT_NUMBER (ТЕКУЩИЙ СЧЕТ), то мы получим две реляционные таблицы, каждая из которых состоит из одного столбца.

CUSTOMER (CUST_NNN)

ACCOUNT (ACCOUNT_NUMBER)

Для того чтобы показать связь между этими двумя таблицами, мы должны включить ссылку на ACCOUNT_NUMBER в таблицу CUSTOMER и и ссылку на

CUST_NNN в таблицу ACCOUNT. Каждый из этих столбцов будет внешним ключом, указывающим на другую таблицу.

CUSTOMER (CUST_NNN, CUST_ACCOUNT_NUMBER)

Внешний ключ: CUST_ACCOUNT_NUMBER ссылается на ACCOUNT_NUMBER.

ACCOUNT (ACCOUNT_NUMBER, ACCOUNT_CUST_NNN)

Внешний ключ: ACCOUNT_CUST_NNN ссылается на CUST_NNN.

Резюме: отношение один-к-одному преобразуется путем помещения одного из объектных множеств в качестве атрибута в таблицу второго объектного множества. Его выбор определяется потребностями конкретного приложения. Во многих случаях оба варианта приемлемы.

Отношение один-ко-многим.

Предположим, что отношение ИМЕЕТ-ТЕКУЩИЙ-СЧЕТ имеет мощность "много" со стороны ACCOUNT.

Рис.9. Отношение один ко многим.

Это означает, что у клиента может быть несколько текущих счетов, но каждым текущим счетом по-прежнему пользуется только один клиент. Таким образом, в любом отношении один-ко-многим в. таблицу, описывающую объект, мощность со стороны которого равна "многим", включается столбец, являющийся внешним ключом, указывающим на другой объект.

Отношение многие-ко-многим.

Отношение ИМЕЕТ-ТЕКУЩИЙ-СЧЕТ имеет мощность многие-ко-многим.

Рис.10. Отношение многие ко многим.

Таким образом, мы предполагаем, что у клиента может быть несколько текущих счетов, и что каждым текущим счетом могут пользоваться несколько клиентов. Для того чтобы преобразовать отношение многие-ко-многим целесообразно создать таблицу пересечения, представляющую элементы двух других таблиц, находящихся в отношении многие-ко-многим.

Рекурсивные отношения

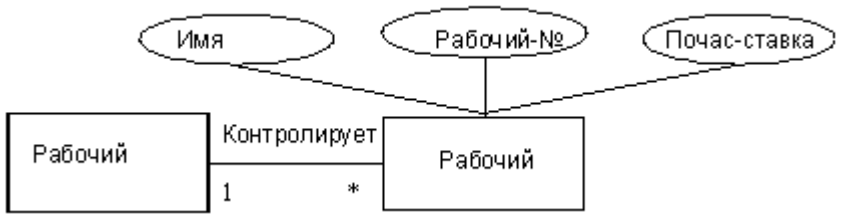


Рис.11. Рекурсивные отношения.

Объектное множество WORKER(РАБОЧИЙ), дважды встречающееся на диаграмме, и это одно и то же объектное множество в обоих случаях. Обе копии объектного множества WORKER(РАБОЧИЙ) имеют одни и те же атрибуты. В этой модели два экземпляра объектного множества WORKER(РАБОЧИЙ) использованы для удобства, чтобы показать отношение SUPERVISES(КОНТРОЛИРУЕТ), существующее между объектами WORKER(РАБОЧИЙ) и WORKER(РАБОЧИЙ). Это отношение называется *рекурсивным*, поскольку оно связывает объектное множество с ним самим. В данном случае отношение мощности один-ко-многим означает, что одному работнику подчиняются несколько других работников.

WORKER (WORKER-ID, NAME, HOURLY-RATE, WORKER-ID)

Чтобы преобразовать объектное множество WORKER вместе с его атрибутами и отношением SUPERVISES в реляционную таблицу нужно изменить имя второго атрибута WORKER-ID на имя, соответствующее отношению SUPERVISES, которое оно представляет. SUPV-ID.

WORKER (WORKER-ID, NAME, HOURLY-RATE, SUPV-ID)

Внешний ключ: SUPV-ID ссылается на WORKER

SUPV-ID - это рекурсивный внешний ключ, поскольку он ссылается на WORKER-ID, то есть ключ своей собственной таблицы. Таким образом, в результате преобразования рекурсивных отношений появляются рекурсивные внешние ключи.

Функциональные зависимости, определенные для реляционной модели, являются атрибутами отношения один-к-одному или один-ко-многим.

Описанный процесс преобразования каждой из этих конструкций в атрибуты реляционных таблиц гарантирует, что они будут зависеть только от ключевых атрибутов. Таким образом, каждая полученная реляционная таблица будет иметь ЗНФ. Многозначные атрибуты реляционной модели встречаются только в отношениях многие-ко-многим. Поскольку они преобразуются в реляционные таблицы, обладающие составными ключами из ключевых атрибутов отдельных объектных множеств, то они гарантированно имеют 4НФ.

2.1.2. Понятие языка определения данных (ЯОД - DBTG)

Язык - средство, при помощи которого определяется структура данных или схема, а также происходит запоминание данных и манипуляция ими. Язык, которым определяется схема, называется *языком определения данных (ЯОД),* а язык, используемый для запоминания данных и манипуляции ими, называется *языком манипуляции данными (ЯМД).*

Процедура применения ЯОД и определения схемы такова:

1. Создается концептуальная модель данных.
2. Концептуальная модель данных преобразуется в диаграмму сетевой структуры данных.
3. Проверяется, существуют ли между типами записей отношения один-ко-многим. Они могут быть непосредственно реализованы в виде наборов DBTG.
4. Если есть отношения мощности многие-ко-многим, то каждое из них преобразуется в два набора путем создания записи связи.
5. Если есть n-арные отношения, то они преобразуются в бинарные отношения.
6. Применяется ЯОД для реализации схемы.

Схема состоит из следующих частей:

1. *Раздел схемы.* Раздел схемы DBTG, задающий имя схемы.
2. *Раздел записей.* Раздел схемы DBTG, определяющий каждую запись: ее элементы данных и ее адрес.
3. *Раздел наборов.* Раздел схемы DBTG, определяющий наборы, включая типы записей владельцев и членов.

Подсхемы - это в основном, подмножества схемы. В подсхеме могут быть сгруппированы элементы данных, которые не были сгруппированы в схеме; записи и наборы могут быть переименованы и порядок описаний может быть изменен.

Принятого стандарта DBTG для подсхемы не существует; однако, обычно используются следующие отделы:

1. Отдел заголовка, позволяющий присвоить имя подсхеме и указать связанную с ней схему.
2. Отдел преобразования, в котором при желании производится замена имен из схемы на нужные в подсхеме.
3. Структурный отдел, в котором задается, какие записи, элементы данных и наборы из схемы должны присутствовать в подсхеме. Этот отдел состоит из разделов записей и наборов.

Раздел записей подсхемы. Раздел структурного отдела, в котором задаются записи, элементы данных и типы данных подсхемы.

Раздел наборов подсхемы. Раздел структурного отдела, в котором задаются наборы, которые должны быть включены в подсхему.

Подсхема позволяет пользователю строить из предопределенной схемы схему, соответствующую нуждам конкретного приложения.

2.1.3. Язык манипуляции данными (ЯМД)

Язык манипуляции данными (ЯМД) обеспечивает эффективные команды манипуляции сетевой системой базы данных. ЯМД позволяет пользователям выполнять над базой данных операции в целях получения информации, создания отчетов, а также обновления и изменения содержимого записей.

Основные команды ЯМД можно классифицировать следующим образом: команды передвижения, команды извлечения, команды обновления записей, команды обновления наборов.

Табл.2. Основные типы команд ЯМД.

№	Наименование типа команд	Назначение
1	*Команды передвижения.*	Команды, применяемые для поиска записей базы данных.
2	*Команды извлечения.*	Команды, применяемые для извлечения записей базы данных.
3	*Команды обновления записей.*	Команды, применяемые для изменения значений записей.
4	*Команды обновления наборов.*	Команды, применяемые для добавления, изменения или удаления экземпляров наборов.

Заключение

Процесс преобразования функциональной модели в реляционную включает создание реляционной таблицы для каждого объектного множества модели. Атрибуты объектного множества становятся атрибутами реляционной таблицы. Если в функциональной модели существует ключевой атрибут, то он может использоваться в качестве ключа реляционной таблицы. В противном случае ключевой атрибут таблицы может быть создан аналитиком. Однако, лучше всего, если такой атрибут естественным образом возникает из моделируемого приложения. Отношения один-к-одному и один-ко-многим преобразуются в реляционную модель путем превращения их в атрибуты соответствующей таблицы. Отношения многие-ко-многим соответствуют многозначным атрибутам и преобразуются в четвертую нормальную форму путем создания ключа из двух столбцов, соответствующих ключам двух объектных множеств, участвующих в отношении. Конкретизирующие множества преобразуются путем создания отдельных реляционных таблиц, ключи которых совпадают с ключами обобщающих объектных множеств. Рекурсивные отношения также можно смоделировать, создав новое смысловое имя атрибута, обозначающее отношение.

2.2. Архитектуры реализации корпоративных информационных систем.

При построении корпоративных информационных сетей, как правило, используются две базовые архитектуры: Клиент-сервер и Интернет/Интранет. В чем же преимущества и недостатки использования каждой из данных архитектур и когда их применение оправдано? Найти ответы на эти вопросы мы постараемся в данном разделе.

Одной из самых распространенных на сегодня архитектур построения корпоративных информационных систем является архитектура КЛИЕНТ-СЕРВЕР.

Рис.12. Компоненты архитектуры Клиент-сервер и их свойства.

В реализованной по данной архитектуре информационной сети клиенту предоставлен широкий спектр приложений и инструментов разработки, которые ориентированы на максимальное использование вычислительных возможностей клиентских рабочих мест, используя ресурсы сервера в основном для хранения и обмена документами, а также для выхода во внешнюю среду. Для тех программных систем, которые имеют разделение на клиентскую и серверную части, применение данной архитектуры позволяет лучше защитить серверную часть приложений, при этом, предоставляя возможность приложениям либо непосредственно адресоваться к другим серверным приложениям, либо маршрутизировать запросы к ним. Средством (инструментарием) для реализации клиентских модулей для ОС Windows в данном случае является, как правило, Delpfi.

Однако при этом частые обращения клиента к серверу снижают производительность работы сети, кроме этого приходится решать вопросы безопасной работы в сети, так как приложения и данные распределены между различными клиентами. Распределенный характер построения системы обуславливает сложность ее настройки и сопровождения. Чем сложнее структура сети, построенной по архитектуре КЛИЕНТ-СЕРВЕР, тем выше вероятность отказа любого из ее компонентов.

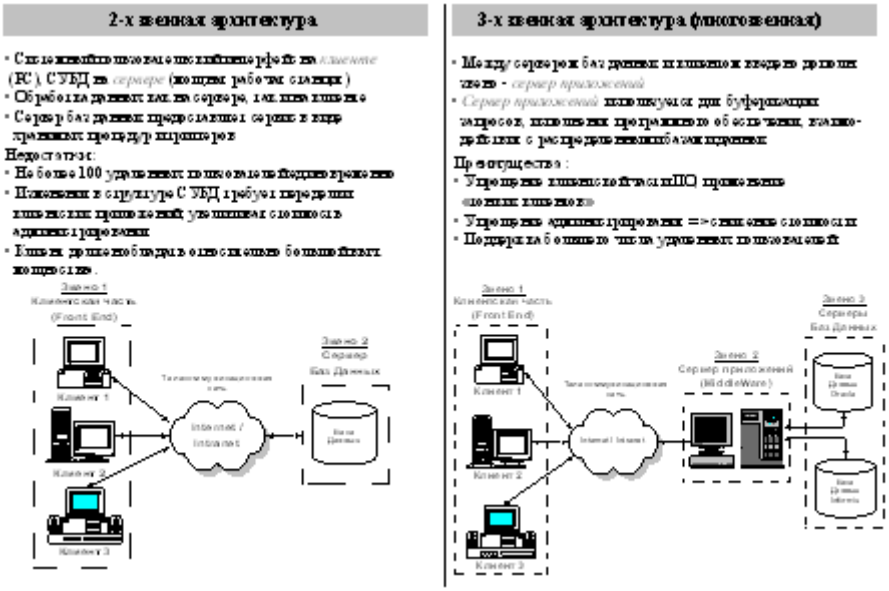


Рис.13. Сравнительные характеристики двух- и трехзвенной архитектуры клиент-сервер.

В последнее время все большее развитие получает архитектура Интернет/Интранет. В основе реализации корпоративных информационных систем на базе данной архитектуры лежит принцип "открытой архитектуры", что во многом определяет независимость реализации корпоративной системы от конкретного производителя. Все программное обеспечение таких систем реализуется в виде аплетов или сервлетов (программ написанных на языке JAVA) или в виде cgi модулей (программ написанных как правило на Perl или C).

Основными экономическими преимуществами данной архитектуры являются:

- относительно низкие затраты на внедрение и эксплуатацию;
- высокая способность к интеграции существующих гетерогенных информационных ресурсов корпораций;
- повышение уровня эффективности использования оборудования (сохранение инвестиций).
- прикладные программные средства доступны с любого рабочего места, имеющего соответствующие права доступа;
- минимальный состав программно-технических средств на клиентском рабочем месте (теоретически необходима лишь программа просмотра - броузер и общесистемное ПО);
- минимальные затраты на настройку и сопровождение клиентских рабочих мест, что позволяет реализовывать системы с тысячами пользователей (причем многие из которых могут работать за удаленными терминалами).

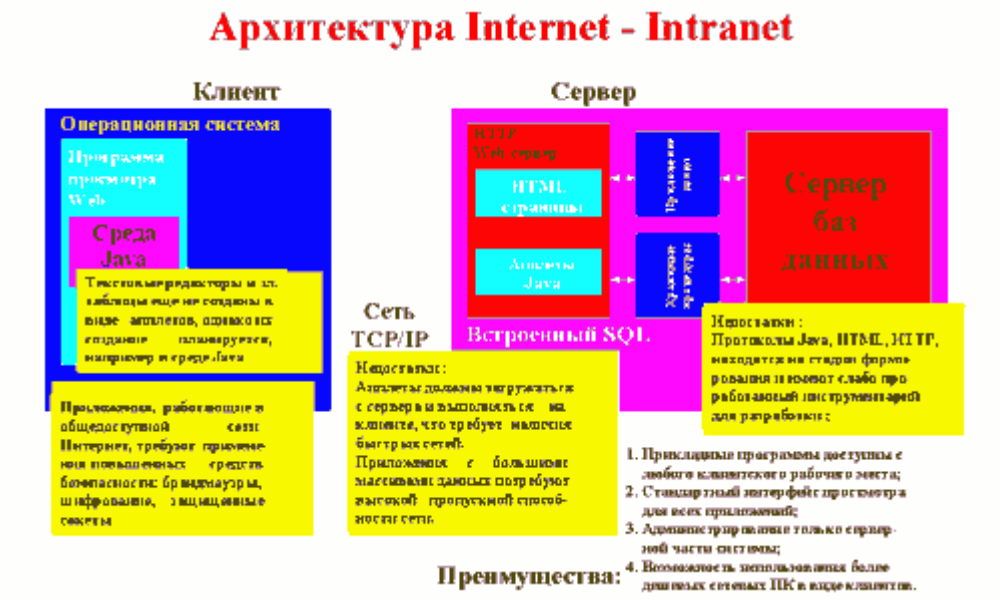


Рис.14 Компоненты архитектуры Интернет-Интранет и их свойства.

В общем случае АИС, реализованная с использованием данной архитектуры включает Web-узлы с интерактивным информационным наполнением, реализованных при помощи технологий Java, JavaBeans и JavaScript, взаимодействующих с предметной базой данных, с одной стороны, и с клиентским местом с другой. База данных, в свою очередь, является источником информации для интерактивных приложений реального времени.

По запросу клиента WEB узел осуществляет следующие операции (рис.7):

- Отправляет ASCII коды HTML страниц (или VRML документов), включающие при необходимости элементы javaScript;
- Отсылает двоичный код запрошенного ресурса (изображения, адио-, видеофайла, архива и т.п.);
- Отсылает байт коды JAVA апплетов.
- Принимает конкретную информацию от пользователя (результат заполнения активной формы, или статистическую информацию запрошенную CGI скриптом);
- Осуществляет заполнение базы данных;
- Принимает сообщения от пользователя и регламентирует доступ к ресурсам Web узла на основе анализа принятой информации (проверка паролей и т.п.);
- Принимает информацию от пользователя и в зависимости от нее динамически формирует HTML страницы, либо VRML документы, обращаясь, при необходимости, к базам данных и существующим на WEB узле HTML страницам и VRML документам.

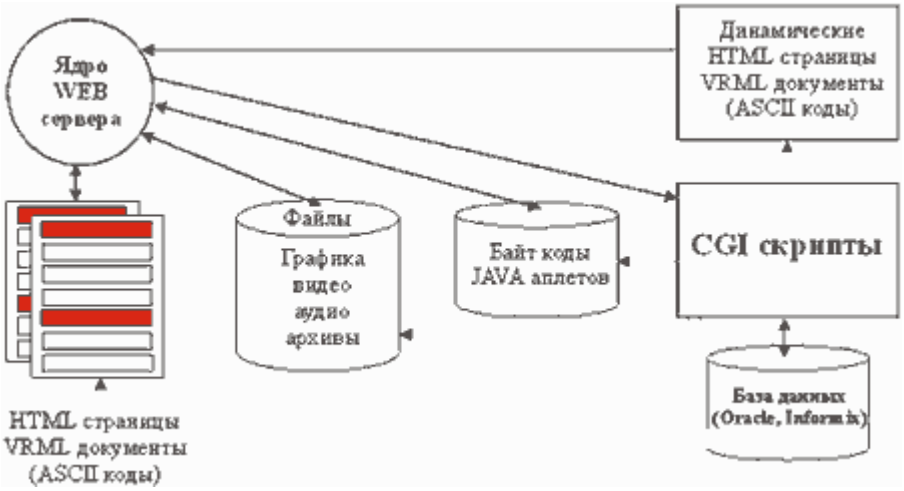


Рис.15. Информационные взаимосвязи компонентов WEB узла

После того, как клиент получил ответ WEB сервера, он осуществляет следующие операции:

- визуализирует HTML страницу либо VRML документ в окне броузера;
- интерпретирует команды JavaScript, модифицирует образ HTML страницы и т.п.;
- интерпретируя байт коды JAVA апплетов, позволяет загружать и выполнять активные приложения;
- ведет диалог с пользователем, заполняющим формы, и создает новые запросы к WEB серверу;
- с помощью утилит воспроизводит коды аудио и видео файлов, поддерживает мультимедийные средства;
- обеспечивает моделирование виртуальной реальности просматривая VRML документы.

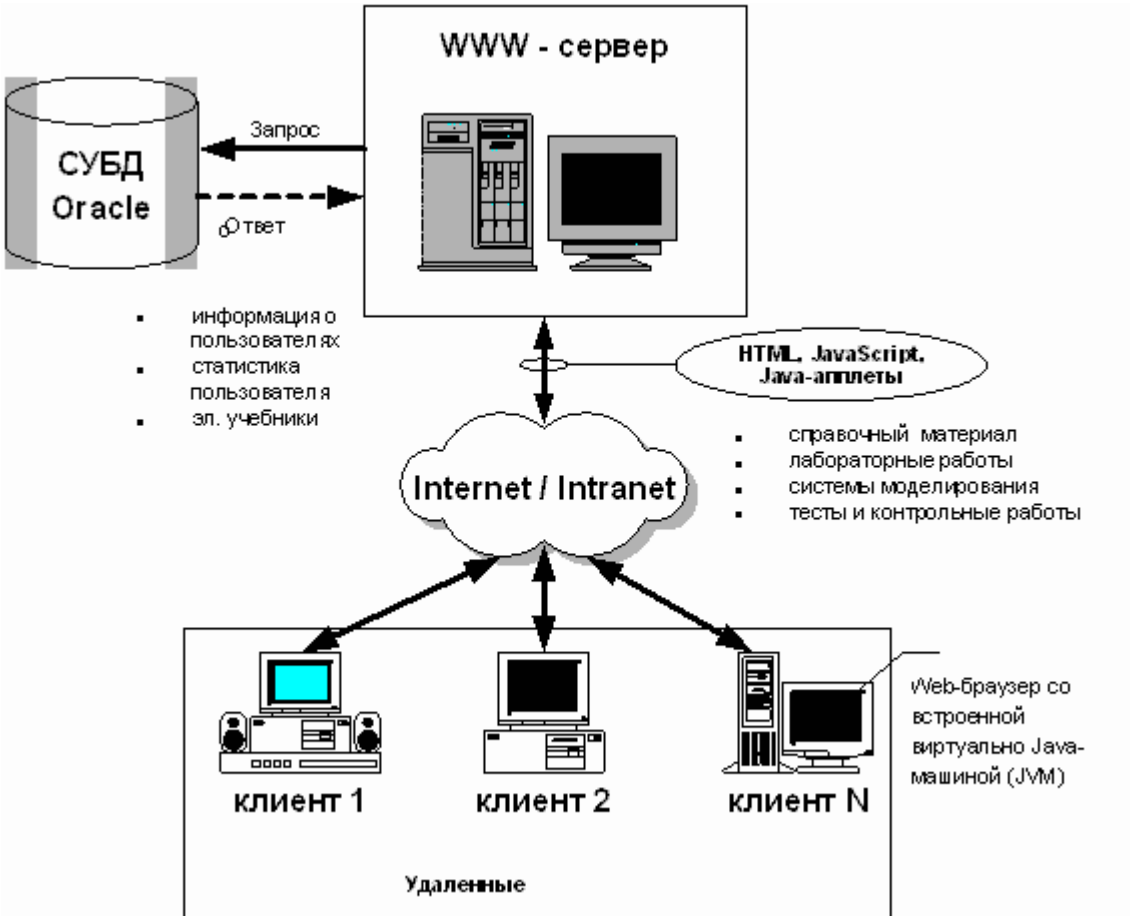


Рис.16. Функциональная схема интерактивного взаимодействия пользователей в архитектуре интернет/интранет.

Перечисленные задачи WEB клиента обеспечиваются возможностями броузера и специализированным программным обеспечением (утилитами), размещенными на рабочей станции клиента. Следует отметить и тот факт, что жестких стандартов на построение WEB клиента пока нет и его компонентный состав может различаться.

На сегодняшний день известны и широко применяются три основных технологии создания интерактивного взаимодействия с пользователем в Web. Первый путь заключается в использовании Стандартного Интерфейса Шлюза (Commom Gateway Interface) - CGI. Второй - включение JavaScript - сценариев в тело Web-страниц. И наконец самый мощный, предоставляющий практически неограниченные возможности способ - применение технологии Java (ипользование Java-аплетов).

CGI - это механизм для выбора, обработки и форматирования информации. Возможность взаимодействия, обеспечиваемая CGI, предоставляется во многих формах, но в основном это динамический доступ к информации, содержащейся в базах данных. Например, многие узлы применяют CGI для того, чтобы пользователи могли запрашивать базы данных и получать ответы в виде динамически сформированных Web-страниц (рис.17).

Имеются в виду узлы, предоставляющие доступ к базам данных, средствам поиска, и даже информационные системы, предающие сообщения в ответ на ввод пользователя. Все эти узлы используют CGI, чтобы принять ввод пользователя и передать его с сервера Web базе данных. База данных обрабатывает запрос и возвращает ответ серверу, который в свою очередь пересылает его опять

броузеру для отображения. Без CGI база данных этого не смогла бы. Данный интерфейс можно считать посредником между броузером, сервером и любой информацией которая должна передаваться между ними.

В отличии от HTML, CGI не является языком описания документов. Собственно, это и не язык вообще; это стандарт. Он просто определяет, как серверы Web передают информацию, используя приложения, исполняемые на сервере. Это способ расширения возможностей сервера Web без преобразования при этом его самого. Подобно тому как броузер Web обращается к вспомогательным приложениям для обработки информации, которую он не понимает, CGI предоставляет серверу Web возможность преложить работу на другие приложения, такие как базы данных и средства поиска.

Рис.17. Схема взаимодействия между броузером, сервером и сценарием CGI

При написании программы шлюза (которая может конвертировать ввод из одной системы в другую) CGI позволяет использовать почти любой язык программирования. Способность использовать при написании программы шлюза любой язык, даже язык сценариев, чрезвычайно важна. Самыми популярными языками являются shell, Perl, C и C++. Сценарием традиционно называют программу, которая выполняется с помощью интерпретатора, выполняющего каждую строку программы по мере ее считывания.

Последовательность действий при взаимодействии клиента с программой запущенной на Web-сервере можно сформулировать как следующая последовательность шагов.

1. Броузер принимает введенную пользователем информацию, как правило с помощью форы.

2. Броузер помещает введенную пользователем информацию в URL, указывающий имя и местоположение сценария CGI, который требуется ввести в действие.
3. Броузер подключается к серверу Web и запрашивает URL. Сервер определяет, что URL должен ввести в действие сценарий CGI, и запускает указанный сценарий.
4. Сценарий CGI выполняется, обрабатывая все передаваемые ему данные.
5. Сценарий CGI динамически формирует Web-страницу и возвращает результат серверу.
6. Сервер возвращает результат клиенту.
7. Броузер отображает результат пользователю

Это является упрощенной схемой взаимодействия между броузером, сервером и сценарием CGI. Наибольшую популярность CGI - сценарии нашли при использовании в качестве обработчиков форм, средства доступа к базам данных, средства осуществления локального и глобального поиска, шлюзовых протоколов.

Наибольшей мощью в реализации клиентского программного обеспечения обладают аплеты - программы написанные на языке JAVA. В узком смысле слова Java - это объектно-ориентированный язык, напоминающий C++, но более простой для освоения и использования. В более широком смысле Java - это целая технология программирования, изначально рассчитанная на интеграцию с Web-сервисом, то есть на использование в сетевой среде. Поскольку Web-навигаторы существуют практически для всех аппаратно-программных платформ, Java-среда должна быть как можно более мобильной, в идеале полностью независимой от платформы.

С целью решения перечисленных проблем были приняты, помимо интеграции с Web-навигатором, два других важнейших постулата.

1. Была специфицирована виртуальная Java-машина (JVM), на которой должны выполняться (интерпретироваться) Java-программы. Определены архитектура, представление элементов данных и система команд Java-машины. Исходные Java-тексты транслируются в коды этой машины. Тем самым, при появлении новой аппаратно-программной платформы в портировании будет нуждаться только Java-машина; все программы, написанные на Java, пойдут без изменений.
2. Определено, что при редактировании внешних связей Java-программы и при работе Web-навигатора прозрачным для пользователя образом может осуществляться поиск необходимых объектов не только на локальной машине, но и на других компьютерах, доступных по сети (в частности, на WWW-сервере). Найденные объекты загружаются, а их методы выполняются затем на машине пользователя.

Несомненно, между двумя сформулированными положениями существует тесная связь. В компилируемой среде трудно абстрагироваться от аппаратных особенностей компьютера, как трудно (хотя и можно) реализовать прозрачную динамическую загрузку по сети. С другой стороны, прием объектов извне требует повышенной осторожности при работе с ними, а, значит, и со всеми Java-программами. Принимать необходимые меры безопасности проще всего в интерпретируемой, а не компилируемой среде. Вообще, мобильность, динамизм и безопасность - спутники интерпретатора, а не компилятора.

Принятые решения делают Java-среду идеальным средством разработки интерактивных клиентских компонентов (апплетов) Web-систем. Особо отметим

прозрачную для пользователя динамическую загрузку объектов по сети. Из этого вытекает такое важнейшее достоинство, как нулевая стоимость администрирования клиентских систем, написанных на Java. Достаточно обновить версию объекта на сервере, после чего клиент автоматически получит именно ее, а не старый вариант. Без этого реальная работа с развитой сетевой инфраструктурой практически невозможна.

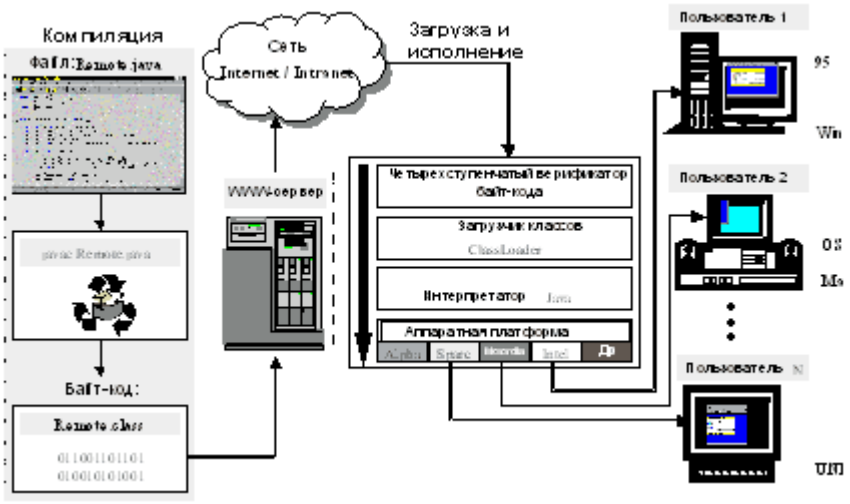


Рис.18. Java технологии при реализации АИС.

Стандартный реляционный доступ к данным очень важен для программ на Java, потому что Java-апплеты по природе своей не являются монолитными, самодостаточными программами. Будучи модульными, апплеты должны получать информацию из хранилищ данных, обрабатывать ее и записывать обратно для последующей обработки другими апплетами. Монолитные программы могут себе позволить иметь собственные схемы обработки данных, но Java-апплеты, пересекающие границы операционных систем и компьютерных сетей, нуждаются в опубликовании открытых схем доступа к данным.

Интерфейс JDBC (Java Database Connectivity - связанность баз данных Java) является первой попыткой реализации доступа к данным из программ Java, не зависящего от платформы и базы данных. В версии JDK 1.1 JDBC является составной частью основного Java API.

JDBC - это набор реляционных объектов и методов взаимодействия с источниками данных. Программа на языке Java открывает связь с таблицей, создает объект оператор, передает через него операторы SQL системе управления базой данных получает результаты и служебную информацию о них. В типичном случае файлы .class JDBC и апплет/приложение на языке Java находятся на компьютере клиенте. Хотя они могут быть загружены из сети, для минимизации задержек во время выполнения лучше иметь классы JDBC у клиента. Система управления базой данных (СУБД) и источник данных обычно расположены на удаленном сервере.

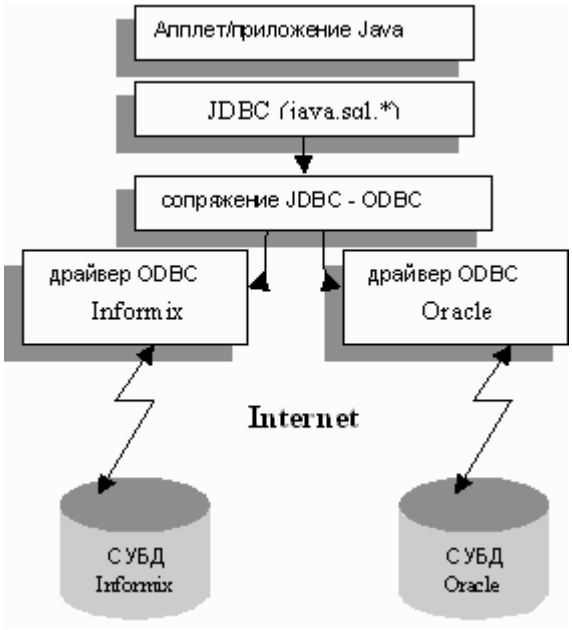
На рисунке 19 показаны различные варианты реализаций связи JDBC с базой данных. Апплет/приложение взаимодействует с JDBC в системе клиента, драйвер отвечает за обмен информацией с базой данных через сеть.

Классы JDBC находятся в пакете java.sql.*. Все программы Java используют объекты и методы из этого пакета для чтения и записи в источник данных. Программе, использующей JDBC, требуется драйвер к источнику данных, с

которым она будет взаимодействовать. Этот драйвер может быть написан на другом (не Java) языке программирования, или он может являться программой на языке Java, которая общается с сервером, используя RPC (Remote Procedure Call) - удаленный вызов процедур или HTTP. Обе схемы приведены на рис.19. Драйвер JDBC может быть библиотекой на другом (не Java), как программа сопряжения ODBC - JDBC, или классом Java, который общается через сеть с сервером базы данных, используя RPC или HTTP.

Допускается, что приложение будет иметь дело с несколькими источниками данных, возможно, с неоднородными. По этой причине у JDBC есть диспетчер драйверов, чьи обязанность заключаются в управлении драйверами и предоставлении программе списка загруженных драйверов.

Хотя словосочетание "База данных" входит в расшифровку аббревиатуры JDBC, форма, содержание и расположение данных не интересуют программу Java, использующую JDBC, поскольку существует драйвер к этим данным.



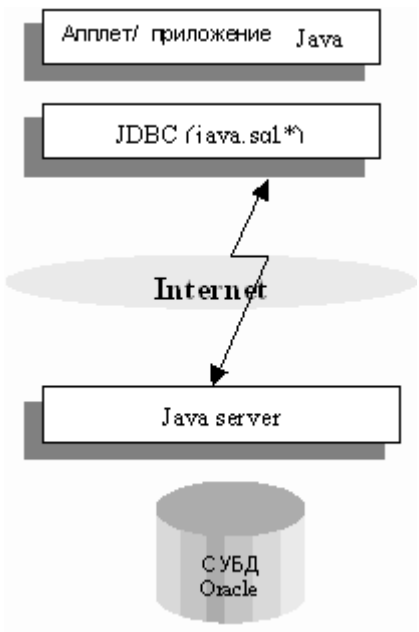


Рис.19 Варианты реализации связи JDBC с базой данных

Сопряжение JDBC - ODBC

В качестве составной части JDBC поставляется драйвер для доступа из JDBC к источникам данных ODBC (Open Database Connectivity), и называется "программа сопряжения JDBC - ODBC". Эта программа сопряжения реализована в виде JdbcOdbc.class и является библиотекой для доступа к драйверу ODBC.

Поскольку JDBC конструктивно близок к ODBC, программа сопряжения является несложной надстройкой над JDBC. На внутреннем уровне этот драйвер отображает методы Java в вызовы ODBC и тем самым взаимодействует с любым ODBC - драйвером. Достоинство такой программы сопряжения состоит в том, что JDBC имеет доступ к любым базам данных, поскольку ODBC - драйверы распространены очень широко.

В соответствии с правилами Internet JDBC идентифицирует базу данных при помощи URL, который имеет форму:

jdbc:<субпротокол>:<имя, связанное с СУБД или Протоколом>

У баз данных в Internet/intranet "имя" может содержать сетевой URL

//<имя хоста>:<порт>/..

<*субпротокол*> может быть любым именем, которое понимает база данных. Имя субпротокола "odbc" зарезервированно для источников данных формата ODBC. Типичный JDBC URL для базы данных ODBC выглядит следующим образом:

jdbc:odbc:<DNS - имя ODBC>;User=<имя пользователя>; PW=<пароль>

Внутреннее устройство JDBC - приложения

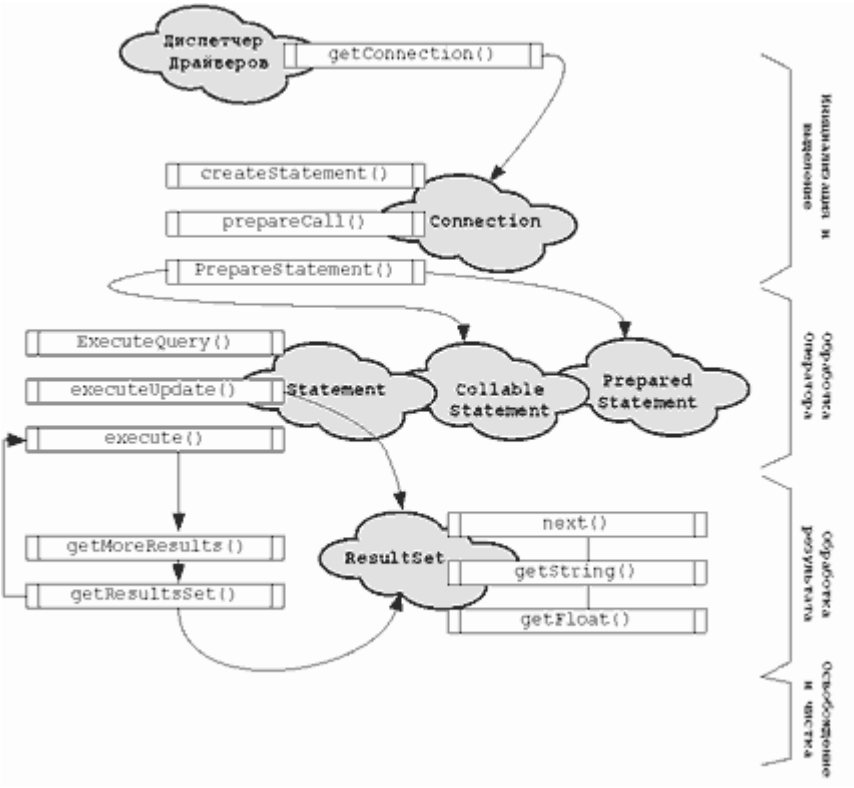


Рис.20. Иерархия классов JDBC и поток API JDBC

Чтобы обработать информацию из базы данных, информационно-обучающая система на языке Java выполняет ряд шагов. На рис.20 показаны основные объекты JDBC, методы и последовательность выполнения, Во-первых, программа вызывает метод *getConnection ()*, чтобы получить объект *Connection.*Затем она создает объект *Statement* и подготавливает оператор SQL.

Оператор SQL может быть выполнен немедленно (объект *Statement*), а может быть откомпилирован (объект *PreparedStatement*) или представлен в виде вызова процедуры (объект *CallableStatement*). Когда выполняется метод *executeQuery()*, возвращается объект *ResultSet*. Операторы SQL, такие как update или delete не возвращают *ResultSet*. Для таких операторов используется метод *executeUpdate()*. Он возвращает целое, указывающее количество рядов, затронутых оператором SQL.

ResultSet содержит ряды данных и анализируется методом *next()*. Если приложение обрабатывает транзакции, можно пользоваться методами *rollback()* и *commit()* для отмены или подтверждения изменений, внесенных оператором

SQL.

Примеры запроса и модификации базы данных с использованием JDBC

Данный пример иллюстрирует как при помощи SQL - опрератора SELECT составляется список всех студентов из базы данных. Ниже приводятся шаги, которые необходимы для выполнения этого задания при помощи API JDBC. Каждый шаг имеет форму текста на языке Java с комментариями.

```
// описать методы и переменные
public void ListStudents () throws SQLException
{
int i,  noOfColumns;
String stNo, stFName, stLName;
// инициализировать и загрузить драйвер JDBC-ODBC
Class.forName ("jdbc.odbc.JdbcOdbcDriver");
// создать объект Connection
Connection ex1Con = DriverManager.getConnection (
"jdbc:odbc:StudentDB;uid="admin";pw="sa"");
// создать простой объект Statement
Statement ex1Stmt = ex1Con.createStatement ();
// Создать строку SQL, передать ее СУБД и
// выполнить SQL-оператор
ResultSet ex1rs = ex1Stmt.executeQuery (
"SELECT StudentNumber, FirstName, LastName FROM Students");
// Обработать каждый ряд и вывести результат на консоль
System.out.println ("Student Number  First Name  Last Name");
while (ex1rs.next())
{
stNo = ex1rs.getString (1);
stFName = ex1rs.getString (2);
stLName = ex1rs.getString (3);
System.out.println (stNo, stFName, stLName);
}
}
```

В следующем примере поле firstName таблицы Students изменяется. Доступ осуществляется через поле StudentNumber.

// описать методы, переменные и параметры

```
public void UpdateStudentName (String stFName, String stLName, String
stNo)


throws SQLException
{
int retValue;
//инициализировать и загрузить драйвер JDBC-ODBC
Class.forName ("jdbc.odbc.JdbcOdbcDriver");
// создать объект Connection
Connection ex1Con = DriverManager.getConnection (
"jdbc:odbc:StudentDB;uid="admin";pw="sa"");
// создать простой объект Statement
Statement ex1Stmt = ex1Con.createStatement ();
// Создать строку SQL, передать ее СУБД и
// выполнить SQL-оператор
String SQLBuffer = "UPDATE Students SET FirstName =" +
stFName + ", lastName =" + stLName +
"WHERE StudentNumber = " + stNo;
retValue = ex1Stmt.executeUpdate (SQLBuffer);
System.out.println ("Модифицированно " + retValue +
" строк в базе данных.")
}
```

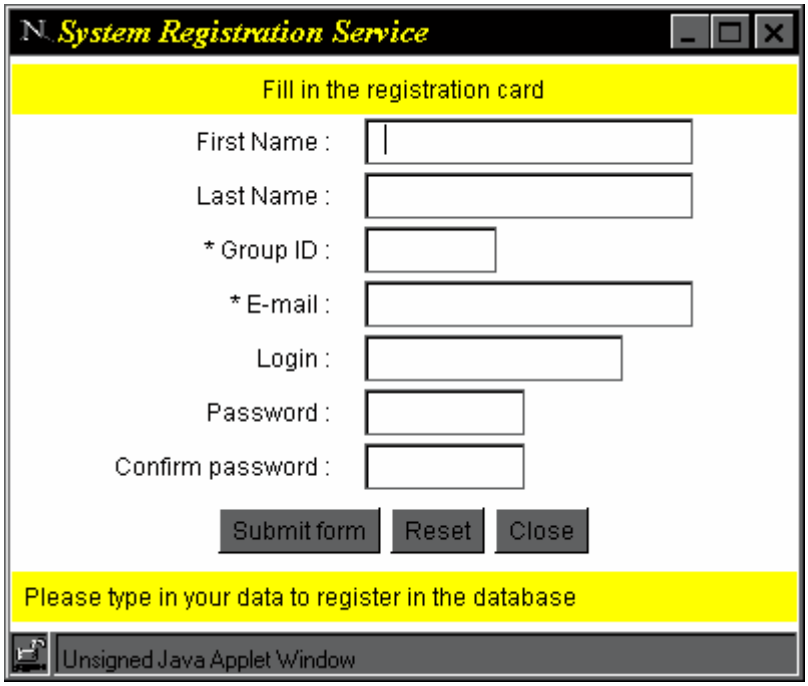


Рис.21. Интерфейс для регистрации пользователя в АИС.

Таким образом, взаимодействие с базами данных из Java также отличается простотой и гибкостью, связанной с эффективной реализацией JDBC API. В сочетании со своей природной платформо-независимостью, Java предоставляет уникальный инструмент для создания интерактивных распределенных информационно-обучающих систем на база Internet/Intranet - технологий.

Основными сложностями при реализации корпоративных систем на базе данной архитектуры являются:

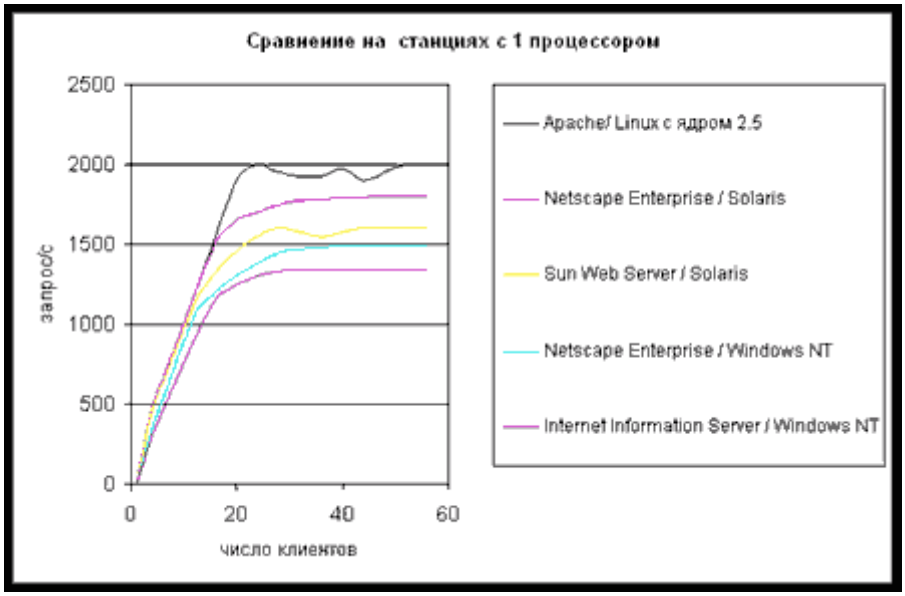
- отсутствие многих популярных приложений и средств разработки реализованных в виде JAVA аплетов;
- относительное высокое время компиляции аплетов на клиентских местах (временно);

- вопросы безопасной работы в сети.

2.2.1. Сравнительные исследования типовых серверных платформ.

Выбирая платформу для АИС, нужно учитывать множество аспектов. На решение влияют соображения, связанные с надежностью (кластеризация и балансировка нагрузки), среды разработки, работы над содержанием узла и защиты информации. Результаты тестирования различных платформ широко представлены в периодической печати, представим здесь лишь некоторые обобщения материалов тестирования [].

При проведении испытаний оценивались **Solaris 2.6, Windows NT Server 4** и **Red Hat Linux 6.02** (ядро **2.2.11**) при эксплуатации четырех **web**-серверов, занимающих ведущие позиции в мире: **Microsoft Internet Information Server 4 (IIS), Netscape Enterprise Server 3.61, Web Server 2.1** корпорации **Sun** и **Stronghold Web Server 2.4.1** (популярный вариант **Web**-сервера **Apache** с функциями защиты от несанкционированного доступа) (на тестах использовались триал версии указанного программного обеспечения). Все платформы были испытаны с помощью новой версии эталонного теста **WebBench** отделения **Ziff-Davis Benchmark Operation.**



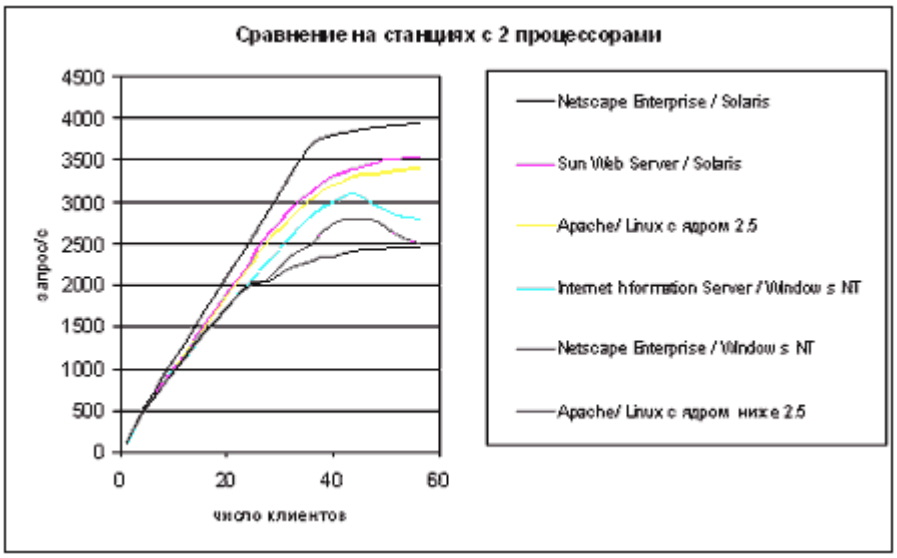


Рис.22. Сравнительные характеристики различных платформ.

2.2.1.2. Особенности функционирования АИС на платформе Sun.

Solaris - это современная операционная система UNIX клона. Примечательно, что она, опережая свое время, позволяет работать с 64-разрядными прикладными программами и имеет собственные расширения, которые помогают ей выдерживать высокие пользовательские нагрузки Web-узлов с интенсивным обменом информацией. В Solaris также предусмотрены замечательные возможности применения серверных прикладных программ и средств разработки сторонних производителей.

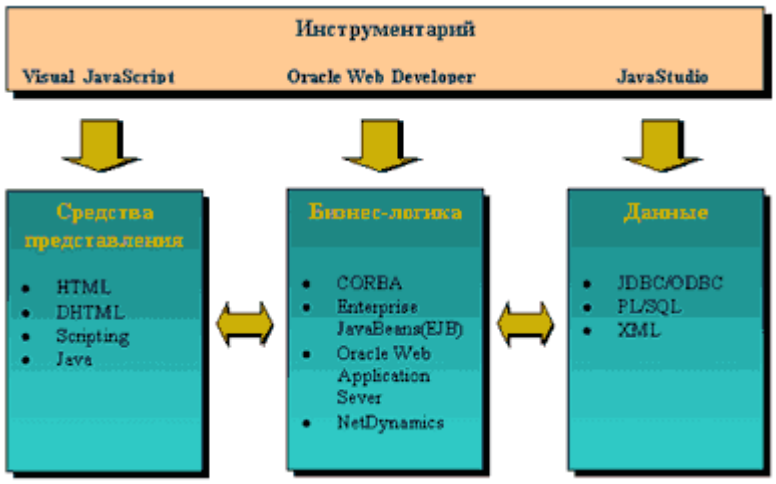


Рис.23. Функциональная схема информационной системы на базе Solaris

Ключевой момент для понимания различий между платформами Linux, Microsoft и Sun - способ, которым серверные программы каждой из них обрабатывают большое число подключений. Обычно это делается в многопотоковом режиме. Многопотоковый режим возникает, когда прикладная программа (также называемая процессом) содержит множество небольших блоков исполняемого кода, работающих независимо друг от друга и, возможно, одновременно на разных процессорах. Эти потоки могут совместно пользоваться ресурсами и представляют собой способ организации программы, позволяющий одновременно выполнять несколько задач.

Модель потоков Solaris весьма сложна. Она состоит из потоков на уровне ядра (kthreads) - реальных объектов, передаваемых отдельному процессору; потоков на пользовательском уровне и промежуточной структуры, называемой облегченным (lightweight) процессом. Это позволяет тонко управлять структурой прикладной программы и реализации в ней прикладной многозадачности.

Stronghhold на платформе Solaris

Создатели Web-сервера Stronghold (и Apache, основы Stronghold) считают, что многопотоковые программы обычно менее надежны, чем "монолитные". Такое различие стратегий объясняет значительные расхождения показателей производительности, поскольку и Sun Web Server 2.1, и Netscape Enterprise используют второй процессор, установленный в испытательных системах. Поэтому Stronghold, в зависимости от прикладного ПО, не столь эффективно использует оборудование Sun, содержащее до 64 процессоров.

Netscape на платформе Solaris

Netscape Enterprise Server 3.61 - Web-сервер, избранный для реализации большинства крупных узлов на основе Solaris, в том числе и корпорации Sun. Инструментальные средства фирмы Netscape, а также предлагаемые независимыми производителями, способствуют разработке сложных прикладных программ для Web с помощью сценариев на языках JavaScript, CORBA, Java.

Еще одна важнейшая система, стоящая за добротными программами для Web на серверах Netscape, - сервер прикладных программ Netscape Application Server (NAS). Сервер NAS - среда программирования для объектов на языках C++ и Java - обеспечивает масштабируемость и устойчивость к сбоям прикладных программ. В NAS имеются инструменты для создания многоуровневых программ, объединяющих HTML и запросы к базам данных на серверах NAS.

Sun Web Server

Sun Web Server (SWS) обеспечивает разработку программ, конечно же, на языке Java. На SWS можно использовать сервлеты и разнообразные возможности, такие как CORBA. Сервлеты (servlet) - это Java-программы, запускаемые на сервере и, подобно CGI, передающие сверстанные HTML-страницы браузеру. Для сервлет существует собственный API к функциям рабочей среды сервера. В SWS также предусмотрена возможность использования серверных Java-страниц (Java Server Pages) - способа обращения к серверным функциям Java со страниц Web и из CGI-программ.

При соответствующем использовании Web-серверов на платформе Solaris, эта операционная система на многопроцессорных станциях превосходит по производительности Windows NT. Такого результата достигла Sun Microsystems благодаря использованию Solaris Network Cache and Accelerator (SNCA) - мощного механизма кэширования для Web-сервера. SWS победил в испытаниях при обслуживании статических страниц. При выполнении динамических CGI-испытаний Netscape на платформе Solaris превзошел и SWS, и IIS для Windows NT.

2.2.1.3. Особенности функционирования АИС на платформе Microsoft.

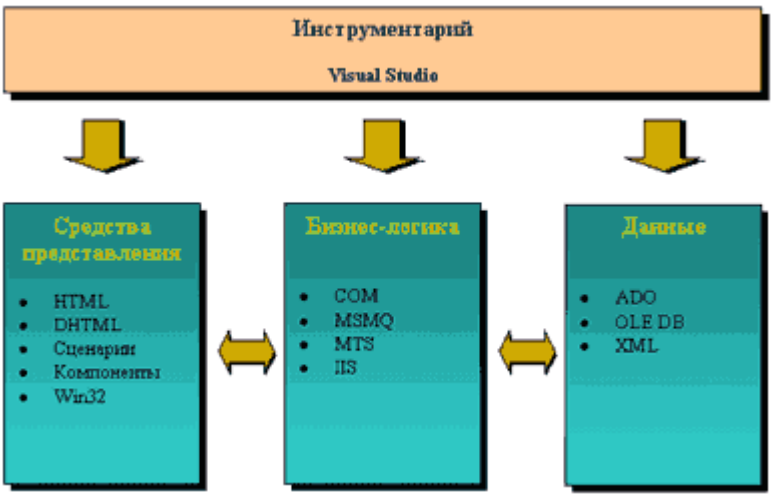


Рис.24. Функциональная схема информационной системы на платформе Windows NT.

Microsoft Windows NT Server

Windows NT 4 Server и Internet Information Server (IIS) являются исключительно коммерческой web-платформой, разработанной компанией Microsoft. Данная ОС имеет удобный интуитивно понятный интерфейс взаимодействия с пользователем, что делает её довольно привлекательной для использования. Windows NT 4 Server оснащена службой балансировки нагрузки (Windows NT Load Balancing Services), которая позволяет создавать группу серверов и распределять нагрузку между ними. Пользователи при этом видят только один IP-адрес и полагают, что существует только один сервер. Однако служба Load Balancing Services - это неполноценная кластерная система, поэтому она не способна обеспечить такое высокое быстродействие, как настоящий кластер. Windows NT не может работать с мощными аппаратными и программными средствами кластеров, в том числе с собственной службой Microsoft Cluster Service, продуктами серии Infinity компании IBM и продуктами NonStop производства Compaq. У Microsoft есть продукты всех этапов разработки для Web, однако обычно их заменяют изделиями других фирм. Пакет Allaire ColdFusion 4.0, как среда разработки для Web, - отличный пример этого.

Netscape Enterprise на платформе Windows NT

Netscape Enterprise в среде Windows NT представляет собой Web-сервер, ориентированный на большие нагрузки. Для него имеется множество моделей программирования. Например, помимо общепринятых моделей разработки HTML и CGI в продукте Netscape предусмотрены возможности работы с JavaScript на стороне сервера. Почти все функции сервера Netscape для Solaris работают и на платформе Windows NT.

При тестировании на производительность IIS показал неплохие результаты. Скорость при работе IIS достигнута за счет хорошо организованной обработкой файлового ввода-вывода. Дополняет обработку сообщений в Windows NT возможность асинхронного ввода-вывода, позволяющая обрабатывать запрос одновременно с выполнением операций ввода-вывода в файл или ЛВС. Подобная функция имеется в Solaris, но до сих пор не полностью реализована в Linux. По результатам теста IIS проигрывает SWW при обработке статических страниц, а Netscape Enterprise на платформе NT оказался менее производительным во всех режимах, чем на платформе Solaris.

2.2.1.4. Особенности функционирования АИС на основе Linux.

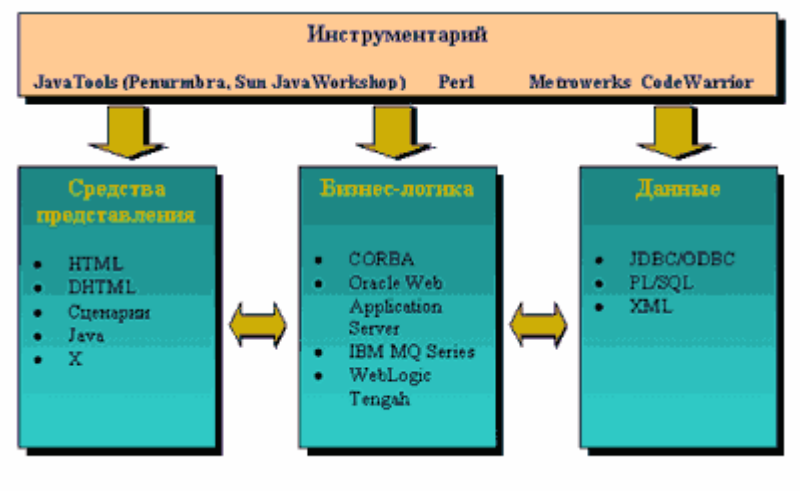


Рис.25. Функциональная схема информационной системы на платформе Linux.

Все больше растет популярность Linux и её респектабельность как платформы разработки для Web и корпоративных сред. Linux характеризуется рядом преимуществ, таких как широкое сообщество разработчиков открытого кода, поддержка многих моделей комплектующих, и, главная особенность состоит в том, что Linux полностью бесплатная ОС. Linux является разновидностью Unix и изначально создавалась для работы в сетях. В каждой новой версии Linux появляются некоторые усовершенствования, направленные на повышение масштабируемости и производительности серверных прикладных программ.

Apache *и* Stronghold

Для тестов в среде Linux был использован Stronghold Web Server 2.4.1 компании C2Net. Stronghold - это сервер с возможностями применения технологии SSL, в основе которого лежит Web-сервер Apache. Сервер Stronghold обладает всеми преимуществами Apache, в том числе мощными средствами обеспечения работы с виртуальными базовыми машинами (способность одного web-сервера обслуживать несколько машин одновременно).

Платформа Stronghold, подобно Linux, имеет заслуженную репутацию надежной и стабильной системы. Но Stronghold - и, следовательно, Apache - не оптимизированы для многопроцессорных сред. Поэтому Web-узлы, основанные на серверах Apache, лучше масштабировать путем добавления серверов, а не процессоров.

Напротив, IIS и Netscape Enterprise имеют многопотоковую архитектуру, которая масштабируется на несколько процессоров одного сервера. При испытаниях на многопроцессорных станциях они, как правило, обгоняли Stronghold.

Apache позволяет тонко настраивать ряд параметров (такие как число процессов, доступных клиентам). Для Apache, как и для других серверов, есть механизм работы сервлетам (Apache Jserv). Механизм работы с сервлетами встраивается в Apache в виде модуля и работает с любой совместимой с JDK 1.1 виртуальной Java-машиной. По производительности дуэт Apache-Linux оказался оптимальным для однопроцессорных систем. По обработке статических страниц он немного уступал SWW и IIS, а по стабильности работы превосходил серверы на платформеWindows NT.

Linux - это функциональность UNIX + пользовательско-ориентированный интерфейс Windows-систем. Большая часть поддерживаемого Linux оборудования - это то, что пользователи реально у себя имеют. Как в результате оказалось - большая часть популярной периферии для 80386/80486 поддерживается (действительно, Linux поддерживает оборудование, которое в ряде случаев не поддерживают некоторые коммерческие UNIX). Хотя некоторые достаточно экзотические устройства пока не поддерживаются.

Важным вопросом при создании АИС является обеспечение жизнестойкости и надежности работы информационных серверов. В качестве иллюстрации эффективности платформы приведем расчет параметров для сервера с 25000 посетителей в день []. Подсчет загрузки: 24ч*60мин*60 сек=86400 секунд в сутках, если каждый посетитель берет с сервера по 10 документов (*.html + графика) то при равномерном распределении загрузки получается 3 обращения к серверу в секунду. Реальное распределение трафика носит характер кривой Гаусса либо синусоиды (в зависимости от содержания сервера), в максимумах которых загрузка достигает 10-20 обр/с. Для нормальной работы такому серверу необходимо около 400 Mb RAM, при хорошей настройке - не менее 200.

При правильной конфигурации сервера не рекомендуется использовать swap. все должно помещаться в оперативной памяти (имеется ввиду, что у сервера swap-область быть должна, но она обязана быть пустой). Для предотвращения перегрузок рекомендуется пользоваться несколькими правилами, снижающими загрузку сервера. Придерживаясь этих правил можно съэкономить около 30% ресурсов сервера.

1. Для статической информации всегда ставить last-modified в атрибут выдачу CGI-скриптов - документ без временного штампа не сохраняется в локальном кэше, и постоянно перезаписывается при просмотре.
2. CGI программы хранить в любом каталоге кроме /CGI-BIN/, т.к. proxy-серверы не кэшируют файлы, находящиеся в этих каталогах, и каждый раз вынуждены обращаться к вам на сервер.
3. Устанавливать поле last-modified у русского apache с автоматическим определением кодировки, чтобы на proxy-серверах не оставались файлы в некорректной кодировке.
4. Не применять авторедиректы по чарсету в русском apache.
5. Не использовать фреймы, т.к. вместо одного файла появляется минимум 3.
6. Не использовать анимированные *.gif, т.к. некоторые NN обращаются к серверу перед каждым циклом.
7. 404 код не делать cgi-скриптом, 404 код не делать "красивым" - с графическими изображениями и указаниями на прочие разделы, т.к. сошедший с ума робот собирает невероятное количество 404 ошибок, зацикливаясь в них на веки.
8. Создать на сервере файл robot.txt, т.к. это самый запрашиваемый документ на сервере, и иначе порождает массу 404 (см. п. 7). А также разумные роботы слушаются запретов в этом файле, что уменьшает нагрузку на сервер.
9. Не ставить баннеры наверху страницы, т.к. баннер сверху отнимает 1-2 реквеста из 4-х и в итоге грузится вперед тормозя ваши сайтовые картинки.
10. При вызове баннера не обращаться каждый раз к CGI, а подставлять вместо случайного числа любое число, что можно сделать, например, получив дату на JavaScript.
11. Вызывать баннеры программами на Си, т.к. Perl работает медленнее.
12. На одной машине должен размещаться только информационный сервер, не одновременно с почтой и др. сервисами.

На сегодня архитектура Internet/Intranet, в том числе и на платформе LINUX, уже используется при построении корпоративных ИС для решения задач автоматизации управления банками, управления проектированием, управления ТП, АСУ ТП, электронной коммерции, оперативной информации по курсу валют и акций и т.п.

2.2.2 Сравнительные характеристики SQL СУБД.

Как было отмечено выше, выбор конкретной архитектуры построения информационной системы включает два основных компонента: выбор серверной платформы (выбор серверной ОС и СУБД) и выбор платформ для клиентских рабочих мест. В данном разделе более подробно остановимся на особенностях

выбора конкретной СУБД. При выборе базы данных очень важно выбрать базу данных, которая в наибольшей степени соответствуют предъявляемым к информационной системе требованиям, т.е. необходимо определиться какая модель автоматизации реализуется (автоматизация документооборота или бизнес - процессов). В первую очередь при выборе СУБД необходимо принимать во внимание следующие факторы:

1. максимальное число пользователей одновременно обращающихся к базе;
2. характеристики клиентского ПО;
3. аппаратные компоненты сервера;
4. серверную операционную систему;
5. уровень квалификации персонала.

На сегодня известно большое число различных серверов баз данных SQL. Остановимся более подробнее на следующих четырех ведущих серверных СУБД - Oracle8i, IBM DB2, Microsoft SQL Server и Informix - и сравним их в работе на каждом из основных этапов функционирования:

1. конфигурирование системы,
2. мониторинг,
3. настройка,
4. обработка запросов,
5. разработка серверных и клиентских модулей.

Данный анализ проведем с учетом того, что число клиентских мест составляет от 50 до 500, а управление СУБД должно быть максимально эффективно. Исследования проводились на серверной платформе на базе Pentium II с 128 Мбайт ОЗУ, укомплектованном 13-Гбайт диском с интерфейсом EIDE в конфигурации RAID уровня 0 (конечно лучше было бы использовать HDD с интерфейсом SCSI). Управление системами было возложено на ОС Windows NT Server 4.0. и Linux.

Oracle8i.

Пакет Oracle8i, наделенный самым развитым набором функций для работы с языком Java и доступа к данным через Интернет, системой оптимизации одновременного доступа. Единственным недостатком данной СУБД является сложность администрирования, однако все затраты на ее внедрение и освоение в последствии окупятся эффективной и надежной работой. В нашей стране на протяжении уже многих лет целым рядом специалистов культивируется негативное отношение к СУБД Oracle, как к дорогой и сложной СУБД. Оба эти тезиса являются спорными. Во-первых, уровень сложности понятие относительное. При использовании СУБД Oracle на платформе NT, она потребует практически тех же усилий, что и при использовании MS SQL. В случае же работы на UNIX-платформе, можно с уверенностью отметить, что для профессиональных юниксоидов среда Oracle является простой, понятной и доступной. Что касается дороговизны, то и тут наметились положительные сдвиги. Кроме того, что компания Oracle предлагает ряд различных масштабируемых решений в зависимости от числа обслуживаемых клиентов, она также следуя общемировым тенденциям разработала версию своей популярнейшей СУБД под LINUX и выложила ее на своем WEB сервере (www.oracle.com) для свободного использования. Среди основных свойств СУБД Oracle следует отметить такие, как:

1. Высочайшая надежность.

2. Возможность разбиения крупных баз данных на разделы (large-database partition), что дает возможность эффективно управлять гигантскими гигабайтными базами;
3. Наличие универсальных средств защиты информации;
4. Эффективные методы максимального повышения скорости обработки запросов;
5. Индексация по битовому отображению;
6. Свободные таблицы (в других СУБД все таблицы заполняются сразу при создании);
7. Распараллеливание операций в запросе.
8. Наличие широкого спектра средств разработки, мониторинга и администрирования.
9. Ориентация на интернет технологии.

Решения, не уступающие разработкам Oracle можно найти только в DB2 фирмы IBM. Ориентация на интернет технологии - основной девиз современных продуктов Oracle. В этой связи можно отметить пакеты interMedia, обеспечивающее обработку данных в мультимедийных форматах, и Jserver, встроенное средство для работы с языком Java, которое объединяет возможности языка Java с возможностями реляционных баз данных (возможность составлять на языке Java не только внутренние программы для баз данных (хранимые процедуры и триггеры), но и разрабатывать компоненты Enterprise JavaBeans и даже запустить их на сервере). Компоненты Enterprise JavaBeans представляют собой базовые модули из которых складываются Интернет-приложения на языке Java.

Фирма Oracle придерживается принципа, что всеми важными функциями необходимо управлять из единого центра, поэтому предлагаемый модуль interMedia предоставляет в распоряжение пользователей самые передовые возможности для работы с мультимедийными объектами:

1. Очень развитые средства для обработки аудио клипов;
2. Неподвижных изображений;
3. Видеофрагментов;
4. Географических данных (с целым набором функций связанных с определением местонахождения входящих в состав модуля Locator).

В Oracle8i реализуются лучшие на сегодняшний день средства для объектно-ориентированного конструирования баз данных, в том числе табличные структуры, допускающие наследование свойств и методов других табличных объектов БД, что позволят избежать ошибок при построении БД и облегчает их обслуживание.

Также необходимо отметить, что разработанная фирмой Oracle система оптимизации одновременного доступа (multiversioning concurrency) является одной из важнейших характеристик архитектуры Oracle (подобная функция есть лишь в СУБД InterBase компании InterBase компании Inprise). Данная функция позволяет исключить ситуацию, когда одному пользователю приходится ждать, пока другой завершит изменения в содержимое баз данных (т.е. в Oracle отсутствуют блокировки на чтение). Эта функция позволяет СУБД Oracle8i выполнять за секунду больше транзакций в расчете на одного пользователя, чем любая другая база данных. По уровню производительности при работе в WEB среде под LINUX Oracle занимает почетное второе место после СУБД MySQL, при этом значительно превосходя все другие СУБД по надежности и безопасности.

СУБД Microsoft SQL Server

Важнейшие характеристики данной СУБД - это:

1. простота администрирования,
2. возможность подключения к Web,
3. быстродействие и функциональные возможности механизма сервера СУБД,
4. наличие средств удаленного доступа,

В комплект средств административного управления данной СУБД входит целый набор специальных мастеров и средств автоматической настройки параметров конфигурации. Также данная БД оснащена замечательными средствами тиражирования, позволяющими синхронизировать данные ПК с информацией БД и наоборот. Входящий в комплект поставки сервер OLAP дает возможность сохранять и анализировать все имеющиеся у пользователя данные. В принципе данная СУБД представляет собой современную полнофункциональную база данных, которая идеально подходит для малых и средних организаций. Необходимо заметить, что SQL Server уступает другим рассматриваемым СУБД по двум важным показателям: программируемость и средства работы. При разработке клиентских БД приложений на основе языков Java, HTML часто возникает проблема недостаточности программных средств SQL Server и пользоваться этой СУБД будет труднее, чем системами DB2, Informix, Oracle или Sybase. Общемировой тенденцией в XXI веке стал практически повсеместный переход на платформу LINUX, а SQL Server функционирует только в среде Windows. Поэтому использование SQL Server целесообразно, по нашему мнению, только если для доступа к содержимому БД используется исключительно стандарт ODBC, в противном случае лучше использовать другие СУБД.

IBM DB2

СУБД IBM DB2 - результат почти 30-х опытно-конструкторских и исследовательских работ фирмы IBM. Последнюю на сегодня версию данной СУБД (6.x) отличает один из наиболее продуманных наборов средств управления и оптимизации и механизм БД, допускающий наращивание от портативного ПК с Windows 95 до целого кластера больших ЭВМ S/390, работающих под управлением OS/390.

Пакет DB2 выпускается в двух редакциях: DB2 Workgroup и DB2 Enterprise Edition. В данной СУБД реализованы все известные по предшествующим версиям DB2 новаторские технологии механизма БД, такие, как распараллеливание обработки запроса, полный набор средств тиражирования, сводные таблицы запросов для повышения производительности БД, возможности объектно-ориентированного конструирования баз данных и средства языка Java. К этому надо добавить, что система DB2 оснащена полым набором мультимедиа-расширений, позволяющих сохранять текст, звук и видео- фрагменты, изображения и географические данные и манипулировать ими. Можно говорить, что по возможностям масштабирования разработанная специалистами IBM технология кластеризации баз данных не имеет аналогов. Эти расширения существенно облегчают процесс разработки приложений для Web, а так же программ, содержащих фотоизображения и объемные текстовые отчеты. Система DB2 вполне конкурентоспособна и в качестве платформы для разработки приложжений т.к существует средство Stored Procedure Builder - автоматически преобразовывающее оператор SQL в соответствующий класс Java и включающее его в структуру базы данных. В версии DB2 6.1 значительно улучшена функциональная совместимость с другими СУБД: пакет позволяет

использовать разработанную Microsoft спецификацию OLE DB, новый стандарт доступа к базам данных. Средства административного управления СУБД DB2, которые в новой версии переписаны на Java и могут быть получены из Web, заслуживают самой высокой оценки.

Основными недостатками данной СУБД является относительная сложность администрирования и отсутствие (пока) реализаций под популярные серверные ОС, например LINUX.

В данной СУБД благодаря Index Smart-Guide возможно осуществлять настройку, формируя оптимальные индексы для заданного числа обращений, характеризующего типичную нагрузку на БД. DB2- единственный пакет позволяющий генерировать сводные таблицы, что значительно эффективность работы СУБД в качестве хранилищ данных. Сводная таблица - это временная рабочая область, используемая базой данных для хранения ответов на часто поступающие запросы. Ну что ж, можно сказать, что оснащенная новыми функциональными возможностями, а также средствами распараллеливания и возможностями выбора практически любого типа соединения и индексов (кроме разве что растровых индексов), модель DB2 6.1 превращается в самую недорогую из высокопроизводительных систем. Средства административного управления этой СУБД вполне соответствуют уровню решаемых задач, кроме того, она предоставляет исключительно широкие возможности для работы с мультимедиа-данными и для программирования (чего явно недостает системе Microsoft SQL Server).

СУБД от Informix.

В последнее время наметился переход от реляционных СУБД к объектно-ориентированным (что явно прослеживается на примере Oracle). Informix также следуя данной концепции анонсировала новое решение СУБД Centaur базирующуюся на реляционной БД Informix Dynamic Server 7.3 и объектно-реляционной БД Informix Universal Data Option и сочетающую в себе высокое быстродействие Dynamic Server при работе с данными с универсальностью и мультимедиа функциями Universal Data Option. Данная реализация предназначена для разработки интернет систем. Предположительно данная СУБД будет обладать гибкой средой разработки, обладающей наращиваемостью, соответствующей характерным для Интернета интенсивным нагрузкам, и средствами работы с новыми типами данных, которые с развитием Web стали использоваться повсеместно. Реализованные в новой системе средства Java позволят разработчикам создавать на этом языке хранимые процедуры, пользовательские программы и компоненты DataBlades, которые в Informix называют заказными расширениями базы данных.

С точки зрения клиентов Inforix, это станет большим шагом вперед, поскольку до настоящего времени при работе с DataBlades они могли пользоваться только языком Си и SPL, внутренним языком фирмы Informix для написания хранимых процедур. Кроме того, пакет Centaur будет оснащен встроенными средствами обработки объектов ActiveX. Это даст возможность, к примеру, создавать хранимые процедуры БД на языке Visual Basic; правда, для этого нужно, чтобы пакет Centaur выполнялся в среде Windows NT.

Centaur будет представлять собой надстройку Informix Dynamic Server и работать с традиционным для этого пакета форматом БД, так что в распоряжении пользователей останутся все прежние функции, а модернизация системы до уровня новой версии не будет сопряжена с большими сложностями. Кроме того, в пакете Centaur будут сохранены все возможности

конструирования и программирования, благодаря которым система Informix Universal Server признана выдающимся техническим достижением. Новая система будет оснащена средствами объектно-ориентированного конструирования баз данных, создания специализированных таблиц и программ индексирования; в ее состав войдет позволит пользователям встраивать в запросы собственные функции и не полагаться исключительно на стандартные средства SQL.

Выводы.

Рассмотрев основные характеристики архитектур построения АИС, серверных операционных систем и СУБД в дальнейшем в качестве архитектуры АИС мы выберем архитектуру интернет/интранет, в качестве серверной ОС Linux, в качестве СУБД Oracle 8i. В сводной таблице представлены сравнительные характеристики двух наиболее распространенных на сегодня решений на базе Microsoft SQL Server 7.0 (на NT) и Oracle8i (на Unix, Linux).

	Microsoft SQL Server 7.0	Oracle8i
Административное управление	Хорошо	Отлично
Графические инструменты	Отлично	Хорошо
Простота обслуживания	Хорошо	Отлично
Механизм данных	Хорошо	Отлично
Работа с несколькими ЦП	Приемлемо	Отлично
Функция соединения и выбор индексов	Отлично	Отлично
Одновременный доступ нескольких пользователей	Хорошо	Отлично
Обработка мультимедиа-даных	Плохо	Отлично
Подключение к Web	Плохо	Отлично
Обработка аудио, видео, изображений	Плохо	Отлично
Поиск по сему тексту	Хорошо	Отлично
Функциональная совместимость	Хорошо	Приемлемо
Сопряжение с другими БД	Хорошо	Плохо
Единая регистрация	Хорошо	Хорошо
Работа под управлением различных ОС	Приемлемо	Хорошо
Возможности программирования	Приемлемо	Отлично
Хранимые процедуры и триггеры	Хорошо	Отлично
Внутренний язык программирования	Плохо	Отлично
Построение баз данных	Хорошо	Отлично
Язык SQL	Отлично	Отлично
Объектно-ориентированные системы	Плохо	Отлично
Работа с филиалами	Отлично	Отлично
Тиражирование	Отлично	Отлично
Распределенная обработка транзакций	Отлично	Отлично

Дистанционное администрирование	Хорошо	Отлично
Организация хранилищ данных и подготовка отчетов	Отлично	Хорошо
Средства загрузки	Отлично	Отлично
Средства анализа	Отлично	Хорошо

Клиентские места при этом могут функционировать практически на любой платформе, средством доступа клиентов к СУБД является либо CGI (Perl) либо JAVA приложения. При этом к серверной части АИС предъявляются следующие требования:

2.3. Реляционная модель, как платформа для разработки современных информационных систем на примере интерактивной системы патентного обеспечения технологического проектирования.

И так мы расссмотрели различные подходы к внутренней организации баз данных. И в результате пришли к выводу о необходимости использования реляционной модели, так как она решает одну из основных проблем - внесения изменений в базу данных в процессе ее использования. Ведь в реляционной безе данных проблемы синхронизации данных не возникает вовсе, так как данные хранятся в одном экземпляре. Для большей ясности этого вопроса приведем отличия традиционных и реляционных баз данных.

Выполняемая операция	Традиционные базы данных	Реляционные базы данных
Разработка приложений	Необходимо определить, какая информация требуется различным приложениям и создать ряд общих файлов.	Необходимо определить виды хранимых данных и взаимосвязи между ними
Реализация приложений	Поступающие данные записываются в основные файлы; в каждую информационную ячейку каждого основного файла записывается один элемент данных.	Различные виды данных записываются в таблицы данных, соответствующие этим видам. В результате каждый элемент информации хранится в одном единственном месте
Модификация приложений	Требуется пересмотр структуры базы данных с последующей перезаписью основных файлов, которые затронуты вносимыми изменениями, и с переработкой всех приложений, использующих эти файлы	Достаточно найти и модифицировать таблицу, в которой должно содержаться определение нового вида данных Сами данные хранятся в других таблицах, не затрагиваемых при подобных изменениях.
Внесение частичных изменений в данные	Необходимо прочитать каждый основной файл с начала до конца, модифицируя изменяемые ячейки данных и оставляя все остальные прочитанные ячейки без изменений.	В соответствующих таблицах достаточно выделить множество строк, в которые необходимо внести изменения, и произвести эти изменения с помощью одного SQL- оператора.

Итак, основные черты реляционных баз данных:

1. Структура реляционной базы данных определяется хранящимися в них данными и не фиксируется в момент завершения разработки (т.е. является гибкой и масштабируемой).
2. Структурам данных можно давать весьма информативные названия.
3. Данные хранятся в единственном экземпляре; все опции чтения и модификации данных производятся только с этим экземпляром данных, что качественно облегчает синхронизацию данных между многими приложениями и пользователями.
4. Данные хранятся в соответствии с четко определенными и строго соблюдаемыми правилами.

2.3.1. Исследование моделей информационного представления данных в современных СУБД.

Исследование различным методов ис средств представления и управления данными в информационных системах проведем на примере интерактивной базы данных патентного обеспечения (ПО) конструкторско-технологического проектирования (КТП). Патент - это документ, свидетельствующий о праве изобретателя на его изобретение. Для стандартизации и облегчения поиска информации были введены различные классификации патентов: (Национальная Классификация Патентов (НКИ), Универсальная Десятичная Классификация (УДК), Международная Классификация Изобретений (МКИ)). Все эти классификации призваны служить инструментом для упорядоченного хранения патентных документов, что облегчает доступ к содержащейся в них информации, быть основой для избирательного распределения информации среди потребителей патентной информации и для получения систематических данных в области промышленного соответствия, что в свою очередь, определяет уровень развития различных областей техники.

Классификации патентов имеют сложную структуру, и для поиска необходимой информации может потребоваться много времени. Возможна организация поиска по всем имеющимся классификациям изобретений, но пока, в качестве примера, мы рассмотрим только Международную Классификацию Изобретений, которая являясь средством для единообразного в международном масштабе классифицирования патентных документов, представляет собой эффективный инструмент для патентных ведомств и других потребителей, осуществляющих поиск патентных документов с целью установления новизны и оценки вклада изобретения в заявленное техническое решение (включая оценку технической прогрессивности и полезности или результата).

Международная Классификация Изобретений (МКИ) имеет иерархическую структуру (представленную на рис.1) и состоит из следующих отделов: 1 - Раздел, 2 - Класс, 3 - Подкласс, 4 - Группа, 5 - Подгруппа. Иерархическая структура классификации МКИ представлена на рис.31.

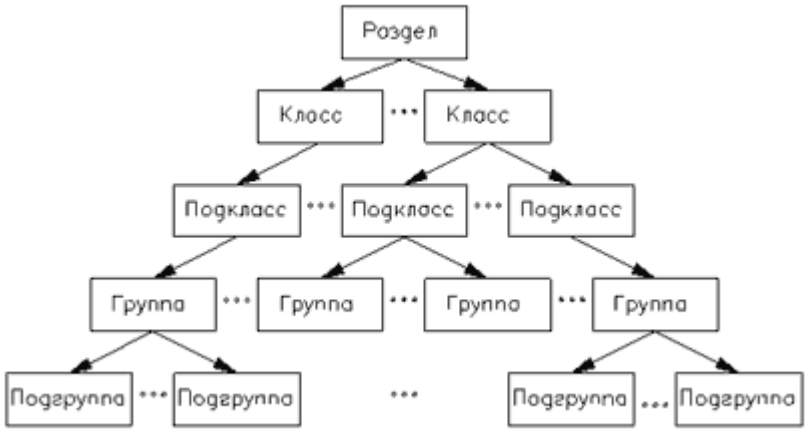


Рис. 31. Иерархическая структура классификации МКИ - как основа АИс патентного обеспечения КТП.

Логическая модель

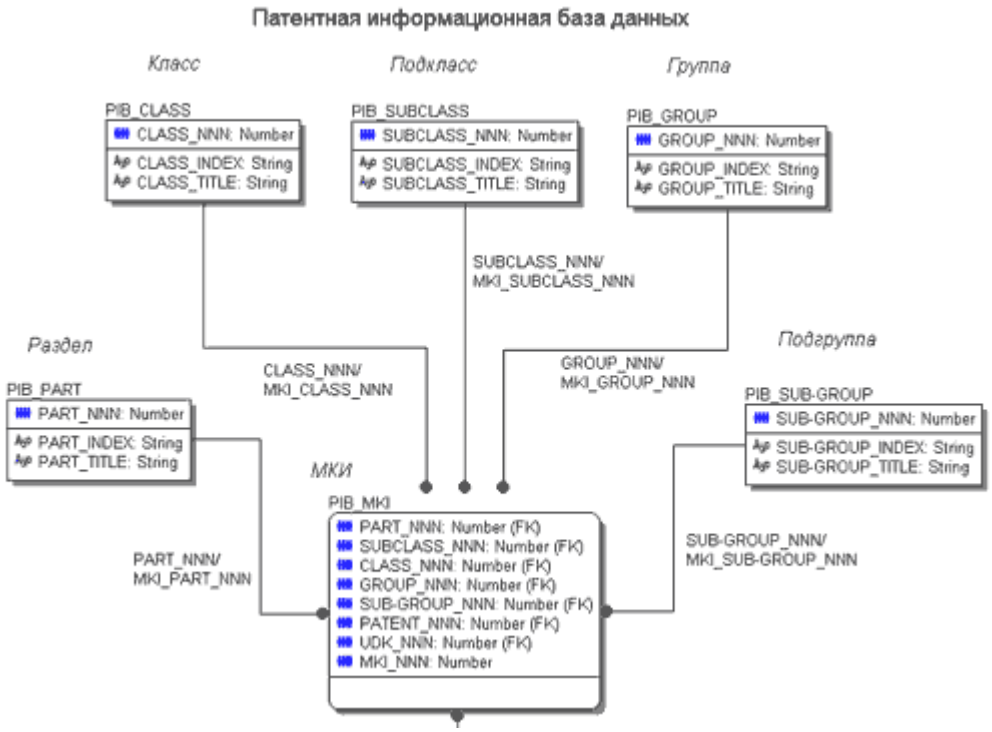
Переход от функциональной модели к логической осуществляется с помощью реляционных методов, при этом иерархическая структура функциональной модели реализуется с использованием отношений один - ко -многим и рекурсивных рис. 27. Реализацией данной логической модели является совокупностью таблиц, объединенных в единый модуль - патентную информационную базу данных (PIB - Patent Information dateBase).

Ядром логической модели является таблица PIB_MKI (МКИ), связывающая таблицы PIB_PART (Раздел), PIB_CLASS (Класс), PIB_SUBCLASS (Подкласс), PIB_GROUP (Группа), PIB_SUB-GROUP (Подгруппа) в единую структуру, определяющую реализацию Международной Классификации Изобретений (МКИ) винформационной системе патентного обеспечения технологического проектирования. Таблица PIB_MKI (МКИ) в свою очередь связана с таблицей PIB_PATENT (Патент), отвечающей за связь с таблицами PIB_GRATHDOC (Графические документы) и PIB_UDK (УДК).Таблица PIB_UDK (УДК) реализует Универсальную Десятичную Классификацию (УДК). Структура таблиц модуля PIB представлена в таблице1.

Таблица 1. Информационно-логическая Структура модуля Международной Классификации Изобретений.

Имя таблицы	Имя поля	Функц. Назначение
PIB_PART	PART_NNN	Уникальный идентификатор
	PART_INDEX	Индекс раздела в МКИ
	PART_TITLE	Название раздела
PIB_CLASS	CLASS_NNN	Уникальный идентификатор
	CLASS_INDEX	Индекс класса в МКИ
	CLASS_TITLE	Название класса
PIB_SUBCLASS	SUBCLASS_NNN	Уникальный идентификатор
	SUBCLASS_INDEX	Индекс подкласса в МКИ
	SUBCLASS_TITLE	Название подкласса

PIB_GROUP	GROUP_NNN	Уникальный идентификатор
	GROUP_INDEX	Индекс группы в МКИ
	GROUP_TITLE	Название группы
PIB_SUB-GROUP	SUB-GROUP_NNN	Уникальный идентификатор
	SUB-GROUP_INDEX	Индекс подгруппы в МКИ
	SUB-GROUP_TITLE	Название подгруппы
PIB_PATENT	PATENT_NNN	Уникальный идентификатор
	PATENT_INDEX	Патентный индекс в МКИ
	PATENT_TITLE	Название патента
	PATENT_AUTHOR	Авторы патента
	PATENT_NOTES	Примечания
PIB_UDK	UDK_NNN	Уникальный идентификатор
	UDK_INDEX	Патентный индекс в УДК
	UDK_NOTES	Примечания
PIB_GRATHDOC	GRATHDOC_NNN	Уникальный идентификатор
	GRATHDOC_FILE	Имя файла
PIB_MKI	MKI_NNN	Уникальный идентификатор



img src="oracle_pr35.gif" border=0 WIDTH=424 height=208>

Рис 32. Логическая модель

Исследование архитектур программно-технологической реализации АИС

В настоящее время существует множество архитектур, служащих для разработки информационных систем, ядром которых является СУБД. Клиент в типичной конфигурации клиент/сервер - это автоматизированное рабочее место,

использующее графический интерфейс (Graphical User Interface - GUI), обычно Microsoft Windows, Macintosh.

Сервер же, в основном, предназначен для хранения, передачи и распределения информации между клиентами. В клиент/серверной конфигурации программные средства имеют разделение на клиентскую и серверную часть, однако, частые обращения клиента к серверу снижают производительность работы сети и обуславливают сложность настройки системы.

Рассмотрим варианты распределения функций СУБД в клиент/серверной системы. СУБД выполняет три основные функции:

1. доступ к данным;
2. предоставление данных;
3. бизнес - функции.

Сервер СУБД может быть реализован на различных платформах, под управлением операционных систем UNIX, NetWare, Windows NT, OS/2 и др.

До появления технологии клиент/сервер большинство приложений функционировало на одной ЭВМ. Одна система отвечала не только за всю обработку данных, но также и за выполнение логики приложения. Кроме того, та же система обрабатывала весь обмен с каждым терминалом; все нажатия клавиш и элементы отображения обслуживались тем же процессором, который обрабатывал запросы к базе данных и логику приложения.

Oracle предоставляет такие возможности, как хранимые процедуры, поддержка ограничений целостности, функции, определяемые пользователем, триггеры базы данных и ряд других. Все это позволяет приложению хранить большое количество бизнес-правил (или семантику модели данных) на уровне базы данных. В результате приложение освобождается для выполнения более тонких задач обработки. Как показано на рис.28, такая СУБД намного более устойчива.

Программные продукты Oracle охватывают все основные компоненты архитектуры клиент/сервер, показанной на рис. 29:

1)полнофункциональный высокопроизводительный сервер RDBMS (система управления реляционной базой данных), масштабируемый от портативных ЭВМ до мэйнфреймов;

1. средства для разработки и запуска клиентских приложений, поддерживающие несколько сред GUI;
2. программный компонент для организации связи между БД на различных ЭВМ, который обеспечивает эффективную и безопасную связь с помощью широкого набора сетевых протоколов.

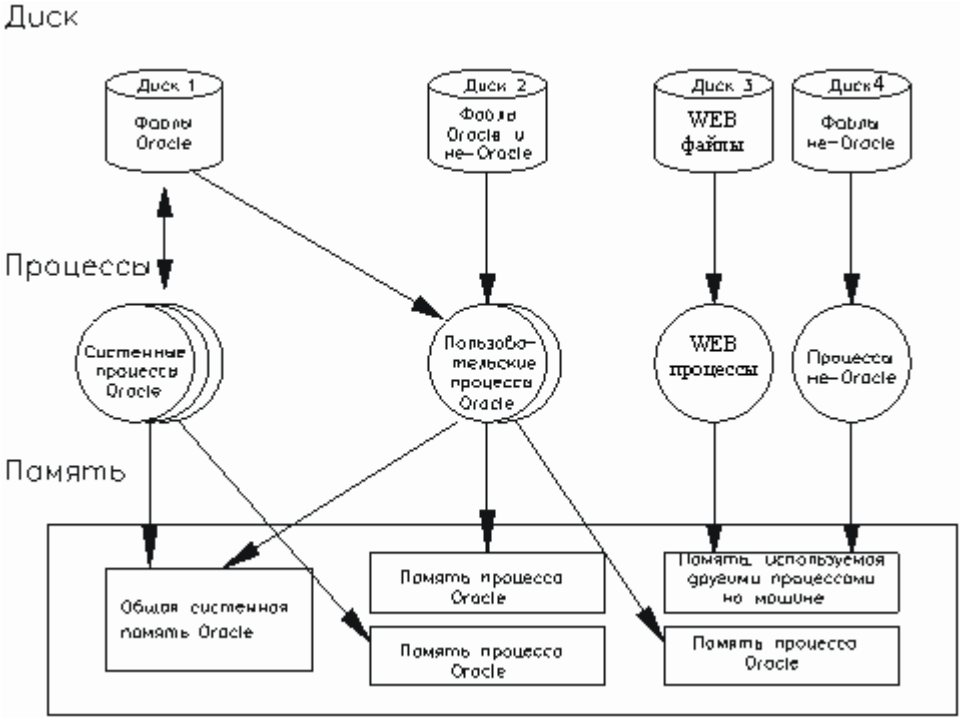


Рис. 33. Взаимодействие основных компонентов в архитектуре Oracle.

Oracle использует память системы (как реальную, так и виртуальную) для выполнения пользовательских процессов и самого программного обеспечения СУБД, и для кэширования объектов данных. В простой конфигурации Oracle файлы базы данных, структуры памяти, фоновые и пользовательские процессы располагаются на одной машине без использования сети. Однако, намного чаще встречается конфигурация, когда БД расположена на машине-сервере, а инструментальные средства Oracle - на другой машине (например, PC с Microsoft Windows). При такой клиент/серверной конфигурации машины связываются посредством некоторого сетевого программного обеспечения, которое позволяет двум машинам поддерживать связь. Для организации взаимодействия клиент/сервер или сервер-сервер необходимо использовать программный продукт Oracle SQL*Net, который позволяет СУБД Oracle взаимодействовать с сетевым протоколом. SQL*Net и поддерживает большинство сетевых протоколов для локальных вычислительных сетей (таких как TCP/IP, IPX/SPX) и для мэйнфреймов (например, SNA). По существу, SQL*Net является промежуточной программной прослойкой между Oracle и сетевым ПО, обеспечивающей связь между клиентской машиной Oracle (на которой работает, например, SQL*Plus) и сервером базы данных или между серверами баз данных. Опции SQL*Net позволяют одной машине работать с одним сетевым протоколом, сообщаясь с другой машиной, работающей с другим протоколом.

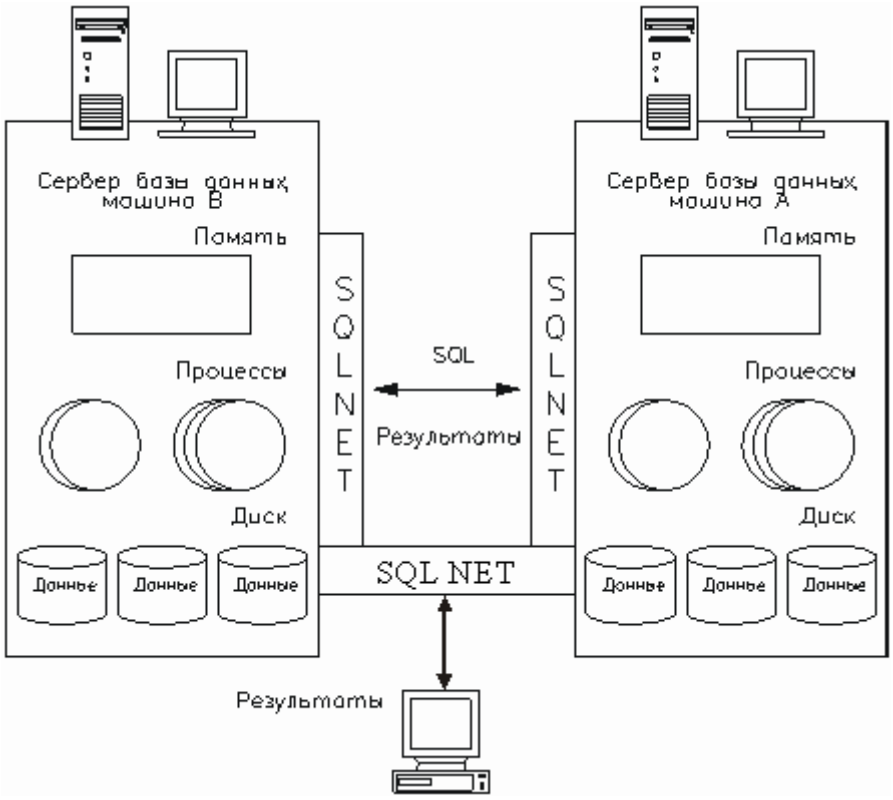


Рис. 34. SQL*NET как средство обеспечения взаимодействия между СУБД и сетью.

В зависимости от размеров, таблицы (и других объектов) всех учетных разделов пользователей могут, очевидно, размещаться в одном файле базы данных, но это - не лучшее решение, так как оно не способствует гибкости структуры базы данных для управления доступом к различным пользовательским разделам, размещения базы данных на различных дисководах или резервного копирования и восстановления части базы данных. В СУБД Oracle предусмотрены привилегии системного уровня, резервное копирование и поддержка национальных языков. Все это позволяет сделать вывод о целесообразности разработки интерактивной информационной системы патентного обеспечения технологического проектирования на основе СУБД Oracle.

2.3.2 Компоненты системы управления реляционной базой данных (RDBMS).

2.3.2.1 Ядро системы управления реляционной базой данных (RDBMS).

Две важные части архитектуры RDBMS - *ядро*, которое является программным обеспечением, и *словарь* данных, который состоит из структур данных системного уровня, используемых ядром, управляющим базой данных.

RDBMS можно рассматривать как операционную систему (или подсистему), разработанную специально для управления доступом к данным; ее *основные функции* - хранение, выборка и обеспечение безопасности данных. Подобно операционной системе, СУБД Oracle управляет доступом одновременно работающих пользователей базы данных к некоторому набору ресурсов. *Подсистемы RDBMS* очень схожи с соответствующими подсистемами ОС и сильно интегрированы с предоставляемыми базовой ОС сервисными функциями доступа

на машинном уровне к таким ресурсам, как память, центральный процессор, устройства и файловые структуры.

RDBMS поддерживают собственный список авторизованных пользователей и их привилегий; управляют кэшем памяти и страничным обменом; управляют блокировкой разделяемых ресурсов; принимают и планируют выполнение запросов пользователя; управляют использованием табличного пространства.

На рис.31. показаны основные подсистемы ядра Oracle, управляющего базой данных.

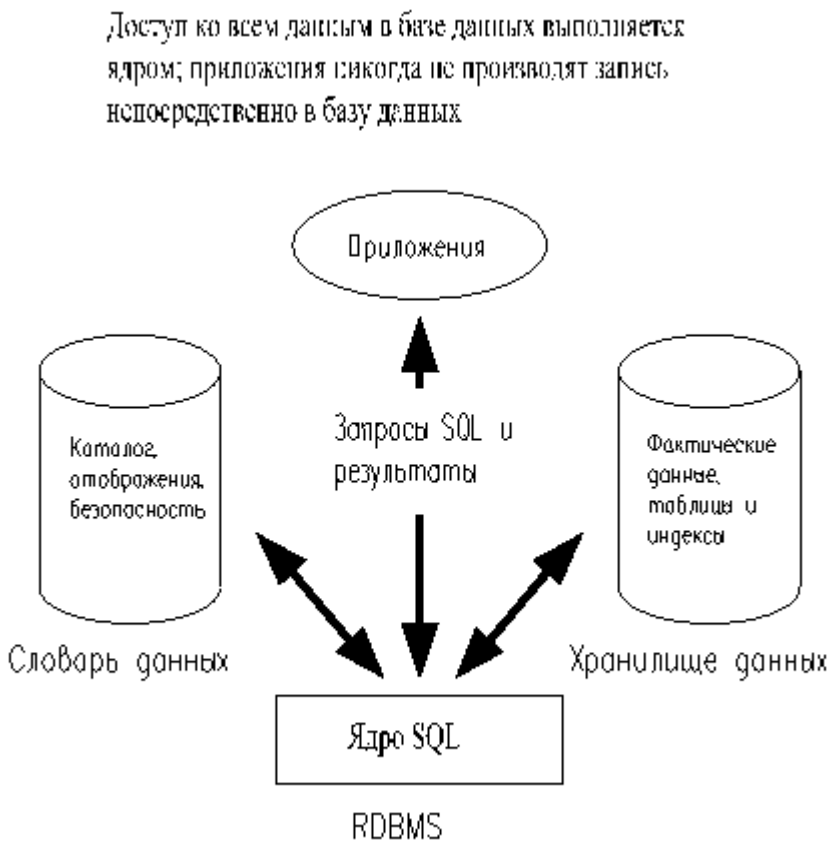


Рис.31. Структура ядра СУБД Oracle.

Итак, *база данных* - собрание данных, между которыми существуют (смысловые) связи. Физическое расположение и реализация базы данных прозрачны для прикладных программ; физическую базу данных можно перемещать и реорганизовывать и это не окажет влияния на работоспособность программ.

Физически база данных Oracle - не более чем набор файлов где-то на диске. Расположение этих файлов несущественно для функционирования (хотя важно для производительности) базы данных.

Логически база данных - это множество пользовательских разделов Oracle, каждый из которых идентифицируется именем пользователя (с паролем), уникальным в данной БД. На рис.29. показана архитектура Oracle.

Существуют три основные группы файлов на диске, составляющие базу данных.

1. Файлы базы данных

2. Управляющие файлы
3. Журнальные файлы

Наиболее важные из них - *файлы базы данных*, где располагаются собственно данные. *Управляющие и журнальные файлы* поддерживают функционирование архитектуры. Для доступа к данным БД все три набора файлов должны присутствовать, быть открытыми и доступными Oracle. Если эти файлы отсутствуют, обратиться к базе данных нельзя, и администратор базы данных должен будет восстанавливать часть или всю БД, используя файлы резервных копий (если их сделали!). Все эти файлы двоичные.

После инсталляции СУБД (об этапах установки подробно написано в [ ]) администратор имеет возможность войти в СУБД используя учетные записи SYS или SYSTEM, с парролем: master или manager, для создание учетных записей других пользовтаелей, при этом пароли учетных записей SYS и SYSTEM необходимо сразу же изменить.

Для работы с файлами базы данных на машине должны существовать системные процессы Oracle и один (или больше) пользовательский процесс.

Системные процессы Oracle (их называют фоновыми) обеспечивают функционирование *пользовательских процессов* - выполняют функции, которые иначе пришлось бы выполнять пользовательским процессам непосредственно.

Дополнительно к *фоновым процессам Oracle* в простейшем случае на одно подключение к базе данных должен существовать один пользовательский процесс. Пользователь должен подключиться к базе данных прежде чем он сможет обратиться к какому-либо объекту. Если один пользователь регистрируется в Oracle, используя SQL*Plus, другой пользователь выбирает Oracle Forms, а еще один пользователь открывает электронную таблицу Excel, значит имеется три пользовательских процесса для работы с этой базой данных - по одному для каждого подключения.

Oracle использует память системы (как реальную, так и виртуальную) для выполнения пользовательских процессов и самого программного обеспечения СУБД, и для кэширования объектов данных. Существуют две главные области памяти Oracle:

1. *разделяемая память*, которая используется всеми процессами, работающими с базой данных,
2. *локальная память* для каждого пользовательского процесса.

Системная память.

Системная память. Oracle для всей базы данных называется SGA (system global area - системная глобальная область или shared global area - разделяемая глобальная область). Данные и управляющие структуры в SGA являются разделяемыми, и все фоновые процессы Oracle и пользовательские процессы могут к ним обращаться.

Память пользовательского процесса. Для каждого подключения к базе данных Oracle выделяет PGA (process global area - глобальную область процесса или program global area - глобальную область программы) в памяти машины и, кроме того, - PGA для фоновых процессов. Эта область памяти содержит

данные и управляющую информацию одного процесса и между процессами не разделяется.

2.3.2.2 Типы обрабатываемых данных

Типы данных обрабатываемых СУБД Oracle представлены в таблице.

Таблица 2. Типы обрабатываемых данных.

Тип данных	*Описание*
CHAR(*size*)	Символьная строка фиксированной длины, имеющая максимальную длину *size* символов. Длина по умолчанию 1, максимальная -255.
CHARACTER(*size*)	То же, что и CHAR.
DATE	Правильные даты в интервале от 1 января 4712 года до н.э. до 31 декабря 4712 года.
LONG	Символьные данные переменной длины до 2 Гигабайт.
LONG RAW	Двоичные данные переменной длины вплоть до 2 Гигабайт или 231-1.
MLSLABEL	Используется в Trusted ORACLE.
NUMBER(*p,s*)	Число, имеющее *p* значащих цифр и масштаб *s*. *p* может быть от 1 до 38. *s* может принимать значения от -84 до 127.
RAW(*size*)	Двоичные данные длиной *size* байт. Максимальное значение для *size* - 2000 байт. Параметр *те для* RAW обязателен.
RAW MLSLABEL	Используется в Trusted ORACLE.
ROWID	Значения псевдостолбца ROWID.
VARCHAR2(*size*)	Символьная строка переменной длины, имеющая максимальную длину *size* символов. Длина по умолчанию 1, максимальная - 2000.
VARCHAR(*size*)	То же что и VARCHAR2.

Извлекать данные можно также и из псевдостолбцов (табл.3), которые похожи на столбцы таблиц, но их значения нельзя изменять при помощи операторов DML.

Таблица 3. Псевдостолбцы.

Название столбца	Возвращаемое значение
sequence.CURRVAL	Текущее значение sequence в данном сеансе (sequence.NEXTVAL должен быть выбран).
sequence.NEXTVAL	Следующее значение sequence в текущем сеансе.
[table.]LEVEL	1 - для корня дерева, 2 - для узлов второго уровня и так далее. Используется в операторе SELECT в иерархических запросах.
[table.]ROWID	Значение, которое идентифицируют строку в таблице table уникальным образом. Значения псевдостолбца ROWID имеют тип данных ROWID, а не NUMBER и не CHAR.

ROWNUM	Порядковый номер строки среди других строк, выбираемых запросом. ORACLE выбирает строки в произвольном порядке и приписывает значения ROWNUM, прежде чем строки будут отсортированы предложением ORDER BY.

Требования к именам объектов базы данных

- должны иметь длину от 1 до 30 бант, за исключением имен баз данных, длина которых ограничена 8 байтами;
- не могут содержать кавычек;
- не могут совпадать с именами других объектов.

Имена, которые всегда заключены в двойные кавычки, могут нарушать, приведенные ниже правила. В противном случае, имена

- должны начинаться с букв A-Z;
- могут содержать только символы A-Z, 0-9, _, $ и #;
- не могут дублировать зарезервированные слова SQL.

Различие между прописными и строчными буквами учитывается только в именах, заключенных о двойные кавычки.

Операции и их приоритеты

Арифметические операции	Символьные операции	Логические операции	Операции сравнения
+ - (один операнд)	**| |**	**NOT**	**=**
*** /**		**AND**	**!= ^= ~= <>**
+ - (два операнда)		**OR**	**> >= < <=**
			IN
			NOT IN
			ANY, SOME
			ALL

2.3.2.3 Непроцедурный доступ к данным (SQL).

Характерной чертой RDBMS является способность обработки данных как множества; файловые системы и СУБД с другими моделями обрабатывают данные способом "запись-за-записью". С RDBMS можно общаться, используя структурированный язык запросов (Structured Query Language - SQL). SQL - непроцедурный язык, который разработан специально для операций доступа к нормализованным структурам реляционных баз данных. Основное различие между SQL и традиционными языками программирования состоит в том, что операторы SQL указывают, какие операции с данными должны выполниться, а не способ их выполнения.

Список, зарезервированных слов SQL

Язык SQL включает зарезервированные слова, имеющие определенное значение в операторах SQL. Эти слова нельзя использовать в качестве имен объектов базы данных.

ACCESS*	DEFAULT*	INTEGER	OPTION*	START*
ADD*	DELETE*	INTERSECT*	OR*	SUCCESSFUL
ALL*	DESC*	INTO*	ORDER*	SYNONYM
ALTER*	DISTINCT*	IS*	PCTFREE*	SYSDATE
AND*	DROP*	LEVEL*	PRIOR*	TABLE*
ANY*	ELSE*	LIKE*	PRIVILEGES	THEN*
AS*	EXCLUSIVE	LOCK	PUBLIC*	TO*
ASC*	EXISTS*	LONG	RAW	TRIGGER
AUDIT	FILE	MAXEXTENTS	RENAME*	UID
BETWEEN*	FLOAT	MINUS*	RESOURCE*	UNION*
BY*	FOR*	MODE	REVOKE	UNIQUE*
CHAR*	FROM*	MODIFY	ROW	UPDATE*
CHECK*	GRANT*	NOAUDIT	ROWID	USER
0CLUSTER*	GROUP*	NOCOMPRESS*	ROWLABEL	VALIDATE
COLUMN	HAVING*	NOT*	ROWNUM*	VALUES*
COMMENT	IDENTIFIED*	NOWAIT	ROWS	VARCHAR*
COMPRESS*	IMMEDIATE	NULL*	SELECT*	VARCHAR2*
CONNECT*	IN*	NUMBER*	SESSION	VIEW*
CREATE*	INCREMENT	OF*	SET*	WHENEVER
CURRENT*	INDEX*	OFFLINE	SHARE	WHERE*
DATE*	INITIAL	ON*	SIZE*	WITH*
DECIMAL*	INSERT*	ONLINE	SMALLINT	

Комментарии

Комментарии, заданные ограничителями '/*' и '*/', могут стоять в любом месте оператора SQL:

```
ALTER USER petrov /* Это комментарий */ IDENTIFIED BY petr;
```

Можно использовать стандартные комментарии ANSI. Все символы после двух дефисов до конца строки игнорируются.

```
ALTER USER petrov /* Это комментарий продолжен до конца строки IDENTIFIED
BY petr;
```

Приоритеты операций

При вычислении выражения, содержащего несколько операций, ORACLE сначала выполняет операции с более высоким приоритетом. Операции, приведенные на одной и тойже строке, имеют одинаковые приоритеты.

Замечание: В выражениях можно использовать круглые скобки, чтобы изменять последовательность выполнения операций, предписываемую приоритетом. Выражения, заключенные в скобки, ORACLE вычесляет в первую очередь. Без скобок операции с одинаковым приоритетом ORACLE выполняет слева направо.

Приоритеты операций SQL

Унарные арифметические операции + - операция PRIOR

Арифметические операции * /

Бинарные арифметические операции + - символьная операция | |

Все операции сравнения

Логическая операция NOT

Логическая операция AND

Логическая операция OR

Приоритеты арифметических операций

Унарные арифметические операции + -

Арифметические операции * /

Бинарные арифметические операции + -

Встроенные операторы SQL.

Как было отмечено ранее SQL (Structured Query Language) - структурированный язык запросов, позволяет оперировать данными в реляционных базах данных. Стандарт SQL определен Американским национальным институтом стандартов и ISO в качестве международного стандарта. Целью данного издания не является полное и всеобъемлющее освещение синтаксиса SQL, для этого есть специализированные справочники и документация [1-10], мы же постараемся на нескольких простых конкретных примерах показать Вам всю элегантность и мощь SQL. Все примеры, приведенные ниже даны, применительно к ER-диаграмме ДОКТОР-ПАЦИЕНТ, приведенной на рис.10.

Что же из себя представляет SQL - программа? Чаще всего это оформленная в виде отдельного файла программная конструкция, написанная в любом текстовом редакторе с учетом требований синтаксиса языка SQL. Такая форма представления SQL программы - называется скриптом и предназначена для выполнения на сервере, например с помощью специальной терминальной

программы SQL+ (строка запуска скрипта в SQL+: @<путь>/<имя скрипта>.sql). Считается хорошим тоном наличие в скрипте комментариев. Для выделения строчных комментариев используется следующий набор символов: --.

Перед тем как перейти непосредственно к рассмотрению использования основных SQL операторов еще несколько слов об организации проектирования БД. Процесс создания БД - это сложный многоэтапный процесс, причем как правило в нем принимают участие большое число разработчиков, поэтому очень важным является правильная организация внесения изменений в БД. Для этих целей очень часто используется технология "РАЗДЕЛЕНИЯ ЗАДАЧ", которая заключается в следующем: каждый разработчик, выполняя конкретную часть создания или модификации БД, оформляет все производимые им изменения в виде скриптов (т.е. отдельных файлов), архивные версии которых перед запуском на сервере размещаются на отдельном, специально выделенном, носителе (диске сервера), причем каждое такое изменение БД оформляется в виде отдельной задачи, имеющей свой уникальный номер. Например, в задаче 000001/VER001/ (физически это просто каталог на диске) находятся все скрипты (файлы) по первоначальному созданию БД. Такой подход позволяет максимально удобно решать задачи "Контроля версий", производить миграцию созданной БД на другой сервер (достаточно на новом сервере выполнить все задачи в последовательности следования номеров задач), обеспечивает достаточно высокий уровень безопасности, возможности отката на любую предыдущую позицию (этап разработки БД) и многое другое. В дальнейшем при выполнении практических примеров приведенных в данном издании советуем вам придерживаться именно этой технологии. Если вы на каком-то из этапов допустили ошибку (которую выявили на этапе выполнения скрипта в БД) не надо исправлять текст непосредственно этого скрипта, оформьте новую версию выполняемой задачи, в которой разместите исправленный скрипт. Это позволит Вам всегда отслеживать все Ваши ошибки. Последняя рекомендация, которую хотелось бы дать, заключается в том, что если в задачу входит несколько скриптов, то целесообразно оформить один дополнительный запускающий скрипт, например start.sql, в который поместить запуск всех остальных скриптов. Поверьте на слово - это значительно съэкономит ваше время в дальнейшем. Например, если в задачу 000001/VER001/ входят файлы: db1.sql., db2.sql, db3.sql, то файл start.sql может быть представлен следующим способом:

```
************************ start.sql ******************************
Spool 000001.log
@db1.sql
@db2.sql
@db3.sql
spool off
*************************************************************
```

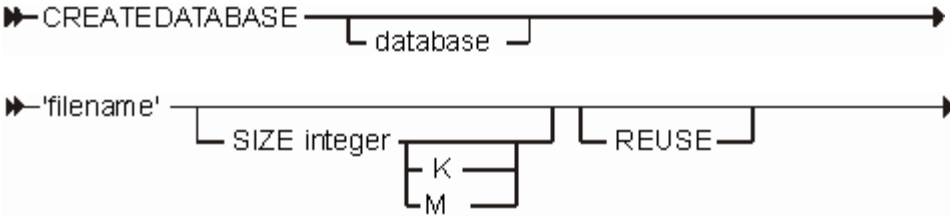
При этом необходимо помнить, что все файлы должны быть в кодировке той среды, из которой вы собираетесь запускать SQL+ (KOI8, Win1251, DOS).

ОПЕРАТОРЫ СОЗДАНИЯ ОБЪЕКТОВ БД.

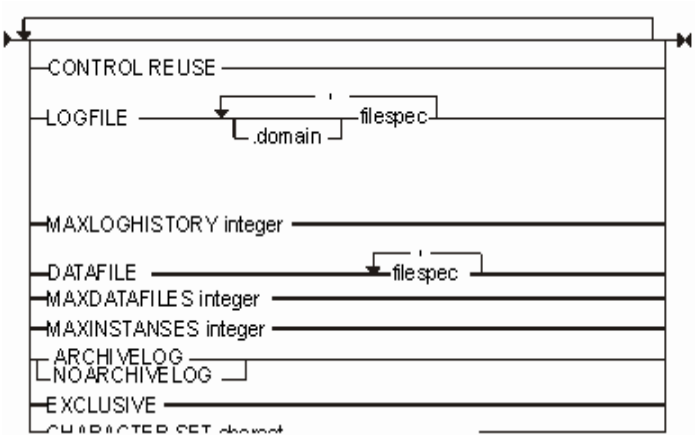
Перед тем как работать с данными с БД ее надо создать, для этих целей используется специальная группа операторов, предназначенных для создания объектов базы данных, все операторы данной группы начинаются с ключевого слова CREATE.

CREATE DATABASE

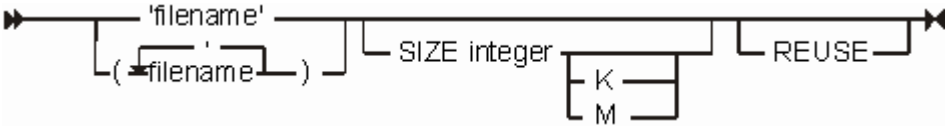
Создает базу данных. Задает и определяет максимальное число экземпляров файлов данных и журнальных файлов, устанавливает режим архивирования.



Filespec (файлы данных)::=



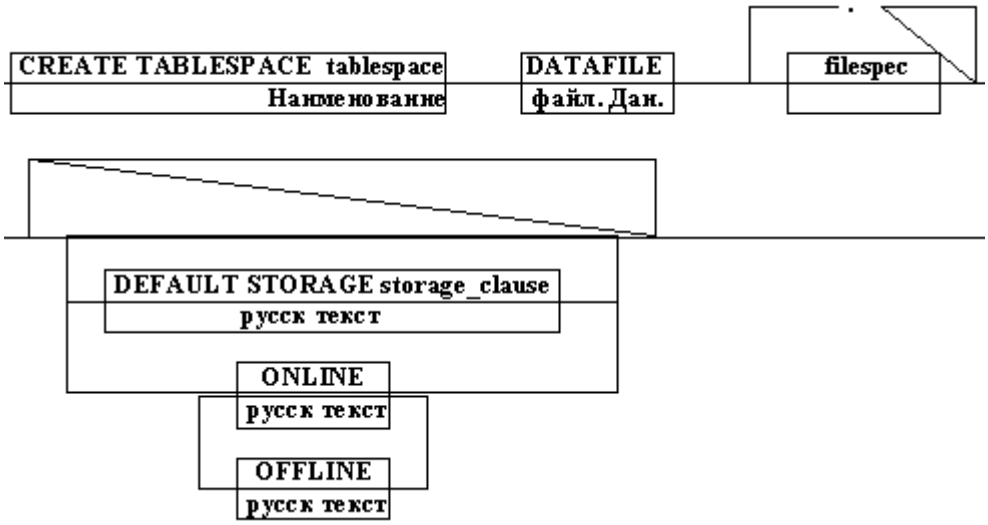
Filespec (журнальные файлы)::=




```
-- Скрипт создания БД CLINICS
spool clinic.log
connect internal
startup nomount pfile=/oracle/dbs/initclinic.ora
-- создаем базу данных с именем clinics
create database "clinics"
    maxinstances 1
    maxlogfiles  10
    character set "RU8PC866"
    national character set "RU8PC866"
    datafile
'/oracle/db/system01.dbf' size   100M
    logfile
'/oracle/db/lo  g01.dbf' size 1M,
'/oracle/db/log02.dbf' size 1M;
disconnect
spool off
```

CREATE TABLESPASE

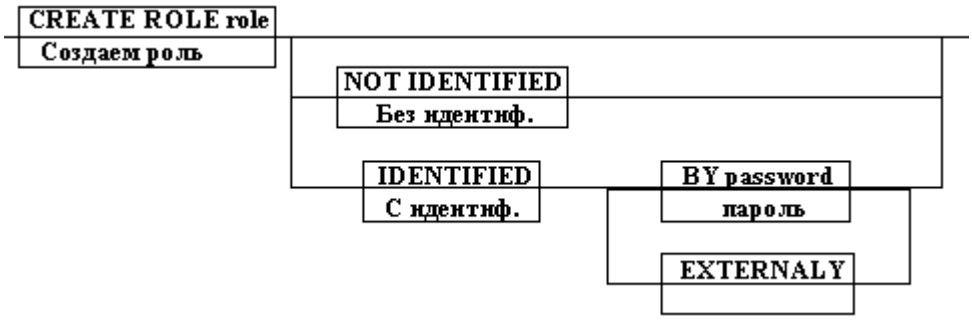
Создает в базе данных область для хранения таблиц, сегментов и индексов, определяет файлы базы данных, параметры хранения по умолчанию и режим табличного пространства (автономный или оперативный).



CREATE USER

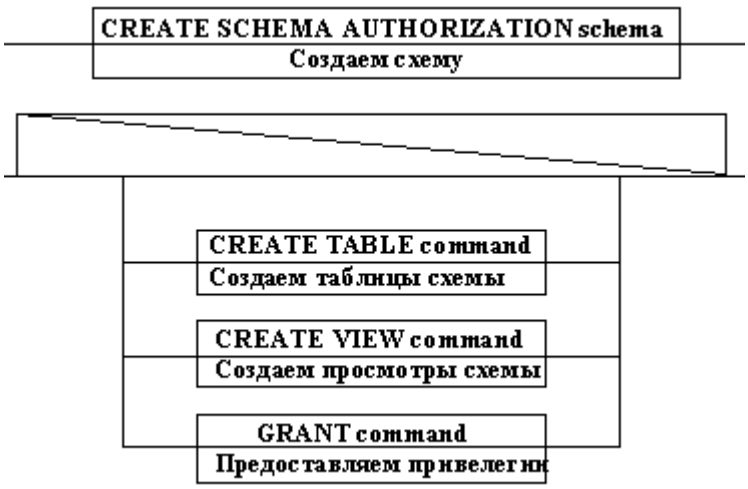
Создает пользователя базы данных. (синтаксис команды приведен упрощенно, за дополнительной информацией обратитесь к документации).

CREATE ROLE



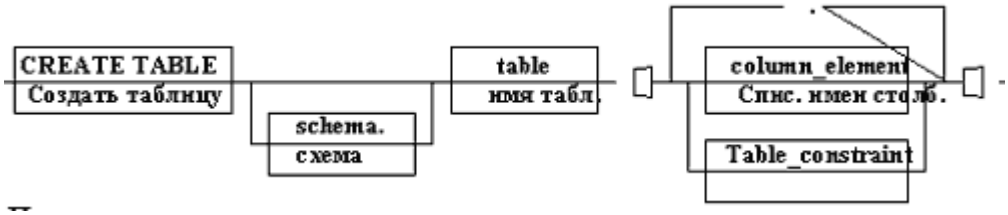
CREATE SCHEMA

Создает несколько таблиц и представлений и предоставляет некоторые привилегии в одной транзакции.



CREATE TABLE

Создает новую таблицу БД, определяя ее столбцы, правила целостности и параметры хранения.

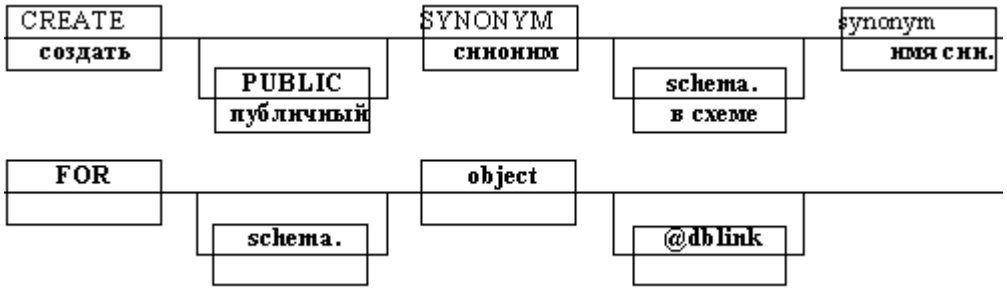


Пример:

```
create table        DOCTORS(
DC_NNN              NUMBER(12,0)
, DC_DC_NNN         NUMBER(12,0)
, DC_NAME           VARCHAR2(255)
, DC_CS_NNN         NUMBER(12,0)
, DC_DIPLOMA_NUMBER NUMBER(12,0)
, DC_SPECIALTY_NNN  NUMBER(12,0)
, DC_SHAT_NNN       NUMBER(12,0)
, DC_CALENDAR       NUMBER(12,0)
 )   tablespace users;
```

CREATE SYNONYM

Создает синоним для таблицы, представления, последовательности, хранимой процедуры или функции, пакетной процедуры, моментальной копии или другого синонима.

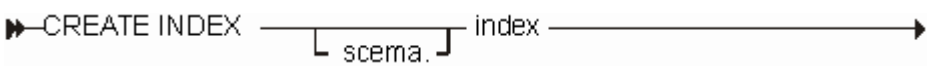


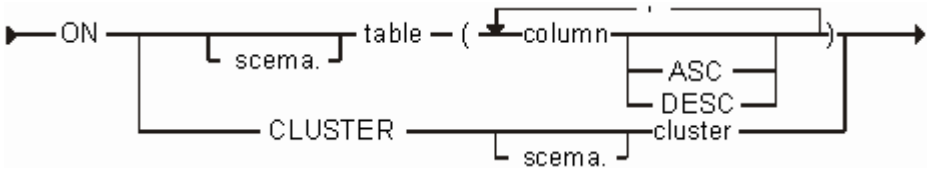
Пример: Сначала удаляем публичный синоним для таблицы DOCTORS, а потом его заново создаем.

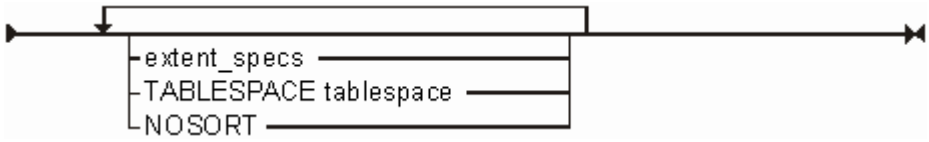
```
drop public synonym DOCTORS;
create public synonym DOCTORS for DOCTORS;
```

CREATE INDEX

Создает индекс для заданных столбцов таблицы или кластера.





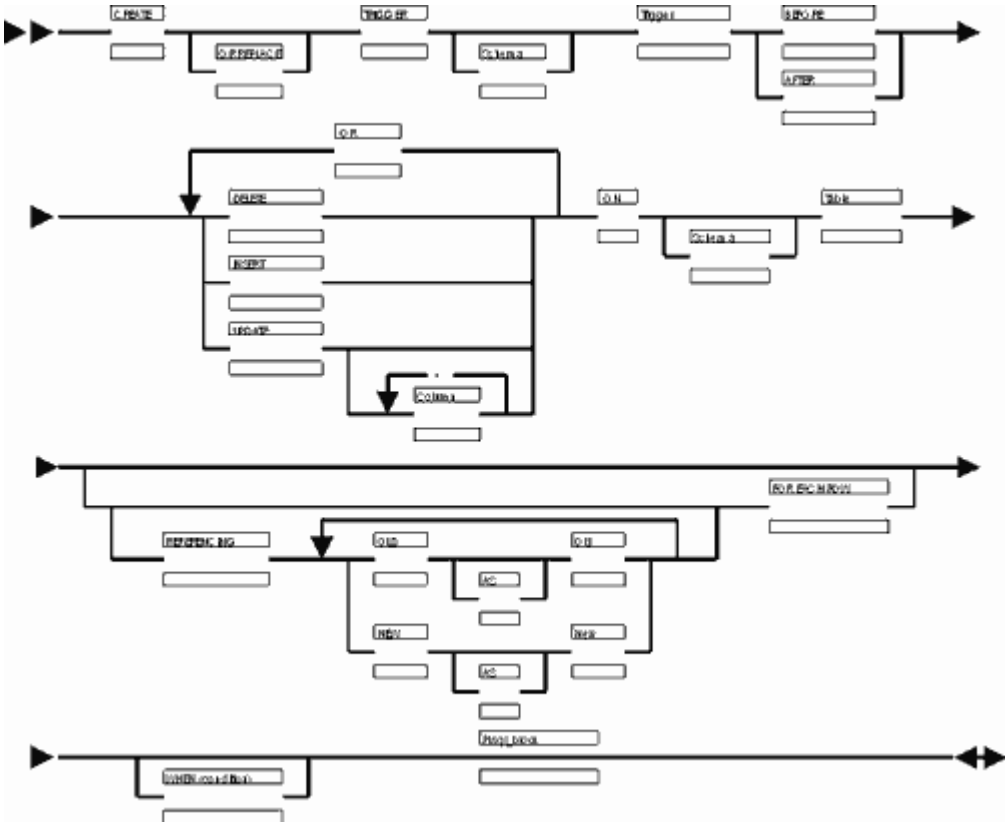


Пример:

```
create  UNIQUE  index I_DOCTORS  on DOCTORS (
   DC_NNN
 ) tablespace users;
```

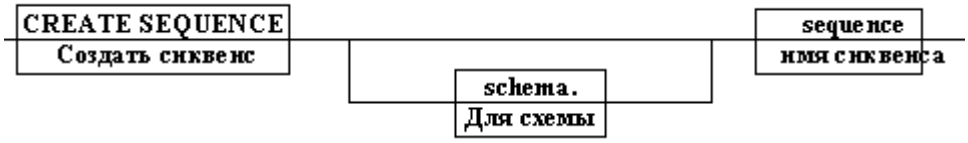
CREATE TRIGGER

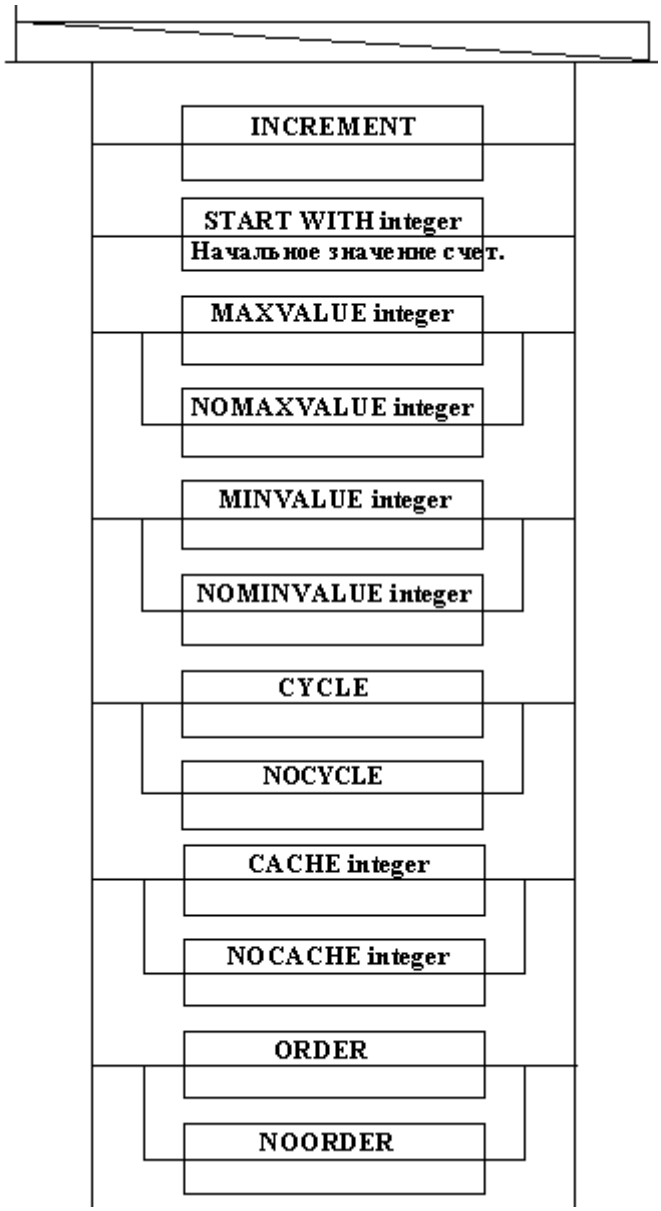
Создает триггер базы данных.



CREATE SEQUENCE

Создает новую последовательность для генерации первичных ключей.





Пример:

```
create sequence S_DOCTORS;
```

Пример: Приведем полный текст скрипта для создания триггера для таблицы DOCTORS, который будет следить за автоматическим увеличением значения поля DC_NNN на единицу при добавлении каждой новой записи в таблицу DOCTORS, что потребует от нас использование и такой конструкции - как сиквенс (генератор последовательностей).

```
-- Сначала удаляем предыдущие изменения в базе (если конечно они были сделаны)
drop sequence S_DOCTORS;
drop trigger tr_DC_NNN;
drop public synonym DOCTORS;
-- создаем таблицу
-- назначаем права доступа
-- создаем публичный синоним
CREATE PUBLIC SYNONYM DOCTORS FOR DOCTORS;
-- создаем сиквенс и устанавливаем его начальное значение в единицу
create sequence S_DOCTORS
start with 1;
-- создаем индексы
create unique index i_dc_nnn on doctors(dc_nnn);
```

```
create index i_dc_name on doctors(dc_name);
-- создаем триггер
create trigger tr_dc_nnn
before insert
on doctors
for each row
begin
  select dc_nnn.nextval into :new.dc_nnn from dual;
end;
/
```

ОПЕРАТОРЫ МАНИПУЛИРОВАНИЯ ДАННЫМИ.

Среди операторов SQL данного класса мы подробно рассмотрим только четыре основных оператора: INSERT - ВСТАВКА ДАННЫХ, SELECT - ВЫБОРКА ДАННЫХ, DELETE - УДАЛЕНИЕ ДАННЫХ, UPDATE - ИЗМЕНЕНИЕ ДАННЫХ.

INSERT

Вставить строки в таблицу или в базовую таблицу представления.

ОТСЮДА

img src="oracle_pr52.gif" border=0 WIDTH=500 HEIGHT=145>

Пример: В качестве примера рассмотрим вставку даннух в таблицу "Праздничные дни" (PRAZDNIKI)

```
insert into prazdniki values ('понедельник');
insert into prazdniki values ('вторник');
insert into prazdniki values ('среда');
insert into prazdniki values ('четверг');
insert into prazdniki values ('пятница');
insert into prazdniki values ('суббота');
insert into prazdniki values ('воскресенье');
```

SELECT

Выбирает данные из одной или нескольких таблиц или представлений. Может использоваться как оператор или как подзапрос в другом операторе.

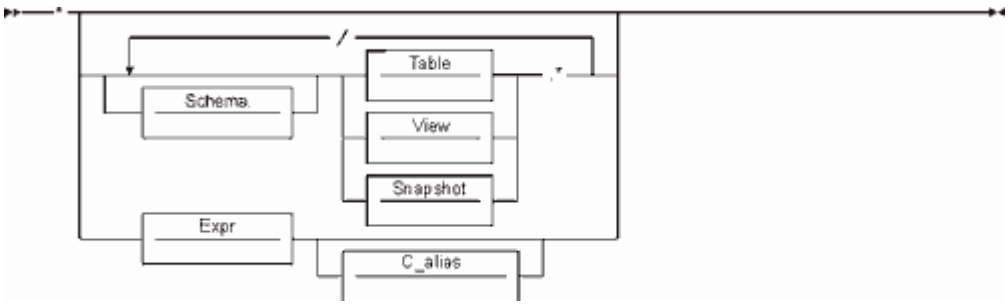
img src="oracle_pr53.gif" border=0 WIDTH=500 HEIGHT=82>

img src="oracle_pr54.gif" border=0 WIDTH=500 HEIGHT=132>

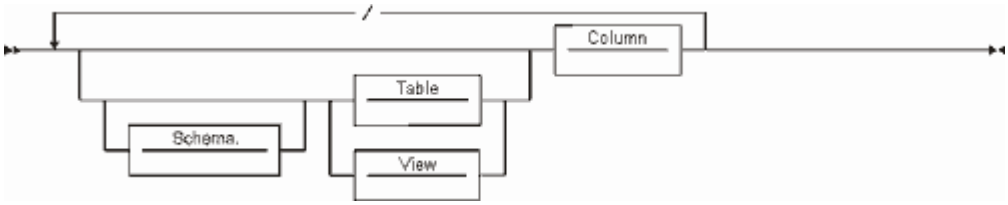
img src="oracle_pr55.gif" border=0 WIDTH=500 HEIGHT=134>

img src="oracle_pr56.gif" border=0 WIDTH=500 HEIGHT=155>

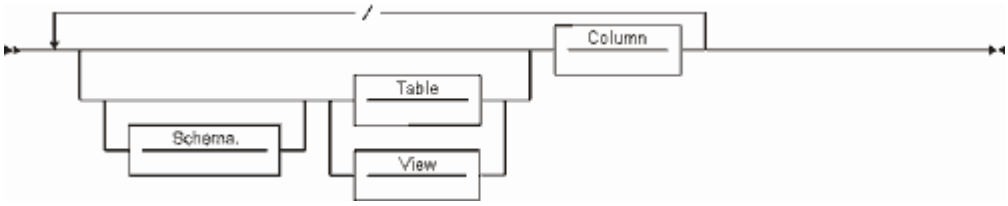
Select_list::=



table_list::=



update_list::=



Пример 1: Лучшим примером, иллюстрирующим работу оператора SELECT, является юмористический пример "Как программист SQL охотится на слонов". Дано: Слон живет в Африке. Задача: Что надо сделать чтобы найти слона? Метод решения: Программист SQL делает SELECT.

```
SELECT "СЛОН" FROM AFRICA; Итог: Все африканские слоны найдены.
```

Проиллюстрируем использование оператора SELECT на нескольких примерах.

Пример 2: Показать всех врачей заведенных в БД (см. рис.10).

```
SELECT * FROM doctors ORDER BY dc_name;
```

Результат: все записи из таблицы DOCTORS отсортированные по полю dc_name по алфавиту.

Пример 3. Показать всех врачей с кодом специальности равным 111.

```
SELECT dc_name
   FROM doctors
   WHERE dc_speciality_nnn = 111
ORDER BY dc_name;
```

Пример 4. Показать всех врачей с кодом специальности равным 111 или 112.

```
SELECT dc_name
   FROM doctors
   WHERE dc_speciality_nnn = 111
            OR dc_speciality_nnn = 112
ORDER BY dc_name;
```

2-ой способ

```
SELECT dc_name
   FROM doctors
   WHERE dc_speciality_nnn in (111, 112)
ORDER BY dc_name;
```

Операции над множествами в операторах SELECT

Операция	Выполняемые функции
UNION	Комбинирует два запроса; возвращает все неповторяющиеся строки, извлеченные хотя бы одним из запросов.
UNION ALL	Комбинирует два запроса; возвращает все строки, извлеченные хотя бы одним из запросов, включая повторяющиеся.
INTERSECT	Комбинирует два запроса; возвращает все неповторяющиеся строки, извлеченные каждым из запросов.
MINUS	Комбинирует два запроса; возвращает все неповторяющиеся строки, извлеченные первым запросом, но не извлеченные вторым.

Операция	Выполняемые функции
(+)	Указывает, что предшествующий столбец является столбцом внешнего соединения.
*	Используется вместо имен столбцов при выборке всех столбцов из таблицы или представления.
PRIOR	Используется в иерархическом древовидном запросе для определения зависимости между родительскими и дочерними строками. Смотрити оператор SELECT.
ALL	Оставляет повторяющиеся строки в результате запроса (установлен по умолчанию ALL, но не DISTINCT).
DISTINCT	Удаляет повторяющиеся строки из результата запроса.

Пример 5. Показать всех врачей с кодом специальности равным 111 и работающих в подразделении №2.

```
SELECT dc_name
   FROM doctors
   WHERE dc_speciality_nnn = 111
        AND dc_shtat_nnn = 2
ORDER BY dc_name;
```

Пример 6. Показать всех пациентов врача Иванова А. А.

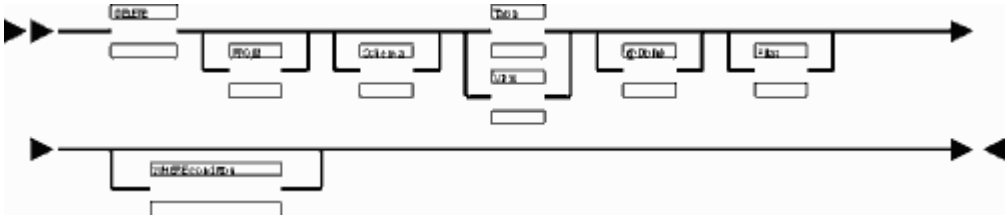
```
SELECT pt.pt_name
   FROM patients pt
   ,doctors dc
   WHERE dc.dc_nnn = pt.pt_dc_nnn
         AND dc.dc_name = 'ИВАНОВ А. А.'
ORDER BY pt.pt_name;
```

На этом примере остановимся подробнее: Первое - здесь впервые появились в запросе псевдонимы таблиц (pt, dc), это очень важный элемент , так как может оказаться, что по нерадивости у Вас в обоих таблицах имеются одинаковые наименования столбцов и тогда для обращения к ним потребуется

использование псевдонимов таблиц. Второе - Делая запрос к нескольким таблицам необходимо использовать джойны (dc.dc_nnn = pt.pt_nnn), т.е. явно задавать те поля, которые определяют отношения между таблицами, причем число джойнов равняется N-1, где N - число таблиц в запросе. Третье - выборка данных по условию dc.dc_name = 'ИВАНОВ А. А.' накладывает очень жесткие требования на правильность ввода данных (они могут быть набиты маленькими буквами, через несколько пробелов и т.п.), не учет этих особенностей приведет к тому, что некоторая нужная информация не будет выбрана. Чтобы избежать этого лучше в условиях использовать числовые поля, например личный номер врача (если он имеется БД).О принципах написание SELECT можно написать несколько томов, мы здесь изложили только несколько, с нашей точки зрения, важных особенностей, более подробную информацию по синтаксису можно всегда найти в справочной литературе.

DELETE

Удаляет строки из таблицы или из базовой таблицы представления, удовлетворяющие условию WHERE. Удаляет все строки, если условие WHERE не задано.

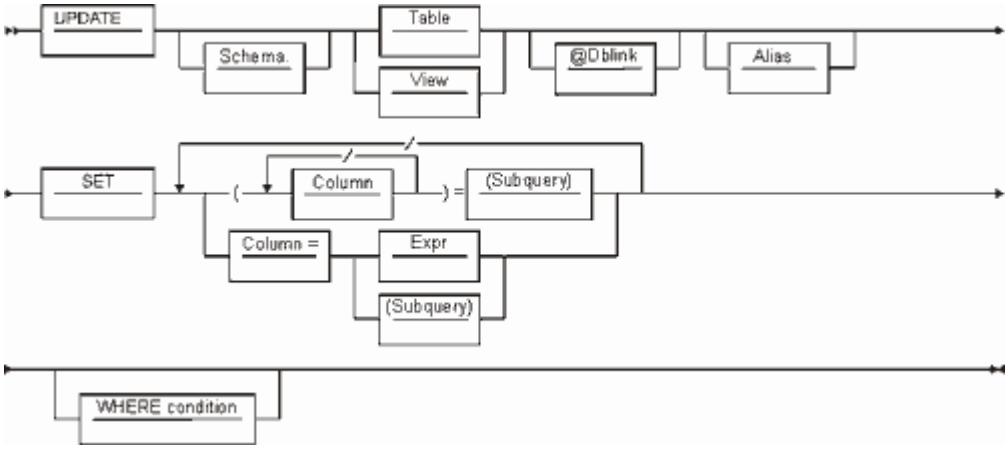


Пример:

Удаляем все записи из таблицы Праздничных дней.delete from prazdniki;

UPDATE

Изменяет существующие значения в таблице или в представлении (View).

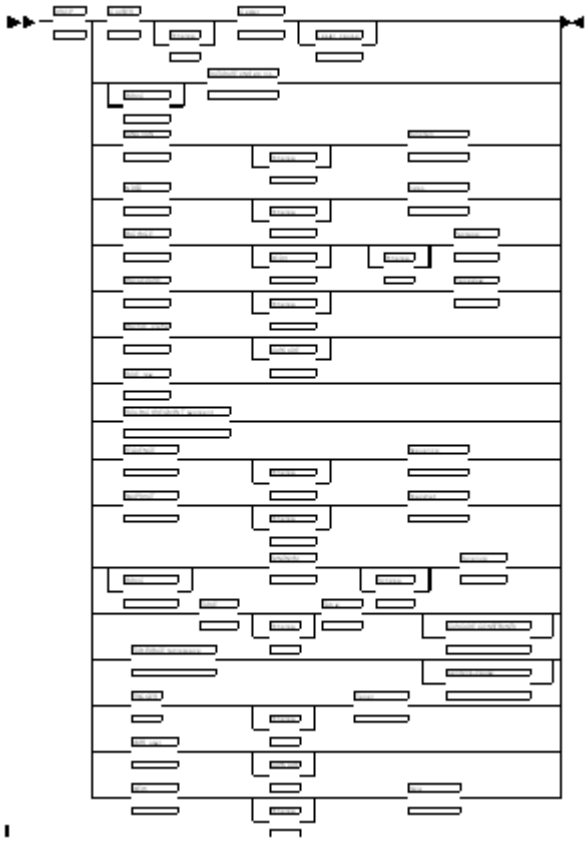


Операции над объектами базы данных.

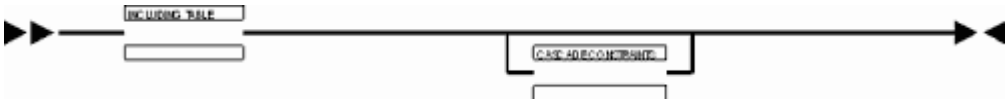
DROP

Эта команда удаляет объекты и ограничения из базы данных. Для этого действия требуются соответствующие привилегии. Например, для удаления общего канала связи базы данных требуется привилегия

DROP PUBLIC DATABASE LINK.



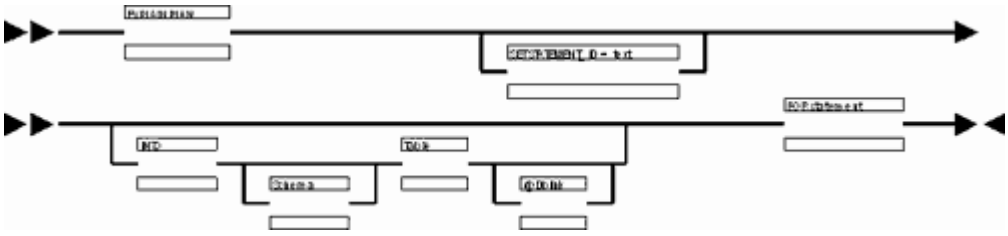
cluster_clause::=



contects_clause::=

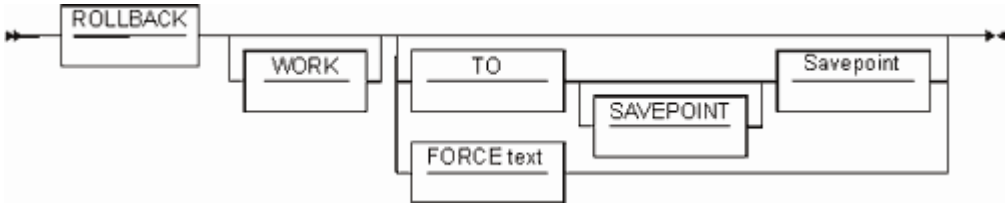
EXPLAIN PLAN

Описывает каждый шаг плана выполнения оператора SQL и помещает (если задано) это описание в указанную таблицу



ROLLBACK (управление транзакцией)

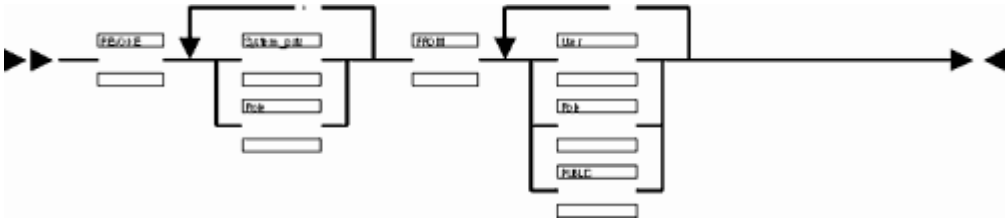
Отменяет все изменения, сделанные до контрольной точки. Отменяет все изменения, произведенные в текущей транзакции, если контрольная точка не задана.



Команды управления привилегиями и ролями

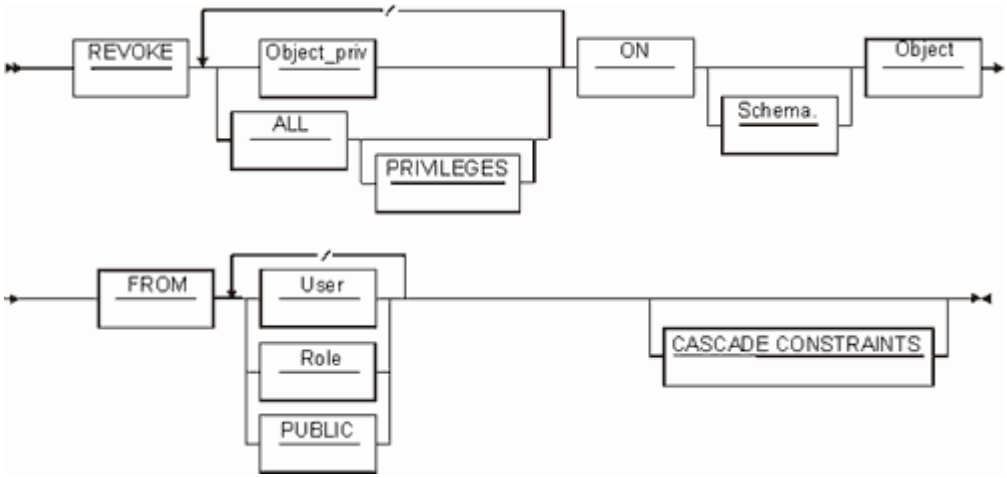
REVOKE (системные привилегии и роли)

Отменяет системные привилегии и роли, ранее предоставленные пользователям и ролям. Действие, обратное команде GRANT (системные привилегии и роли) .



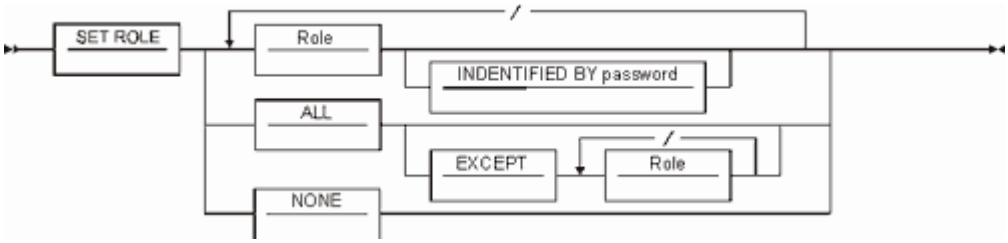
REVOKE (привилегии доступа к объектам)

Отменяет привилегии доступа к определенному объекту, ранее предоставленные пользователям и ролям. Действие, обратное команде GRANT (привилегии доступа к объектам). CREATE ROLE**Создает роль.**



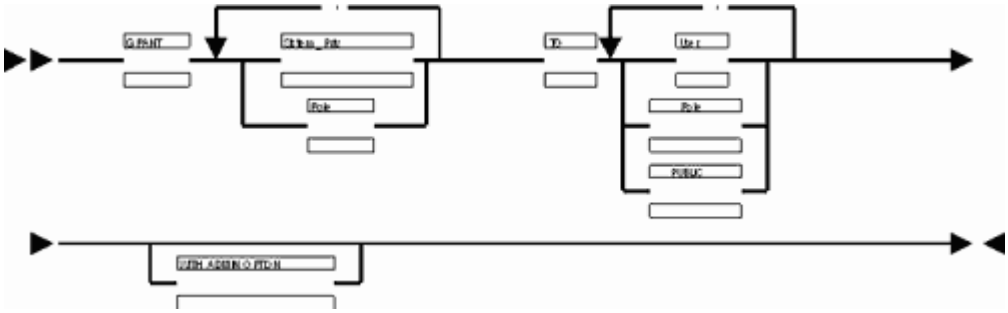
SET ROLE (управление сеансом)

Разрешает заданную роль в текущем сеансе и запрещает псе другие роли пользователя. Должна выполняться в начале транзакции вместе с оператором SET TRANSACTION.



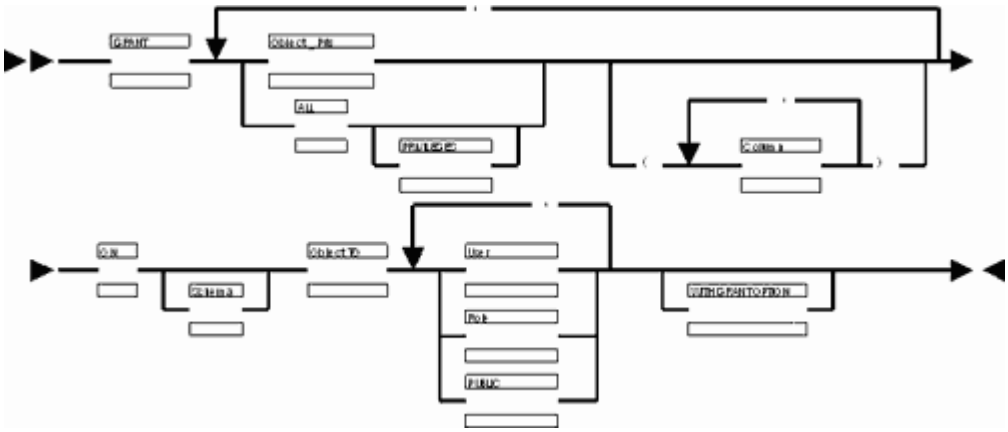
GRANT (системные привилегии и роли)

Предоставляет системные привилегии пользователям и ролям. Предоставляет роли пользователям и другим ролям.



GRANT (привилегии доступа к объектам)

Предоставляет привилегии доступа к определенным объектам (таблицам, представлениям, синонимам, пакетам, процедурам и т.д.) пользователям и ролям.



Назад | Содержание

2.3.2.4 Процедурное расширение языка SQL - PL/SQL.

$title="Oracle - технологии создания распределенных информационных систем";

Oracle также присущи черты, связанные с используемым языком программирования, которые способствуют ускорению разработки и улучшению эффективности серверной части приложений:

Один из основных компонентов Oracle Server - его процессор PL/SQL. (PL - Procedural Language - процедурный язык.)

PL/SQL - язык Oracle четвертого поколения, объединяющий структурированные элементы процедурного языка программирования с языком SQL, разработанный специально для организации вычислений в среде клиент/сервер. Он позволяет передать на сервер программный блок PL/SQL, содержаший логику приложения, как оператор SQL, одним запросом. Используя PL/SQL, можно значительно уменьшить объем обработки в клиентской части приложения и нагрузку на сеть. Например, может понадобиться выполнить различные наборы операторов SQL в зависимости от результата некоторого запроса. Запрос, последующие

операторы SQL и операторы условного управления могут быть включены в один блок PL/SQL и пересланы серверу за одно обращение к сети.

При этом вся логика приложений делится на клиентскую и серверную части. Серверная часть может быть релаизована в виде функций, хранимых процедур и пакетов.

Функции. Часть логики приложения ориентированной на выполнение конкретного комплекса операций на сервере, результат которых возвращается в виде значения функции. Откомпилированные функции и их исходные тексты содержатся в базе данных.

Хранимые процедуры. Часть логики приложения, особенно нуждающаяся в доступе к базе данных, может храниться там, где она обрабатывается (на сервере). Хранимые процедуры не возвращают значения результата, обеспечивают удобный и эффективный механизм безопасности. Откомпилированные хранимые процедуры и их исходные тексты содержатся в базе данных.

Пакеты. Часть логики приложений: фукций и пакетов, предназначеных для решениях задач в рамках одного модуля (подсистемы) АИС.

Триггеры базы данных. Можно использовать триггеры, чтобы организовать сложный контроль целостности, выполнять протоколирование (аудит) и другие функции безопасности, реализовать в приложениях выдачу предупреждений и мониторинг.

Декларативная целостность. Ограничения активизируются сервером всякий раз, когда записи вставляются, обновляются или удаляются. В дополнение к ограничениям ссылочной целостности, которые проверяют соответствие первичного и внешнего ключей, можно также накладывать ограничения на значения, содержащиеся в столбцах таблицы. Поддержка целостности на сервере уменьшает размер кода клиентской части, необходимого для проверки допустимости данных, и увеличивает устойчивость бизнес- модели, определенной в базе данных.

Список, зарезервированных слов PL/SQL

Язык PL/SQL также включает зарезервированные слова, имеющие определенное значение в операторах PL/SQL.

ABORT	DEFINITION	NOT	TADAUTH
ACCEPT	DELAY	NULL	TABLE
ACCESS	DELETE	NUMBER	TABLES
ADD	DELTA	NUMBER_BASE	TASK
ALL	DESC	OF	TERMINATE
ALTER	DIGITS	ON	THEN
AND	DISPOSE	OPEN	TO
ANY	DISTINCT	OPTION	TRUE
ARRAY	DO	OR	TYPE
ARRAYLEN	DROP	ORDER	UNION

AS	ELSE	OTHERS	UNIQUE
ASC	ELSIF	OUT	UPDATE
ASSERT	END	PACKAGE	USE
ASSIGN	ENTRY	PARTITION	VALUES
AT	EXCEPTION	PCTFREE	VARCHAR
AUTHORIZATION	EXCEPTION_INIT	PRAGMA	VARCHAR2
AVG	EXISTS	PRIOR	VARIANCE
BEGIN	EXIT	PRIVATE	VIEW
BETWEEN	FALSE	PROCEDURE	VIEWS
BODY	FETCH	PUBLIC	WHEN
BOOLEAN	FOR	RAISE	WHERE
BY	FORM	RANGE	WHILE
CASE	FROM	REAL	WITH
CHAR	FUNCTION	RECORD	WORK
CHAR_BASE	GENERIC	RELEASE	XOR
CHECK	GOTO	REM	
CLOSE	GRANT	RENAME	
CLUSTER	GROUP	RESOURCE	
CLUSTERS	HAVING	RETURN	
COLAUTH	IDENTIFIED	REVERSE	
COLUMNS	IF	REVOKE	
COMMIT	IN	ROLLBACK	
COMPRESS	INDEX	ROWNUM	
CONNECT	INDEXES	ROWTYPE	
CONSTANT	INDICATOR	RUN	
COUNT	INSERT	SAVEPOINT	
CRASH	INTEGER	SCHEMA	
CREATE	INTERSECT	SELECT	
CURRENT	INTO	SEPARATE	
CURSOR	IS	SET	
DATABASE	LEVEL	SIZE	
DATA_BASE	LIKE	SPACE	
DATE	LIMITED	SQL	
DBA	LOOP	SQLCODE	
DEBUGOFF	MAX	SQLERRM	
DEBUGON	MIN	START	
DECIMAL	MINUS	STATEMENT	
DECLARE	MOD	STDDEV	

DEFAULT	NEW	SUBTYPE	
	NOCOMPRESS	SUM	

Функции

Числовые функции

Функция	*Возвращаемое значение*
ABS(n)	Абсолютное значение величины *п.*
CEIL(n)	Наименьшее целое, большее или равное *п,*
COS(n)	Косинус *п* (угла, выраженного в радианах).
COSH(n)	Гиперболический косинус п.
EXP(я)	e в степени n.
FLOOR(n)	Наибольшее целое, меньшее или рапное *п.*
LN(n)	Натуральный логарифм п, где п>0.
LOG(m,n)	Логарифм м по основанию *т.*
MOD(m.n)	Остаток от деления *т* на *п.*
POWER(w.n)	*т в* степени *п.*
ROUND(n[,m])	*п,* округленное до *т* позиций после десятичной точки. По умолчанию *т* равно нулю.
SIGN(n)	Если.п<0,-1;еслии=0, 0; еслип>0, 1. ,
SIN(n)	Синус л (угла, выраженного в радианах).
SINHM	Гиперболический синус.
SQRT(n)	Квадратный корень отп. Если п<0, возвращает значение NULL.
TAN(n)	Тангенс я (угла, выраженного в радианах).
TANH(n) .	Гиперболический тангенс *п.*
TRUNC(n[,m])	*п,* усеченное до *т* позиций после от десятичной точки. По умолчанию *т* равно нулю.

Символьные функции

Символьные функции, возвращающие символьные значения:

Функция 1	*Возвращаемое значение*
CHR(n)	Символ с кодом *п.*
CONCAT(char1,c*har2)*	Конкатенация символьных строк *char1* и *char2.*
INITCAP(char)	Символьная строка char, первые буквы всех слов в которой преобразованы в прописные.
LOWER(char)	Символьная строка *char,* осе буквы которой

	преобразованы d строчные.
LPAD(char1.n *[,char2})*	Символьная строка *chur1,* которая дополняется слева последовательностью символов из *char2* так, чтобы общая длина строки стала равна *п.* Значение *chur2* по умолчанию -" (один пробел). Если часть многобайтового символа не помещается в добавляемой строке, то конец строки заполняется пробелами.
LTRIM(char[,set])	Символьная строка *char,* в которой удалены все символы от начала вплоть до первого символа, которого нет в строке set. Значение set по умолчанию - '' (один пробел).
NLS_INITCAP(char[,nls_sort])	Символьная строка *char,* в которой первые буквы всех слов преобразованы в прописные. Параметр *nIs_sort* определяет последовательность сортировки.
NLS_LOWER(char[,nls_sort])	Символьная строка *char,* все буквы которой преобразованы в строчные. Параметр *tils-sort* определяет последовательность сортировки.
NLS_UPPER(char[,nls_sort])	Символьная строка *char,* все буквы которой преобразованы в прописные. Параметр *nts_sort* определяет последовательность сортировки.
REPLACE(char, search_string [,*replacement_string])*	Символьная строка *char,* в которой все фрагменты *search_string* заменены на *replacement_string.* Если параметр *replacement_string* не определен, все фрагменты *search-string* удаляются.
RPAD(char1.n[,char2])	Символьная строка *charl,* которая дополнена справа последовательностью символов из *chur2* так, что общая длина строки равна n. Если часть многобайтового символа не помещается в добавляемой строке, то конец строки заполняется пробелами.
RTRIM(char[,set])	Символьная строка *char,* в которой удалены все символы справа вплоть до первого символа, которого нет в строке *set.* Значение параметра *set* по умолчанию -^{1} ^{1} (один пробел).
SOUNDEX(char)	Символьная строка, содержащая фонетическое представление для *char,* на английском языке.
SUBSTR(char,m[,n])	Фрагмент символьной строки *char,* начинающийся с символа *т,* длиной *п* символов (до конца строки, если параметр *п* не указан).
SUBSTRB(char,m[,n])	Фрагмент символьной строки *char,* начинающийся с символа *т,* длиной л байтов (до конца строки, если параметр *п* не указан).
TRANSLATE(char,from, *to)*	Символьная строка *char,* в которой все символы, встречающиеся в строке *from,* заменены на соответствующие символы из *to.*
UPPER(char)	Символьная строка *char,* в которой все буквы

	преобразованы в прописные.

Символьные функции, возвращающие числовые значения

Функция	*Возвращаемое значение*
ASCll(char)	Возвращает десятичный код первого символа строки *char* в кодировке, принятой в базе данных. (Код ASCII в системах, использующих кодировку ASCII). Возвращает значение первого байта многобайтового символа.
INSTR(charl.char2[,n[,m]])	Позиция первого символа m-ого фрагмента строки *charl,* совпадающего со строкой *char2,* начиная с n-ого символа. По умолчанию n u m равны 1. Номер символа отсчитывается от первого символа строки *charl,* даже когда *п*> 1.
INSTRB(charl.char2[,n[,m]])	Позиция первого символа *т-ого* фрагмента строки *charl,* совпадающего со строкой *char2,* начиная с m-ого байта. По умолчанию *п* и *т* равны 1. Номер байта отсчитывается от первого символа строки *charl,* даже когда л> 1. .
LENGTH(char)	Длина строки *char* в символах.
LENGTHB(c/iar)	Длина строки *char* в байтах.
NLSSORT(char1,char2[,n[,m]])	Зависящее от национального языка значение, используемое при сортировке строки *char.*

Групповые функции

Функция	*Возвращаемое значение*
AVG([DISTINCT|ALL]n)	Среднее значение от n, нулевые значения опускаются.
COUNT([ALL]*)	Число строк, извлекаемых в запросе или подзапросе.
COUNT(IDISTINCT|ALL] e*xpr)*	Число строк, для которых e*xpr* принимает не пустое значение.
MAX([D1STINCT|ALL] e*xpr)*	Максимальное значение выражения e*xpr.*
MIN((DISTINCT|ALL] e*xpr)*	Минимальное значение выражения e*xpr.*
STDDEV([DISTINCT|ALL] n)	Стандартное отклонение величины л, нулевые значения опускаются.
SUM([DISTINCT|ALL] *n)*	Сумма значений n
VARIANCE([DIST1NCTIALL]n)	Дисперсия величины *п,* нулевые значения опускаются.

Функции работы с датами

Функция	*Возвращаемое значение*
ADD-MONTHS (d,n)	Дата d плюс n месяцев.
LAST-DAY (d)	Последнее число месяца, указанного в d
MONTHS-BETWEEN (d,e)	Число месяцев между датами d1 и d*2.*
NEW-TIME (d,a,b)	Дата и время в часовом поясе a, соответствующие дате и времени в часовом поясе b, при этом d,a и b значения типа CHAR, определяющие часовые пояса.
NEW-DAY (d,char)	Дата первого после даты (/дня недели, название которого записано в *с1шг.*
SYSDATE	Текущая дата и время.

Усечение и округление дат

Функция	Возвращаемое значение
ROUND(d[,fmt])	Дата d, округленная до единиц, указанных в форматной маске.
TRUNC(d[,fmt])	Дата d, усеченная по форматной маске fmt.

Форматные маски дат для функций ROUND и TRUNC.

В таблице перечислены форматные маски, которые можно использовать в функциях ROUND и TRUNC. По умолчанию используется форматная маска "DD".

Форматная маска	Возвращаемое значение
CC или SCC	Первый день столетия
SYYYY или YYYY или YYY или YY или Y или YEAR или SYEAR	Первый день года (округляется до 1 июля)
Q	Первый день квартала (округляется до 16 числа второго месяца квартала)
MONTH или MON или MM или RM	Первый день месяца (округляется до 16 числа)
WW или IW	Тот же день недели, что и первый день текущего года
W	Тот же день недели, что и первый день текущего месяца
DDD или DDD или J	День
DAY или DY или D	Первый день недели
HH HH12 HH24	Час
MI	Минута

Функции преобразования

Функция	Возвращаемое значение

CHARTOROWID(char)	Char преобразуется из типа данных CHAR в тип данных ROWID
CONVERT(char, dest_char_set [,source_char_set])	Преобразует символьную строку из набора символов source_char_set в набор символов dest_char_set
HEXTORAW (char)	Преобразует значение char, содержащее шестнадцатиричные цифры, в значение типа данных RAW
RAWTOHEX (raw)	Преобразует raw в символьное значение, содержащее его шестнадцатиричный эквивалент
ROWIDTOCHAR (rowid)	Преобразует значение типа ROWID в значение типа CHAR
TO_CHAR (expr [,fmt [,'nls_num_fmt']])	Преобразует значение expr типа DATE или NUMBER в значение типа CHAR по формату форматной маски fmt. Если fmt отсутствует, значения типа DATE преобразуются по формату, заданному по умолчанию, и значения типа NUMBER- в значение типа CHAR с шириной, достаточной для того, чтобы вместить все значащие цифры. Значение 'nls_num_fmt' определяет связанные с языком форматные маски. В Trusted ORACLE преобразует значения MLS или MLS_LABEL в значение типа VARCHAR2
TO_DATE (char[,fmt [,'nls_lang']])	Преобразует char в значение типа DATE с помощью форматной маски fmt. Если fmt опускается, используется форматная маска для даты, принятая по умолчанию.'nls_ang' задает язык, используемый в названиях месяцев и дней
TO_MULTI_BYTE (char)	Преобразует однобайтовые символы, имеющие многобайтовые эквиваленты, в соответствующие многобайтовые символы
TO_NUMBER (char [,fmt [,'nls_lang']])	Преобразует char, содержащее число в формате, указанном параметром fmt, в значение типа NUMBER. 'nls_lang' задает язык, определяющий символы валют и числовые разделители
TO_SINGLE_BYTE (char)	Преобразует многобайтовые символы, имеющие однобайтовые эквиваленты, в соответствующие однобайтовые символы

Форматные маски.

Этот раздел описывет форматные маски дат и чисел.

Форматные маски дат в TO_CHAR и TO_DATE.

Элементы форматной маски даты перечислены в приведенной ниже таблице. Любую комбинацию этих элементов можно использовать как аргумент fmt функций TO_CHAR или TO_DATE. По умолчанию fmt равен 'DD-MON-YY'.

Элемент	Возвращаемое значение

формата	
SCC или CC	Столетие; если указано 'S' то перед датами до нашей эры ставится '-'.
YYYY или SYYYY	Год; если указано'S' то перед датами до нашей эры ставится '-'. YYY или YY или Y] Последние 3, 2, или1 цифра года.
IYYY	4 цифры года по стандарту ISO. IYY или IY или I] Последние 3, 2, или1 цифра года по стандарту ISO.
Y,YYY	Год с запятой в указанной позиции.
SYEAR или YEAR	Год, записанный словами, а не цифрами; если указано'S' то перед датами до нашей эры ставится '-'.
RR	Последние 2 цифры года; для указания года в других столетиях.
BC или AD	BC- до нашей эры(до н.э.); AD - нашей эры
B.C. или A.D.	B.C.- до нашей эры(до н.э.); A.D. - нашей эры
Q	Квартал (1, 2, 3, 4;JAN-MAR=1).
MM	Месяц(01-12; JAN=1).
RM	Нумерация месяцев римскими цифрами(I-XII; JAN=I).
MONTH	Название месяца, дополненное пробелами до 9-ти символов.
MON	Сокращенное название месяца.
WW или W	Неделя года (1-52) или месяца (1-5).
IW	Неделя года (1-52 или 1-53) по стандарту ISO.
DDD или DD или D	День года (1-366) или месяца (1-31) или недели (1-7).
DAY	Название дня, дополненное пробелами до 9-ти символов.
DY	Сокращенное название дня.
J	Дата юлианского календаря; число дней, считая с первого января 4712 года до н.э.
AM или PM	AM -до полудня,PM- после полудня
A.M. или P.M.	A.M. -до полудня,P.M.- после полудня
HH или HH12	Час дня (1-12).
HH24	Час дня (0-23).
MI	Минута (0-59)
SS или SSSSS	Секунда (0-59) или количество секунд после полуночи (0-86399).
-/,.;:	Знаки пунктуации.
"...текст..."	Текст воспр в возвращенном значении.

Префикы и суффиксы элементов формата даты

К элементам формата даты можно добавлять следующие префиксы:

FM	"Режим заполнения".Подавляет заполнение пробелами, когда стоит перед MONTH или DAY
FX	"Точный формат". Этот модификатор задает точное соответствие символьного аргумента и форматной маски даты в функции TO_DATE.

К элементам формата даты можно добавлятть следующие суффиксы:

TH	Порядковый номер ("DDTH" для "4TH").
SP	Номер, записанный словами ("DDSP" для "FOUR").
SPTH и THSP	Порядковый номер, записанный словами ("DDSPTH" для "FOURTH").

Прописные и строчные буквы в элементах формата даты.

Следующие строки задают вывод прописными буквами, вывод прописными буквами только начальных букв слов, или вывод строчными буквами.

Прописные	Прописная начальная	Строчные
DAY	Day	.day
DY	Dy	.dy
MONTH	Month	.month
MON	Mon	.mon
YEAR	Year	.year
AM	Am	.am
PM	Pm	.pm
A.M.	A.m.	a.m.
P.M.	P.m.	p.m.

Если к элементу формата даты добавляется префикс или суффикс, то регистр (прописные, строчные буквы) определяется элементом формаиа , а не префиксом или суффиксом. Например, 'ddTH ' задает "04th" а не "04TH".

Элементы формата числа для TO_CHAR

В следующей таблице перечислены элементы формата числа. Комбинацию этих элементов можно использовать как аргумент *fnu* функции TO_CHAR.

Элемент формата	*Пример*	*Описание*
9	'999'	Количество девяток указывает число возвращаемых значащих цифр.
0	'0999'	Добавляет нули перед числом.
$	'$9999'	Добавляет знак доллара перед числом.
B	'B9999'	Заменяет нулевые значения пробелами.

Ml	'99999MI'	Возвращает знак '-' после отрицательных значении.
S	S9999	Возвращает знак '+' для положительных значений и знак '-' для отрицательных значений в указанную позицию.
PR	'9999PR'	Возвращает отрицательные значения в <угловых скобках>.
D	99D99	Возвращает символ, представляющий десятичную точку, в указанную позицию.
C	9G999	Возвращает символ разделения цифр на группы в указанную позицию.
C	C999	Возвращает международной знак валюты в указанную позицию.
L	L999	Возвращает знак местной валюты в указанную позицию.
,	'9,999'	Возвращает запятую в указанную позицию.
.	'99.99'	Возвращает точку в указанную позицию.
V	'999V99'	Умножает значение на 10^n, где *n* количество девяток после 'V'.
EEEE	'9.999EEEE'	Возвращает значение в нормализованной форме. В *fnu* должно быть ровно четыре буквы 'E'.
RN или rn	RN	Возвращает римские цифры прописными или строчными буквами (целое число в диапазоне от 1 до 3999).
DATE	'DATE'	Возвращает значение, преобразованное из даты юлианского календаря в формат 'MM/DD/YY'.

Другие функции

Функция	Возвращаемое значение
DECODE (expr, search1, return1, [search2, return2,]...[default])	Если expr равно search, возвращается соответствующий результат return. Если совпадающей пары не найдено, возвращается default.
DUMP(expr[, return_format [, start_position[, length]]])	Expr во внутреннем формате Oracle
GREATEST(expr[, expr]...)	Наибольшее значение expr
LEAST(expr[, expr]...)	Наименьшее значение expr
NVL(expr1, expr2)	Возвращает expr2, если expr1 имеет пустое значение, в противном случае возвращает expr1.
UID	Целое число, которое уникально идентифицирует текущего пользователя.
USER	Имя текущего пользователя ORACLE.
USERENV(option)	Возвращает информацию о текущем сеансе.

	Аргументы помещаются в одиночных кавычках. Аргументы: ENTRYID, SESSIONSID, TERMINAL, LANGUAGE или LABEL.
VSIZE(expr)	Длина в байтах внутреннего проедставления для expr.

Подведем некоторые итоги: гибкость СУБД Oracle во многом определяется тем, что отдельные блоки кода PL/SQL программ можно хранить как объекты базы данных в формате хранимых процедур, функций и пакетов. Т.е. часть кода программы храниться там, где обрабатывается !!, т.е. на сервере.

Пакет - совокупность функций и процедур, объединенных по общему функциональному признаку, в тело пакетов входят процедуры и функции.

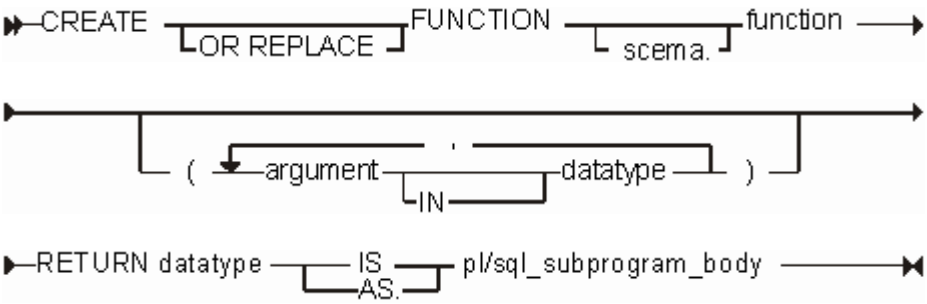
Процедура - объект базы данных обеспечивающий выполнение конкретных действий с передаваемыми параметрами процедуры.

Функция - объект базы данных обеспечивающий выполнение конкретных действий над параметрами функции и возвращающая результат такой обработки.

Для создания функций, процедур, пакетов базы данных используются следующие команды:

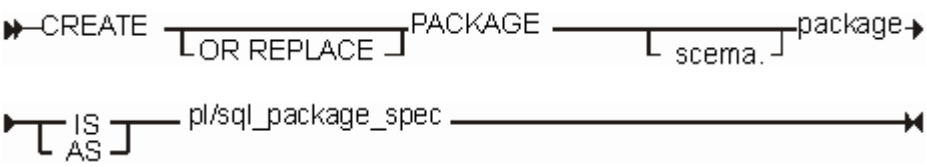
CREATE FUNCTION

Создает автономную хранимую функцию.



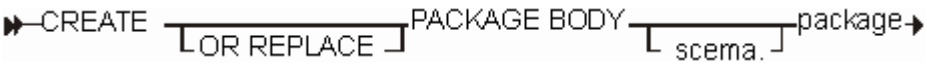
CREATE PACKAGE

Создает спецификацию для хранимого пакета.



CREATE PACKAGE BODY

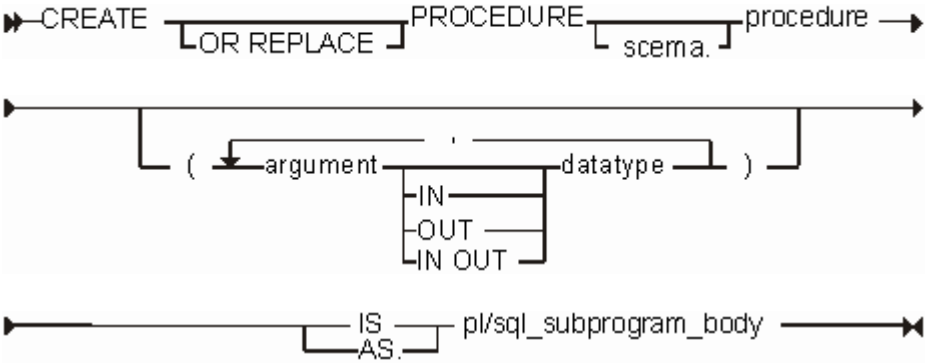
Создает тело хранимого пакета.



img src="oracle_pr77.gif" border=0 WIDTH=461 height=26>

CREATE PROCEDURE

Создает автономную хранимую процедуру



Приведем примеры реализации пакетов, функций и процедур.

```
/* ******Пакет обработки ошибок ******************** */
create or replace package app_err as
/* Получение текста аварийного завершения */
    function get_err  return varchar2;
/* Установка текста аварийного завершения */
    procedure set_err ( error_text in varchar2);
/* Установка текста аварийного завершения и само завершение */
    procedure raise_err ( error_text in varchar2);
 end app_err;
create or replace package body app_err as
 app_err_text varchar2(32767);
 /* ********************************************************** */
 function get_err return varchar2
 is
   A varchar (32767);
 Begin
   A := app_err_text;
   app_err_text := null;
   Return ( A );
 End;
 /* ********************************************************** */
 procedure set_err(error_text in varchar2)
    is
 Begin
    app_err_text := error_text;
 End;
 /* ********************************************************** */
 procedure raise_err ( error_text in varchar2)
 is
 Begin
    app_err_text := error_text;
    raise_application_error (-20000, error_text);
 End;
 end app_err;
/* ** Функция осуществляющая расшифровку пароля пользователя в БД ** */
create or replace function password return varchar2 is
  name   varchar2(23);
  pass   varchar2(23);
  begin
  select ORANAME, CRYPT_PASSWORD
  into name, pass
  from USERS
  where USER_ORANAME = user;
  pass   := encrypt( pass, name );
  return( pass );
  end;
```

```
/*Удаляется публичный синоним*/
drop public synonym password;
/*Создается публичный синоним*/
create public synonym password for password;
/* устанавливаем привелегии*/
grant all on password to admin;
```

2.3.2.5 Системные объекты базы данных.

Словарь данных. Первыми таблицами, создаваемыми в любой базе данных, являются системные таблицы, или словарь данных Oracle. Системные таблицы хранят информацию о структуре базы данных и объектов внутри нее, и Oracle обращается к ним, когда нуждается в информации о базе данных или когда выполняет оператор DDL (Data Definition Language - язык определения данных) либо оператор DML (Data Manipulation Language - язык манипулирования данными). Эти таблицы никогда непосредственно не обновляются, однако обновление в них происходит в фоновом режиме всякий раз, когда выполняется оператор DDL. Главные таблицы словаря данных содержат нормализованную информацию, которая является довольно трудной для восприятия человеком, так что в Oracle предусмотрен набор представлений, выдающих информацию главных системных таблиц в более понятном виде. Oracle запрашивает информацию из таблиц словаря данных для синтаксического разбора любого оператора SQL. Информация кэшируется в области словаря данных разделяемого пула в SGA.

Сегменты отката. Когда данные в Oracle изменяются, изменение должно быть или подтверждено, или отменено. Если изменение отменяется ("откатывается назад"), содержимое блоков данных восстанавливается в исходное состояние, существовавшее до изменения. Сегменты отката - это системные объекты, которые поддерживают этот процесс. Всякий раз, когда осуществляются какие-либо изменения в таблицах приложения или в системных таблицах, в сегмент отката автоматически помещается предыдущая версия изменяемых данных, так что старая версия данных всегда доступна, если требуется отказ. Другие пользователи при необходимости чтения данных, в то время как изменение не завершено, всегда имеют доступ к прежней версии из сегмента отката. Им предоставляется *непротиворечивая по чтению* версия данных. После того как изменение фиксируется, доступной становится измененная версия данных. Сегменты отката получают внешнюю память таким же образом, как другие сегменты - экстентами. Сегменту отката, однако, нужно первоначально распределить минимум два экстента.

Временные сегменты используют пространство в файлах базы данных, чтобы создать временную рабочую область для промежуточных стадий обработки SQL и для больших операций сортировки. Oracle создает временные сегменты в процессе работы и они автоматически удаляются, когда фоновый процесс SMON больше в них не нуждается. Если требуется только небольшая рабочая область, Oracle не создает временного сегмента, но вместо этого как временная рабочая область используется часть памяти PGA (глобальная область программы). Администратор базы данных может определять, в каких табличных пространствах будут располагаться временные сегменты для различных пользователей.

Сегмент начальной загрузки (или кэш-сегмент) - специальный тип объекта в базе данных, выполняющий начальную загрузку кэша словаря данных в область

разделяемого пула SGA. Oracle использует кэш-сегмент только при запуске экземпляра и не обращается к нему вплоть до рестарта экземпляра. Сегмент необходим, чтобы выполнить начальную загрузку кэша словаря данных, после чего занимаемая им память освобождается.

Словарь данных.

Фундаментальное различие между RDBMS и другими БД и файловыми системами заключается в способе доступа к данным. RDBMS позволяет обращаться к физическим данным в более абстрактной, логической форме, обеспечивая легкость и гибкость при разработке кода приложения. Программы, использующие RDBMS, обращаются к данным через "машину" базы данных без непосредственной зависимости от фактического источника данных, изолируя приложение от деталей "нижележащих" физических структур данных. RDBMS сама заботится о том, где поле хранится в базе данных. Такая независимость данных возможна благодаря словарю данных RDBMS, который хранит метаданные (данные о данных) для всех объектов, расположенных в базе данных.

Словарь данных Oracle - множество таблиц и объектов базы данных, которое хранится в специальной области базы данных и ведется исключительно ядром Oracle. Словарь данных содержит информацию об объектах базы данных, пользователях и событиях. К этой информации можно обратиться с помощью представлений словаря данных. Как показано на рис.31, запросы чтения или обновления базы данных обрабатываются ядром Oracle с использованием информации из словаря данных.

Информация в словаре данных предназначена для подтверждения существования объектов, обеспечения доступа к ним и описания фактического физического расположения в памяти.

RDBMS не только обеспечивает размещение данных, но также определяет оптимальный путь доступа для хранения или выборки данных. Oracle использует сложные алгоритмы, которые позволяют выбирать информацию с наибольшей производительностью, исходя из критерия скорейшего получения первых строк результата или критерия минимального времени выполнения запроса в целом.

Представления словаря данных

Словарь данных содержит информацию об объектах базы данных, пользователях и событиях. К этой информации можно обратиться с помощью представления словаря данных:

ALL_OBJECTS	Объекты, доступные пользователю.
ALL_SEQUENCES	Описание последовательностей, доступных пользователю.
ALL_SNAPSHOTS	Все моментальные копии, доступные пользователю.
ALL_SOURCE	Исходный текст объектов, доступных пользователю.
ALL_SYNONYMS	Все синонимы, доступные пользователю.
ALL_TABLES	Описание таблиц, доступных пользователю.
ALL_TAB_COLUMNS	Столбцы всех таблиц, представлений и кластеров, доступных пол.

ALL_TAB_COMMENTS	Комментарии к таблицам и представлениям, доступным пользователю.
ALL_TAB_PRIVS	Привилегии на объекты, которые получил пользователь непосредственно, через роль или как PUBLIC.
ALL_TAB_PRIVS_MADE	Привилегии пользователя и привилегии на его объекты.
ALL_TAB_PRIVS_RECD	Привилегии на объекты, которые получил пользователь непосредственно, через роль или как PUBLIC.
ALL_TRIGGERS	Триггеры, доступные пользователю.
ALL_TRIGGER_COLS	Использование столбцов в пользовательских триггерах, в триггерах для его таблиц или во всех триггкрах, если он имеет привелегию CREATE ANY TRIGGER.
ALL_USERS	Информация о всех пользователях базы данных.
ALL_VIEWS	Текст представлений, доступных пользователю.
AUDIT_ACTIONS	Коды типов аудиторских действий.
CAT	Синоним для USER_CATALOG.
CLU	Синоним для USER_CLUSTERS.
CODE_PIECES	Используется для создания представлений _OBJECT_SIZE.
CODE_SIZE	Используется для создания представлений _OBJECT_SIZE.
COLS	Синоним для USER_TAB_COLUMNS.
COLUMN_PRIVILEGES	Привилегии на столбцы, которые принадлежат пользователю, которые он выдал или получил непосредственно, через роль или как пользователь PUBLIC.
DBA_2PC_NEIGHBORS	Информация от вновь поступивших и отработанных запросов задержанных транзакций.
DBA_2PC_PENDING	Информация о транзакциях, в которых произошел сбой во время фазы подготовки.
DBA_AUDIT_EXISTS	Журнал записей протокола, созданных командой AUDIT EXISTS.
DBA_AUDIT_OBJECT	Журнал протокола команд над объектами. Создается в файле CATAUDIT.SQL.
DBA_AUDIT_SESSION	Журнал протокола команд входа и выхода из ORACLE.
DBA_AUDIT_STATEMENT	Синоним для USER_AUDIT_STATEMENT.
DBA_AUDIT_TRAIL	Журнал протокола всей системы.
DBA_BLOCKERS	Все сеансы, которые держат блокировки, которых ожидает кто-то другой.
DBA_CATALOG	Все таблицы, представления, синонимы и последовательности, принадлежащие пользователю.
DBA_CLUSTERS	Описание всех кластеров.
DBA_CLU_COLUMNS	Соответствие столбцов таблиц столбцам кластера.

DBA_COL_COMMENTS	Комментарии к столбцам всех таблиц и представлений.
DBA_COL_PRIVS	Привилегии на все столбцы базы данных.
DBA_CONSTRAINTS	Определения правил целостности для всех таблиц базы данных.
DBA_CONS_COLUMNS	Информация о столбцах в определениях правила целостности, созданных пользователем.
DBA_DATA_FILES	Файлы базы данных.
DBA_DB_LINKS	Все связи базы данных.
DBA_DDL_LOCKS	Все блокировки DDL в базе данных и все связанные с ними запросы к блокировкам DML.
DBA_DEPENDENCIES	Зависимости (от) всех объектов базы данных.
DBA_DML_LOCKS	Все блокировки DML в базе данных и все связанные с ними запросы к блокировкам DML.
DBA_ERRORS	Текущие ошибки для всех хранимых объектов.
DBA_EXP_FILES	Описание экспортных файлов.
DBA_EXP_OBJECTS	Объекты, которые экспортировались.
DBA_EXP_VERSION	Номер версии последнего экспорта.
DBA_EXTENTS	Экстенты всех сегментов базы данных.
DBA_FREE_SPACE	Свободные экстенты в табличных пространствах, доступных пользователю.
DBA_INDEXES	Описание индексов, доступных пользователю.
DBA_IND_COLUMNS	Столбцы индексов пользователь или его индексированных таблиц.
DBA_LOCKS	Все блокировки и задержки в базе данных, а также все поступающие на них запросы.
DBA_OBJECTS	Все объекты базы данных.
DBA_OBJECT_SIZE	Размер объектов PL/SQL базы данных.
DBA_OBJ_AUDIT_OPTS	Параметры аудиторства для всех таблиц и представлений.
DBA_PRIV_AUDIT_OPTS	Параметры аудиторства для привилегий.
DBA_ROLES	Все роли в базе данных.
DBA_ROLE_PRIVS	Роли, выданные пользователям или другим ролям.
DBA_ROLLBACK_SEGS	Описание сегментов отката базы данных.
DBA_SEGMENTS	Распределение пространства для всех сегментов базы данных.
DBA_SEQUENCES	Описание всех последовательностей в базе данных.
DBA_SNAPSHOTS	Все моментальные копии в базе данных.
DBA_SNAPSHOT_LOGS	Все журналы моментальных копий в базе данных.
DBA_SOURCE	Исходный текст всех хранимых объектов.
DBA_STMT_AUDIT_OPTS	Параметры системного аудиторства.
DBA_SYNONYMS	Все синонимы в базе данных.

DBA_SYS_PRIVS	Системные привилегии, выданные пользователям или ролям.
DBA_TABLES	Описание всех таблиц базы данных.
DBA_TABLESPACES	Описание всех табличных пространств в базе данных.
DBA_TAB_COLUMNS	Столбцы всех таблиц, представлений и кластеров.
DBA_TAB_COMMENTS	Комментарии к таблицам и представлениям базы данных.
DBA_TAB_PRIVS	Привилегии на объекты всей базы данных.
DBA_TRIGGERS	Описание всех триггеров базы данных.
DBA_TRIGGERS_COLS	Использование столбцов в пользовательских триггерах или в триггерах для его таблиц.
DBA_TS_QUOTAS	Квоты всех пользователей в табличном пространстве.
DBA_USERS	Информация о всех пользователях базы данных.
DBA_VIEWS	Текст всех представлений базы данных.
DBA_WAITERS	Все сеансы, ожидающие или владеющие блокировками.
DBMS_ALERT_INFO	Таблица регистрируемых сигналов тревоги.
DBMS_LOCK_ALLOCATED	Таблица пользовательских блокировок.
DEPTREE	Дерево зависимости объектов.
DICT	Синоним для DICTIONARY.
DICTIONARY	Описание таблиц и представлений словаря данных.
DICT_COLUMNS	Описание столбцов таблиц и представлений словаря данных.
ERROR_SIZE	Используется для создания представлений _OBJECT_SIZE.
GLOBAL_NAME	Содержит одну строку с глобальным именем текущей базы данных.
IDEPTREE	Отсортированный, сформатированный вариант DEPTREE.
IND	Синоним для USER_INDEXES.
INDEX_HISTOGRAM	Содержит статистику команды ANALYZE INDEX VALIDATE STRUCTURE.
INDEX_STATS	Содержит статистику команды ANALYZE INDEX VALIDATE STRUCTURE.
LOADER_COL_INFO	Представление SQL*LOADER, используемое для прямой загрузки.
LOADER_CONSTRAINT_INFO	Представление SQL*LOADER, используемое для прямой загрузки.
LOADER_INDCOL_INFO	Представление SQL*LOADER, используемое для прямой загрузки.
LOADER_IND_INFO	Представление SQL*LOADER, используемое для прямой загрузки.
LOADER_PARAM_INFO	Представление SQL*LOADER, используемое для прямой загрузки.

LOADER_TAB_INFO	Представление SQL*LOADER, используемое для прямой загрузки.
LOADER_TRIGGER_INFO	Представление SQL*LOADER, используемое для прямой загрузки.
OBJ	Синоним для USER_OBJECTS.
PARSED_PIECES	Используется для создания представлений _OBJECT_SIZE.
PARSED_SIZE	Используется для создания представлений _OBJECT_SIZE.
PUBLIC_DEPENDENCY	Зависимости между объектами.
RESOURCE_COST	Стоимость каждого ресурса.
ROLE_ROLE_PRIVS	Информация о ролях, назначенных другим ролям.
ROLE_SYS_PRIVS	Информация о системных привилегиях, назначенных ролям.
ROLE-TAB-PRIVS	Информация об объектных привилегиях, назначенных ролям.
SEQ	Синоним для USER_SEQUENCES.
SESSION-PRIVS	Привилегии, которые пользователь имеет в настоящий момент.
SESSION-ROLES	Роли, включенные для пользователя в настоящий момент.
SOURCE-SIZE	Используется для создания представлений - OBJECT_SIZE.
STMT_AUDIT_OPTION_MAP	Таблица описания кодов типов параметров протоколирования.
SYN	Синоним для USE_SYNONYMS.
SYSTEM_PRIVILEGE_MAP	Таблица описания кодов системных привилегий.
TABLE_PRIVILEGES	Привилегии на объекты, к которым пользователь получил привилегии, выдал, для которых он является владельцем или привилегия выдана пользователю PUBLIC.
TABS	Синоним для USER_TABLES.
USER_AUDIT_OBJECT	Записи протокольного журнала для команд обращающихся к объекту.
USER_AUDIT_.SESSION	Записи протокольного журнала о входах и выходах в систему.
USER_AUDIT_STATEMENT	Записи протокольного журнала о следующих командах: GRANT, REVOKE, AUDIT, NOAUDIT и ALTER SYSTEM.
USER_AUDIT_TRAIL	Записи протокольного журнала относящиеся к пользователю.
USER_CATALOG	Все таблицы, представления, синонимы и последовательности, принадлежащее пользователю.
USER_CLUSTERS	Описание кластеров пользователя.
USER_CLU_COLUMNS	Соответствие столбцов таблиц столбцам кластера.

USER_COL_COMMENTS	Комментарии к столбцам пользовательских таблиц и представлений.
USER_COL_PR1VS	Привилегии на столбцы, для которых пользователь является владельцем, выдал или получил привилегии.
USER_COL_PRIVS_MADE	Привилегии на столбцы, для которых пользователь является владельцем.
USER_COL_PRIVS_RECD	Привилегии на столбцы, для которых пользователь является владельцем, выдал или получил привилегии.
USER_CONSTRAINTS	Определения правил целостности для таблиц пользователя.
USER_CONS_COLUMNS	Информация о столбцах в определениях правила целостности, созданных пользователем.
USER.DB_LINKS	Связи базы данных, принадлежащие пользователю.
USER_DEPENDENCIES	Зависимости объектов пользователя.
USER_ERRORS	Текущие ошибки для всех объектов, принадлежащих пользователю.
USER_EXTENTS	Экстенты сегментов, выделенные объектам, принадлежащим пользователю.
USER_FREE_SPACE	Свободные экстенты в табличных пространствах, доступных пользователю
USER_INDEXES	Описание индексов, доступных пользователю
USER_IND_COLUMNS	Столбцы индексов пользователь или его индексированных таблиц.
USER_OBJECTS	Объекты, принадлежащие пользователю.
USER_OBJECT_SIZE	Размер объектов PL/SQL, принадлежащих пользователю.
USER_OBJ_AUDIT_OPTS	Параметры аудиторства для пользовательских таблиц и представлении.
USER_RESOURCE_LIMITS	Ограничения на ресурсы, доступные текущему пользователю.
USER_ROLE_PRIVS	Роли, выданные текущему пользователю.
USER_SEGMENTS	Распределение пространства для сегментов объектов пользователя.
USER_SEQUENCES	Описание последовательностей, созданных пользователем.
USER_SNAPSHOTS	Все моментальные копии, доступные пользователю.
USER_SNAPSHOT_LOGS	Все журналы моментальных копий, принадлежащие пользователю.
USER_SOURCE	Исходный текст хранимых объектов, принадлежащих пользователю.
USER_SYNONYMS	Все частные синонимы пользователя.
USER_SYS_PRIVS	Системные привилегии, выданные текущему пользователю.
USER_TABLES	Описание таблиц пользователя.
USER_TABLESPACES	Описание доступных табличных пространств.

USER_TAB_COLUMNS	Столбцы всех таблиц, представлений и кластеров.
USER_TAB_COMMENTS	Комментарии к таблицам и представлениям, принадлежащим пользователю
USER_TAB_PR1VS	Привилегии на объекты, для которых пользователь является владельцем, выдал или получил привилегии.
USER_TAB_PRIVS_MADE	Все привилегии на объекты, принадлежащие пользователю.
USER_TAB_PR1VS_RECD	Привилегии на объекты, которые получил пользователь.
USER_TRIGGERS	Описание всех пользовательских триггеров.
USER_TRIGGER_COLS	Использование столбцов в пользовательских триггерах или в триггерах для его таблиц.
USER_TS_QUOTAS	Квоты пользователя в табличном пространстве.
USER_USERS	Информация о текущем пользователе базы данных.
USER_VIEWS	Текст представлений пользователя.

Динамические таблицы производительности, доступные пользователю SYS, позволяют управлять производительностью работы сервера СУБД.

V$ACCESS	Заблокированные на текущий момент объекты и сеансы, в которых они используются.
V$ARCHIVE	Информация о журналах архива для каждого потока системы базы данных. .
V$BACKUP	Статус сброса всех ON-LINE баз данных.
V$BGPROCESS	Описание фоновых процессов.
V$CIRCUIT	Информация о виртуальных цепях.
V$DATABASE	Информация из контрольного файла о базе данных.
V$DATAFILE	Информация из контрольного файла о файлах базы данных.
V$DBFILE	Информация о всех файлах базы данных.
V$DB-OBJECT-CACHE	Объекты базы данных, находящиеся в библиотечном кеше.
V$DISPATCHER	Информация о процессах диспетчера.
V$ENABLEDPRIVS	Включенные привилегии.
V$F1LESTAT	Информация о статистике ввода/вывода в файл.
V$FIXED-TABLE	Все таблицы, представления и производные та
V$INSTANCE	блицы в базе данных.
V$INSTANCE	Статус текущего экземпляра
V$ LATCH	Число задержек каждого типа. (Строки этой таблицы однозначно соответствуют строкам таблицы V$ATCHHOLDER)
V$LATCHHOLDER	Информация о владельцах задержек.
V$LATCHNAME	Закодированные имена задержек из таблицы V$ATCH.
V$LIBRARYCACHE	Статистика по управлению буферами библиотечной памяти.

V$LICENSE	Параметры лицензии.
V$ADCSTAT	Статистика SQL*Loader при выполнении прямой загрузки.
V$LOADTSTAT	Статистика SQL* Loader при выполнении прямой загрузки.
V$LOCK	Блокировки и ресурсы.
V$LOG	Информация о журнальном файле.
V$LOGFILE	Информация о журнальных файлах.
V$LOGHIST	Информация об истории журнального файла.
V$LOG-HISTORY	Информация об истории журнального файла.
U$NLS-PARAMETERS	Текущие значения параметров NLS.
V$OPEN-CURSOR	Открытые пользователями курсоры.
V$PARAMETER	Информация о текущих значениях параметров.
V$PROCESS	Информация о всех активных процессах.
V$QUEUE	Информация об очереди мульти-серверных сообщений.
V$RECOVERY-LOG	Журнальные файлы, необходимые для полного восстановления базы данных.
V$RECOVER-FILE	Статус файлов, которые нужно восстанавливать.
V$REQD1ST	Гистограмма времен обращения, разделенная на 12 столбцов или периодов времени.
V$RESOURCE	Информация о ресурсах.
V$ROLLNAME	Имена всех активных сегментов отката.
V$ROLLSTAT	Статистика для всех активных сегментов отката.
V$ROWCACHE	Статистика активности словаря данных. (Одна строка для каждого буфера памяти)
V$SECONDARY	Представление Trusted ORACLE, u котором перечислены вторичнее смонтированные базы данных.
V$SESS10N	Информация о текущих сеансах.
V$SESS10N-WA1T	Список ресурсов или событий, которых ожидает текущий сеанс.
V$SESSTAT	Статистика для текущих сеансов.
V$SGA	Суммарная информация об SGA.
V$SHARED-SERVER	Информация о всех разделяемых процессах сервера.
V$SQLAREA	Статистика о разделяемых буферах памяти курсора. Одна строка для каждого курсора.
V$SQLTEXT	Текст команд SQL, находящихся в разделенных курсорах SGA.
V$STATNAME	Раскодированные имена для статистик .из таблицы V$SESSTAT.
V$SYSLABEL	Представление Trusted ORACLE, в котором перечислены системные метки.
V$SYSSTAT	Текущие значения статистик из таблицы V$SESSTAT.
V$THREAD	Информация о потоках, содержащихся п контрольном файле.
V$TIMER	Текущее время в сотых долях секунды.

V$TRANSACTION	Информация о транзакциях.
V$TYPE-SIZE	Размеры различных компонентов базы данных.
V$VERSION	Имена версии компонентов библиотеки ядра ORACLE.
V$WAITSTAT	Статистика содержимого блока. Обновляется только при включенной временной статистики.

Специальные таблицы

Таблица CHAINED_ROWS

Список сцепленных строк таблицы или кластера, использованного в команде ANALYZE.

Столбец	*Тип данных*
OWNER-NAME	VARCHAR2
TABLE-NAME	VARCHAR2
CLUSTER-NAME	VARCHAR2
HEAD_ROWID	ROWID
TIMESTAMP	DATE

Таблица EXCEPTIONS

Эта таблица используется для определения строк, нарушающих правила целостности, если правила целостности включены.

Столбец	*Тип данных*
HEAD.ROWID	ROWID
OWNER	VARCHAR2
TABLE-NAME	VARCHAR2
CONSTRAINT	VARCHAR2

PLAN_TABLE

Эта таблица может заполняться командой EXPLAIN PLAN для того, чтобы описать план выполнения оператора SQL.

Столбец	*Тип данных*
STATEMENT.ID	VARCHAR2
TIMESTAMP	DATE
REMARKS	VARCHAR2
OPERATION	VARCHAR2
OPTIONS	VARCHAR
OBJECT_NODE	VARCHAR2
OBJECT_OWNER	VARCHAR2

OBJECT.NAME	VARCHAR
OBJECT_INSTANCE	NUMBER
OBJECT_TYPE	VARCHAR2
SEARCH_COLUMNS	NUMBER
ID	NUMBER
PARENT.ID	NUMBER
POSITION	NUMBER
OTHER	LONG

2.3.3 Защита данных.

Транзакции, фиксация и откат. Изменения в базе данных не сохраняются, пока пользователь явно не укажет, что результаты вставки, модификации и удаления должны быть зафиксированы окончательно. Вплоть до этого момента изменения находятся в отложенном состоянии, и какие-либо сбои, подобные аварийному отказу машины, аннулируют изменения.

Транзакция - элементарная единица работы, состоящая из одного или нескольких операторов SQL;

Все результаты транзакции или целиком сохраняются (фиксируются), или.целиком отменяются (откатываются назад). Фиксация транзакции делает изменения окончательными, занося их в базу данных, и после того как транзакция фиксируется, изменения не могут быть отменены. Откат отменяет все вставки, модификации и удаления, сделанные в транзакции; после отката транзакции ее изменения не могут быть зафиксированы. Процесс фиксации транзакции подразумевает запись изменений, занесенных в журнальный кэш SGA, в оперативные журнальные файлы на диске. Если этот дисковый ввод/вывод успешен, приложение получает сообщение об успешной фиксации транзакции. (Текст сообщения изменяется в зависимости от инструментального средства.) Фоновый процесс DBWR может записывать блоки актуальных данных Oracle в буферный кэш SGA базы данных позже. В случае сбоя системы Oracle может автоматически повторить изменения из журнальных файлов, даже если блоки данных Oracle не были перед сбоем записаны в файлы базы данных.

Oracle также реализует идею отката на уровне оператора. Если произойдет единственный сбой при выполнении оператора, весь оператор завершится неудачей. Если оператор терпит неудачу в пределах транзакции, остальные операторы транзакции будут находиться в отложенном состоянии и должны либо фиксироваться, либо откатываться.

Все блокировки, захваченные транзакцией, автоматически освобождаются, когда транзакция фиксируется или откатывается, или когда фоновый процесс PMON отменяет транзакцию. Кроме того, другие ресурсы системы (такие как сегменты отката) освобождаются для использования другими транзакциями.

Точки сохранения позволяют устанавливать маркеры внутри транзакции таким образом, чтобы имелась возможность отмены только части работы, проделанной в транзакции. Целесообразно использовать точки сохранения в длинных и сложных транзакциях, чтобы обеспечить возможность отмены изменения для определенных операторов. Однако это обусловливает дополнительные затраты ресурсов системы - оператор выполняет работу, а изменения затем

отменяются; обычно усовершенствование в логике обработки могут оказаться более оптимальным решением. Oracle освобождает блокировки, захваченные отмененными операторами.

Целостность данных связана с определением правил проверки достоверности данных гарантирующих, что недействительные данные не попадут в ваши таблицы. Oracle позволяет определять и хранить эти правила для объектов базы данных, которых они касаются, таким образом, чтобы кодировать их только однажды. При этом они активируются всякий раз, когда какой-либо вид изменения проводится в таблице, независимо от того, какая программа выполняет вставки, модификации или удаления. Этот контроль осуществляется в форме ограничений целостности и триггеров базы данных.

Ограничения целостности устанавливают бизнес-правила на уровне базы данных, определяя набор проверок для таблиц системы, Эти проверки автоматически выполняются всякий раз, когда вызываются оператор вставки, модификации или удаления данных в таблице. Если какие-либо ограничения нарушены, операторы отменяются. Другие операторы транзакции остаются в отложенном состоянии и могут фиксироваться или отменяться согласно логике приложения.

2.3.4 Привилегии системного уровня

$title="Oracle - технологии создания распределенных информационных систем";

Например, прежде чем создать триггер для таблицы (даже если вы владелец таблицы как пользователь Oracle), нужно иметь системную привилегию, называемую CREATE TRIGGER, назначенную вашему учетному разделу пользователя Oracle, или роли, присвоенной учетному разделу.

Привилегия CREATE SESSION - другая часто используемая привилегия системного уровня. Чтобы выполнить соединение с базой данных, учетный раздел Oracle должен иметь привилегию системного уровня CREATE SESSION.

Привилегии объектного уровня. Привилегии объектного уровня обеспечивают возможность выполнить определенный тип действия (выбрать, вставить, модифицировать, удалить и т.д.) с указанным объектом. Владелец объекта имеет полный контроль над объектом и может выполнять любые действия с ним; он не обязан иметь привилегии объектного уровня. Фактически владелец объекта - пользователь Oracle, который может предоставлять привилегии объектного уровня другим пользователям.

Например, если пользователь, который владеет таблицей, желает, чтобы другой пользователя вставлял и выбирал строки из его таблицы (но не модифицировал или удалял), он предоставляет другому пользователю привилегии (объектного уровня) отбора и вставки для этой таблицы. Вы можете предоставлять привилегии объектного уровня непосредственно пользователям или ролям, которые затем назначаются учетным разделам пользователей Oracle.

Привилегии выдаются пользователям и ролям командой GRANT и отбираются командой REVOKE. Все привелегии можно разделить на системные и объектные. Системные привилегии относятся ко всему классу объектов, а объектные относятся к заданным объектам.

Системные привелегии

Наименование	Назначение
ANALYZE ANY	Позволяет анализировать любые таблицы, кластеры или индексы в любой схеме
AUDIT ANY	Позволяет протоколировать любой объект в любой схеме
TRUNCATE ANY	Позволяет удалить все строки любой таблицы или кластера в любой схеме
CREATE CLUSTER	Позволяет создать кластер в собственной схеме
ALTER ANY CLUSTER	Позволяет изменить любой кластер в любой схеме
CREATE ANY CLUSTER	Позволяет создать кластер в любой схеме
DROP ANY CLUSTER	Позволяет удалить любой кластер в любой схеме
ALTER DATABASE	Позволяет изменить базу данных
CREATE DATABASE LINK	Позволяет создать личный канал доступа в собственной схеме
CREATE PUBLIC DATABASE LINK	Позволяет создать общий канал доступа
DROP PUBLIC DATABASE LINK	Позволяет удалить общий канал доступа
CREATE INDEX	Позволяет создать индекс в собственной схеме для любой таблицы этой схемы
ALTER ANY INDEX	Позволяет изменить индекс в любой схеме
CREATE ANY INDEX	Позволяет создать индекс в любой схеме для любой таблицы любой схемы
DROP ANY INDEX	Позволяет удалить любой индекс в любой схеме
CREATE PROCEDURE	Позволяет создать хранимые процедуры, функции и пакеты в собственной схеме
ALTER ANY PROCEDURE	Позволяет изменить любую хранимую процедуру, функцию и пакет в любой схеме
CREATE ANY PROCEDURE	Позволяет создать хранимые процедуры, функции и пакеты в любой схеме
DROP ANY PROCEDURE	Позволяет удалить хранимые процедуры, функции и пакеты в любой схеме
EXECUTE ANY PROCEDURE	Позволяет выполнить любую хранимую процедуру, функцию и пакет или ссылку на общую переменную пакета в любой схеме
CRANT ANY PRIVILEGE	Позволяет выдать системные привилегии, даже если вы ими не обладаете
ALTER PROFILE	Позволяет изменить любой профиль в базе данных
CREATE PROFILE	Позволяет создать профиль
DROP PROFILE	Позволяет удалить любой профиль в базе данных

ALTER RESOURCE COST	Позволяет задать стоимости ресурсов сеанса
CREATE ROLE	Позволяет создать роли
ALTER ANY ROLE	Позволяет изменить любую роль в базе данных
DROP ANY ROLE	Позволяет удалить любую роль в базе данных
GRANT ANY ROLE	Позволяет предоставить любую роль в базе данных
ALTER ROLLBACK SEGMENT	Позволяет изменить сегмент отката
CREATE ROLLBACK SEGMENT	Позволяет создать сегмент отката
ROP ROLLBACK SEGMENT	Позволяет удалить сегмент отката
ALTER SESSION	Позволяет изменить параметры текущего сеанса работы: средства трассировки SQL, национальный язык или канал связи базы данных
CREATE SESSION	Позволяет соединиться с базой данных
RESTRICTED SESSION	Позволяет войти после запуска базы данных с параметром STARTUP RESTRICT
CREATE SEQUENCE	Позволяет создать последовательность в собственной схеме
ALTER ANY SEQUENCE	Позволяет изменить любую последовательность в любой схеме
CREATE ANY SEQUENCE	Позволяет создать последовательность в любой схеме
DROP ANY SEQUENCE	Позволяет удалить любую последовательность в любой схеме
SELECT ANY SEQUNCE	Позволяет обратиться к любой последовательности в любой схеме
CREATE SNAPSHOT	Позволяет создать моментальную копию в собственной схеме. Требует привилегии CREATE TABLE
ALTER ANY SNAPSHOT	Позволяет изменить любую моментальную копию в любой схеме
CREATE ANY SNAPSHOT	Позволяет создать моментальную копию в любой схеме. Требует привилегии CREATE ANY TABLE
DROP ANY SNAPSHOT	Позволяет удалить любую моментальную копию в любой схеме
CREATE SYNONYM	Позволяет создавать синоним в собственной схеме
CREATE ANY SYNONYM	Позволяет создать синоним в любой схеме
DROP ANY SYNONIM	Позволяет удалить любой синоним в любой схеме, кроме общих синонимов
CREATE PUBLIC SYNONYM	Позволяет создать общий синоним
DROP PUBLIC SYNONYM	Позволяет удалить общий синоним

AUDIT SYSTEM	Позволяет протоколировать системные события
ALTER SYSTEM	Позволяет изменить параметры системы: ограничения ресурсов, процессы разделяемого сервера или процессы диспетчера, группы журнальных файлов, контрольные точки, распределенное восстановление, проверку доступа к файлам
CREATE TABLE	Позволят создавать таблицу в собственной схеме. Требует привилегии UNLIMITED TABLESPACE или квоту в табличном пространстве
ALTER ANY TABLE	Позволяет изменить любую таблицу в любой схеме
BACKUP ANY TABLE	Позволяет использовать утилиту Export для экспорта любой таблицы в любой схеме
COMMENT ANY TABLE	Позволяет комментировать любую таблицу или столбец в любой схеме
CREATE ANY TABLE	Позволяет создать любую таблицу в любой схеме
DELETE ANY TABLE	Позволяет удалить строки любой таблицы, представления или моментальной копии в любой схеме
DROP ANY TABLE	Позволяет удалить любую таблицу в любой схеме
INSERT ANY TABLE	Позволяет добавить строки в любую таблицу, представление или моментальную копию в любой схеме
LOCK ANY TABLE	Позволяет блокировать любую таблицу в любой схеме
SELECT ANY TABLE	Позволяет сделать запрос из любой таблицы представления или моментальной копии в любой схеме
UPDATE ANY TABLE	Позволяет изменить строки любой таблицы, представления или моментальной копии в любой схеме
ALTER TABLESPACE	Позволяет изменить табличное пространство
CREATE TABLESPACE	Позволяет создать табличное пространство
DROP TABLESPACE	Позволяет удалить табличное пространство
MANAGE TABLESPACE	Позволяет переводить табличное пространство в автономный и оперативный режимы, а также начинать и завершать спасение табличного пространства
UNLIMITED TABLESPACE	Позволяет использовать неограниченное количество любого табличного пространства
CREATE TRIGGER	Позволяет создать триггер в собственной схеме
ALTER ANY TRIGGER	Позволяет разрешить, запретить или откомпилировать любой триггер в любой схеме
CERATE ANY TRIGGER	Позволяет создать триггер в любой схеме, связанный с любой таблицей любой схемы
DROP ANY TRIGGER	Позволяет удалить любой триггер в любой схеме
ALTER USER	Позволяет изменить у других пользователей пароль, табличные пространства и квоты, присвоить профили и роли, назначенные по умолчанию

BECOME USER	Позволяет стать другим пользователем. Используется программой Import при загрузке данных в схему другого пользователя
CREATE USER	Позволяет создать пользователей, установить квоты в любом табличном пространстве, установить табличное пространство по умолчанию и временное табличное пространство, присвоить профили
DROP USER	Позволяет удалить другого пользователя
CREATE VIEW	Позволяет создать представление в собственной схеме
CREATE ANY VIEW	Позволяет создать представление в любой схеме
DROP ANY VIEW	Позволяет удалить любое представление в любой схеме

Объектные привилегии

Наименовнаие	Назначение
ALL	Выдаются все привилегии для данного объекта
ALL PRIVILEGES	То же что и ALL
ALTER	Позволяет изменить определение
DELETE	Позволяет удалить строки
EXECUTE	Позволяет выполнить объект, а также осуществлять доступ к его переменным
INDEX	Позволяет создавать индекс
INSERT	Позволяет добавлять строки
REFERENCES	Позволяет создавать ограничение, которое ссылается на таблицу. Эту привилегию нельзя предоставлять роли
SELECT	Позволяет осуществлять запрос
UPDATE	Позволяет изменять строки

Соответствие между объектами быза данных и привелегиями.

Наименование привилегии	Таблицы	Представления	Последова-тельности	Процедуры Функции Пакеты	Моментальные копии
ALTER	X		X		
DELETE	X	X			
EXECUTE				X	
INDEX	X				
INSERT	X	X			
REFERENCES	X				
SELECT	X	X	X		X
UPDATE	X	X			X

Пользователи и роли.

Роль - тип объекта, который используется, для упрощения управления привилегиями системного и объектного уровня. Вместо того, чтобы назначать привилегии непосредственно учетным разделам пользователей, можно назначать привилегии ролям, которые затем назначаются пользователям.

Роли, по существу, - группы привилегий системного и объектного уровня. Они делают управление привилегиями намного проще, так как можно один раз сконфигурировать привилегии для отдельных типов пользователей и затем назначать эти привилегии ролям. Когда пользователь нуждается в какой-то группе привилегий, можно использовать единственную команду назначения роли, чтобы удовлетворить этого пользователя. Без ролей пришлось бы вводить по несколько команд для каждой из требуемых привилегий.

Кроме того, можно создавать различные роли с привилегиями, даже если еще нет учетных разделов пользователей Oracle, которые нуждаются в этих ролях. Можно назначать роль другой роли, формируя их иерархию. Также можно защитить роль паролем, который пользователь должен ввести для активизации роли.

Как уже говорилось, физическая база данных может содержать много учетных разделов пользователей Oracle, которые защищены паролями. Вы должны ввести имя пользователя и пароль независимо от того, какой инструмент вы используете, для получения доступа к базе данных. Роли - это не то же самое, что пользователи Oracle; вы не можете соединяться с базой данных, вводя имя роли и пароль.

Протоколирование (аудит).

Механизм протоколирования Oracle обеспечивает три типа протоколов.

1. Протокол привилегий прослеживает, какие системные привилегии используются.
2. Протокол операторов отслеживает, какие операторы SQL используются для всех объектов.
3. Протокол объектного уровня контролирует доступ к объектам.

Вы можете запустить эти протоколы, чтобы проследить, когда операторы отрабатываются успешно и когда они терпят неудачу. Вы можете использовать протоколирование, чтобы отследить любую попытку вторжения в систему.

Кроме того, можно устанавливать, что записывается в протокол. Может записываться один элемент на операцию независимо от того, сколько попыток выполнить операции было сделано в течение сеанса соединения. Или, альтернативно, записывается один элемент протокола для каждой попытки (успешной или нет) операции в течение сеанса.

Информация протокола хранится в таблице словаря данных, которая содержит протоколируемую операцию, идентификатор пользователя, выполняющего операцию, дату и время операции. Oracle обеспечивает набор представлений словаря данных, чтобы сделать информацию в таблице протокола более читабельной. Вставленные в протокол строки продолжают храниться, даже если пользователь отменяет транзакцию.

Администратор базы данных может периодически очищать или архивировать протокол.

2.3.5. Поддержка национальных языков

Средство поддержки национальных языков Oracle (National Language Support - NLS) позволяет пользователям использовать базу данных на их собственных языках. Это средство обеспечивает следующие функции:

1. Поддержка различных схем кодирования, т.е. данные, созданные в схеме кодирования на одной машине, могут быть обработаны и представлены на другой.
2. Управление языком вывода ошибок сервера и информационных сообщений, чисел, дат, форматов валюты и начального дня недели.
3. Поддержка лингвистической сортировки гарантирует, что символы появляются в корректном порядке.

Можно добавлять поддержку для новых языков, используя программный продукт NLS*WorkBench, который, по существу, поддерживает таблицы перевода для интерпретации ввода от пользователя и для вывода на экран результатов.

Когда в поставку прикладной системы входят приложения на различных языках, наиболее важной частью пользовательского интерфейса являются различные подсказки, библиотека стандартных текстов и сообщения приложения. В настоящее время непосредственно разработчики приложения определяют, как библиотека стандартных текстов, подсказки и сообщения прикладной системы изменяются от одного языка к другому. Oracle работает над программным продуктом автоматического перевода с целью упрощения решения этой задачи.

Приложение 1.

Практическое задание по курсу "Разработка и эксплуатация конструкторско-технологических баз данных"

Разработать, используя инструментальные средства разработки и СУБД Oracle, автоматизированную систему управления конструкторско-технологическим проектированием (АСУ КТП), включающую базу данных и пользовательские приложения для работы с ней.

Этапы выполнения работы:

1. Разработка архитектуры и технологических взаимосвязей взаимодействия пользователей с автоматизированной системы управления конструкторско-технологическим проектированием (АСУ КТП) на предприятии радиопромышленности (предприятие состоит из подразделений: администрация, отдел автоматизации, конструкторский отдел, отдел технологической подготовки производства, производство - цех, в каждом из которых имеется по два автоматизированных рабчих места - руководителя (manager) и исполнителя - разработчика (developer)).

   Итог - функциональная структура предприятия с указанием имен сотрудников (как реальных, так и ораклических (пользовательских)) и модель процессов проектирования, т.е. продвижения документации по подразделениям с указанием прав доступа конкретных пользователей к конкретным документам.

2. Установка trial версии СУБД Personal Oracle, ее настройка и заведение всех пользователей АСУ КТП, назначив им имена и привилегии.

Итог: работоспособная база данных с определенным табличным пространством USER (где будут созданы пользовательские таблицы).

3. Формализация функциональной модели АСУ КТП (логической модели). Разработка табличной структуры БД АСУ КТП и используя CASE средства провести моделирование спроектированной структуры базы данных на работоспособность.

   Итог - документирование информационных потоков, ER - диаграммы и справочник таблиц БД АСУ КТП.

4. Проектирование общесистемного меню АСУ КТП и функциональных подсистем с использованием средств автоматизированной разработки.

Итог - создание работоспособной АСУ КТП.

Вариант №1	Вариант №2	Вариант №3	Вариант №4	Вариант №5
АРМ отдела автоматизации	АРМ руководителя	АРМ конструктора	АРМ технолога	цеховой АРМ
1. Общесистемное меню доступа к базе данных 2. Модули администрирования (загрузка новых модулей, пользователей, контроль версий, управление правами доступа, управление меню, почтовая система, работа со справочной информацией, WEB технологии)	1. Модуль просмотра хода выполнения проекта. 2. Модуль управления качеством (прогноз и принятие решений) 3. Модуль управления персоналом и бухучета 4. Модуль формирования отчетности	1. Модуль управления конструкторским проектирование 2. Модуль загрузки/выгрузки КД (файлы *.dwg и т.п.) 3. Модуль формирования отчетности по конструкторскому проектированию	1. Модуль управления технологическим проектированием 2. Модуль загрузки/выгрузки ТД (файлы *.dwg и т.п.) 3. Модуль формирования отчетности по технологическому проектированию	1. Модуль управления и контроля за техпроцессом (маршрутные карты, сроки, эксплуатация оборудования и т.п.) 2. Модуль складского учета (инструменты, запчасти, комплектующие, полуфабрикаты и готовые изделия)

Пример анализа результатов этапа разработки логической модели (создания таблиц БД) (нормализация и оценка возможности опимизации структуры базы и формирования отчетности):

1. Целесообразно объеденить таблицы ASU_SHEMA_DOCS и ASU_KONSTR_DOCS в одну таблицу введя дополнительное поле признака документа (конструкторский, схемотехнический и т.п. При больших объемах обрабатываемых документов целесообразно ввести различные таблицы, например по годам, а формирование данных обеспечить посредством View, в которую включать данные за конкретный год, определяемый по параметру.
2. Целесообразно для хранения всех чертежей создать отдельную таблицу, в которой будет храниться не только сами файлы чертежей, но и дополнительные данные (дата создания, подробные комментарии и т.п.) - это позволит организовать контроль за версиями проекта, т.е.отслеживать динамику стадий проекта.
3. Целесообразно провести нормализацию таблицы пользователей, т.е. выделить содержание поля "должность" в отдельную таблицу - справочник

должностей, это позволит заводить различные должности без привязки к пользователям и обеспечит единообразие отражаемых должностей.

Перечнь основных таблиц БД

1. Таблица пользователей ASUKTP_USER

USER_NNN	Ф.И.О. пользователя	Ораклическое имя	Ссылка на подразделение	Ссылка на должность	Паспортные данные

1. Справочник подразделений ASUKTP_PODR

PODR_NNN	Наименование подразделения	Ссылка на подразделение высшего уровня	Контактная информация

3. Штатное расписание

SHTAT_NNN	Наименование должности	Ссылка на подразделение	Оклад по должности

1. Таблица управления проектами

PROEKT_NNN	Наименование проекта	Описание проекта	Ссылка на руководителя

1. Таблица схемотехнических документов

SHEMA_NNN	Наименование документа	Описание документа	Ссылка на NNN проекта	Ссылка на разработчика	имя файла чертежа

5. Таблица конструкторских документов по сборочным единицам

K_SBED_NNN	Наименование сборочной единицы	Описание	Ссылка на NNN проекта	Ссылка на разработчика (подразделение)	имя файла чертежа

6. Таблица конструкторских документов по деталям

K_DETAL_NNN	Наименование детали	Описание	Ссылка на NNN сборочной единицы	Ссылка на разработчика (подразделение)	имя файла чертежа

7. Таблица графических документов

GRAFDOC_NNN	Наименование файла	Дата создания	Тип файла (расширение)	Ссылка на разработчика (подразделение)	Описание

8. Таблица технологических документов по сборочным единицам

T_SBED_NNN	Ссылка на наименование СБ единицы	Описание	Ссылка на NNN проекта	Ссылка на разработчика (подразделение)	имя файла чертежа

9. Таблица технологических документов по деталям

T_DETAL_NNN	Ссылка на наименование детали	Описание	Ссылка на NNN тех док. По сборочной единицы	Ссылка на разработчика (подразделение)	имя файла чертежа

10. Таблица управления производственным процессом

TP_CONTROL_NNN	Ссылка на техпроцесс	Ссылка на операцию	Ссылка на NNN проекта	Ссылка на разработчика	Отметка о выполнении

11. Справочник техпроцессов

TP_SPR_NNN	Наименование ТП	Описание

12. Таблица операций техпроцессов

TP_OPER_NNN	Ссылка на NNN техпроцесса	Описание операции	Ссылка на справочник оборудования	Ссылка на подразделение	Комментарии

Здесь представлены только бозовае таблицы АСУ КТП, в зависимости от вашего варианта (разрабатываемого модуля) перечень дополнительных таблиц, для конкретного модуля) должен быть создан на этапе проектирования структуры БД модуля АСУ КТП (этап 3).

Таблица управления проектами ASU_PROEKT_CONTROL

Уникальный ключ PROEKT_NNN	Наименование проекта PROEKT_NUMBER	Описание проекта PROEKT_COMMENT	Ссылка на руководителя PROEKT_USER_NNN
1	Проект №0011	Блок питания	1
2	Проект №0066	Плата ВЗУ	9
3	Проект №2011	Модуль памяти	11
4	Проект №0014	Блок контроля	1
5	Проект №0015	Кардиограф	1
6	Проект №4011	Кардиостимулятор	1
7	Проект №3011	Кассовый аппарат	1

Таблица схемотехнических документов ASU_SHEMA_DOCS

Уникальный ключ SHEMA_NNN	Наименование документа SHAMA_NAME	Описание документа SHEMA_COMMENT	Ссылка на NNN проекта SHEMA_PR_NNN	Ссылка на разработчика SHEMA_USER_NNN	имя файла чертежа SHEMA_GRAPH
1	0011-ПС001	ПС цепи питания	1	4	ps11-1
2	0011-ПС002	ПС цепи земли	1	4	ps11-2
3	0014-ПС001	ПС блока конт.	4	4	ps14-1
4	0015-ПС001	ПС цепи питания	5	5	ps15-1
5	0015-ПС002	ПС вх. Цепи	5	5	ps15-2
6	0015-ПС003	ПС вых.. цепи	5	5	ps15-3
7	0015-ПС004	ПС индкации	5	5	ps15-4
8	3011-ПС001	ПС цепи питания	7	4	ps3-11-1
9	3011-ПС002	ПС циф. Обр.	7	4	ps3-11-2

Таблица конструкторских документов по сборочным единицам ASU_KONSTR_DOCS

Уникальный ключ KDOCS_NNN	Наименование сб. единицы KDOCS_NAME	Описание KDOCS_COMMENT	Ссылка на NNN проекта KDOCS_PR_NNN	Ссылка на разработчика KDOCS_USER_NNN	имя файла чертежа KDOCS_GRAPH
1	3011-СБ001	Блок питания	7	2	sb3-11-1
2	3011-СБ002	Решающий блок	7	3	sb3-11-2
3	0015-СБ001	Блок питания	5	2	sb15-1
4	0015-СБ002	Блок обработки	5	3	sb15-2
5	0015-СБ003	Блок индикации	5	3	sb15-3

Таблица пользователей ASU_USER

Уникальный ключ USER_NNN	Ф.И.О. пользователя USER_FIO	Ораклическое имя USER_ORANAME	Ссылка на подразделение USER_PODR_NNN	должность USER_SHTAT_NNN	Паспортные данные USER_PASPORT
1	Иванов Иван Иванович	ivanov	1	руководитель проекта	XXX МЮ-1109
2	Петров Петр Петрович	petrov	2	начальник констр. Отд.	XX МВ-9109
3	Сидоров	sidorov	2	ведущий констр.	XXI ВЮ-

	Сидор Сидорович				1203
4	Иванов Петр Петрович	pivanov	3	начальник системотех. Отд.	XIX АЮ-2105
5	Петров Иван Петрович	ipetrov	3	инженер-системотехник	IXX МА-1114

Литература

1. DiasoftInfo / Корпоративный журнал компании DIASOFT. - М. 1999 г.
2. Материалы аналитической компании СПЛАН.
3. Е.Голенцова Три основных вопроса СУД. ОАО "Весть". 1998.
4. А. Громов Управление бизнес-процессами на основе технологии Workflow// Открытые системы , №1. 1997.
5. Р. Майкл Oracle 7.3. Энциклопедия пользователя: Пер с англ. - К.: Издательство "Диасофт". 1997. - 832 с.
6. Фаронов В.В., Шумаков П.В. Delphi 4. Руководство разработчика баз данных - М.: "Нолидж", 1999. - 560 с., ил.
7. С.Урман Oracle 8. Программирование на языке PL/SQL - М.: Изд-во ЛОРИ, 1999. - 607 с.
8. К. Луни Oracle 8. Настольная книга администратора. - М.: Изд-во ЛОРИ, 1999. - 500 с.
9. С.Бобровски Oracle 8. Архитектура. - М.: Изд-во ЛОРИ, 1999. - 207 с.
10. Г.Хансен, Д.Хансен Базы данных: разработка и управление: Пер. с англ. - М.: ЗАО "Издательство БИНОМ", 1999. - 704 с.: ил.
11. С.Дунаев Доступ к базам данных и техника работы в сети. Практические приемы современного программирования. - М.: ДИАЛОГ-МИФИ, 1999 - 416 с.
12. Р.Петерсен Linux: руководство по операционной системе: в 2 т: Пер с англ. - 2-е изд., перераб. и доп. - К. Издательская группа BHV, 1998.
13. Власов А.И. Технология создания WEB узлов / Конспект лекций - М.: УЦ ОАО Газпром, 1999. - 102 с.
14. Власов А.И., Овчинников Е.М. Банковские и корпоративные автоматизированные информационные системы, принципы, средства и системы документооборота коммерческого банка / Конспект лекций. - М.: Учебный Центр ОАО Газпром, 1999. - 107 с.
15. Овчинников Е.М. Корпоративные информационные системы и технологии / Конспект лекций - М.: Учебный Центр ОАО Газпром, 1999. - 78 с.
16. Материалы периодической печати: Компьютерпресс, "Мир ПК", "Интернет", "Мир интернет", Byte Россия, PC magazine (RE).
17. http://www.oracle.ru http://www.diasoft.ru http://www.vest.msk.ru http://info.iu4.bmstu.ru http://cdl.iu4.bsmtu.ru http://www.microsoft.com http://www.ibm.com http://www.citforum.ru
18. Материалы выставок Comtec, NetCom, UnixExpo, Информатика и др.
19. Материалы 3-го, 4-го и пятого форумов разработчиков АБС.