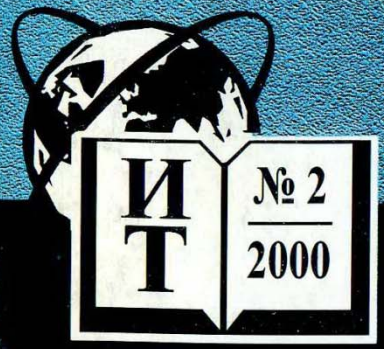


БИБЛИОТЕЧКА ЖУРНАЛА
"ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ"

А.И. Власов
С.Л. Лыткин
В.Л. Яковлев

Краткое практическое руководство разработчика по языку PL/SQL

БИБЛИОТЕЧКА ЖУРНАЛА
“ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ”

А.И. Власов
С.Л. Лыткин
В.Л. Яковлев

Краткое практическое руководство разработчика по языку PL/SQL

Москва
«Машиностроение»
2000

УДК 004.65

ББК 32.973

Х19

Власов А.И., Лыткин С.Л., Яковлев В.Л.

Краткое практическое руководство разработчика по языку PL/SQL - М.: "Машиностроение", 2000. 64 с. (Библиотечка журнала "Информационные технологии")

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. Основные принципы работы с синтаксическими конструкциями языка и элементами словаря данных проиллюстрированы на конкретных практических примерах. Руководство рассчитано на разработчиков прикладных информационных систем под СУБД Oracle, студентов, аспирантов и преподавателей, специализирующихся в разработке информационных систем и баз данных.

Изложенные в руководстве 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.

1. ВВЕДЕНИЕ В ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ

1.1 МЕТОДЫ ПРОЕКТИРОВАНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

Индустрия разработки автоматизированных информационных систем управления возникла 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. Несмотря на имеющиеся различия в реализации функциональных модулей данных систем, общие подходы к их разработке 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 можно выделить два основных субъекта: рынок АБС и рынок корпоративных информационных систем промышленных предприятий. Несмотря на сильную взаимосвязь этих двух рынков систем автоматизации, предлагаемые на них решения пока еще недостаточно интегрированы между собой, однако следует ожидать, что в недалеком будущем это положение изменится.

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 управления конструкторско-технологическим проектированием*** (АСУ КТП) будем понимать комплекс аппаратно-программных средств, реализующих мультикомпонентную информационную систему, обеспечивающую современное управление процессами принятия решении, проектирования, производства и сбыта в режиме реального времени при транзакционной обработке данных.

Как видно, оба определения достаточно схожи. Среди методов построения автоматизированных информационных систем (АИС), можно выделить следующие.

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 сохранился при автоматизации и до сих пор. 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. Вместе с тем, 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 остудили пыл как тех, так и других. Итак, первый подход сводился к проектированию ***"снизу - вверх".*** В том 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.

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, но не конкретных станков и других изделий, 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 потребностям 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 очевидным, что для успешной реализации задачи полной автоматизации банка следует изменить идеологию построения АИС.

Принципы "дуализма" и многокомпонентности. Развитие банковских структур и промышленных предприятий, увеличение числа филиалов, рост числа 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 привело к тому, что с одной стороны, в ядре системы сохранялась возможность работы "от лицевого счета", с автоматическим формированием соответствующих бухгалтерских проводок, а 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. В результате стало возможным:

- 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.

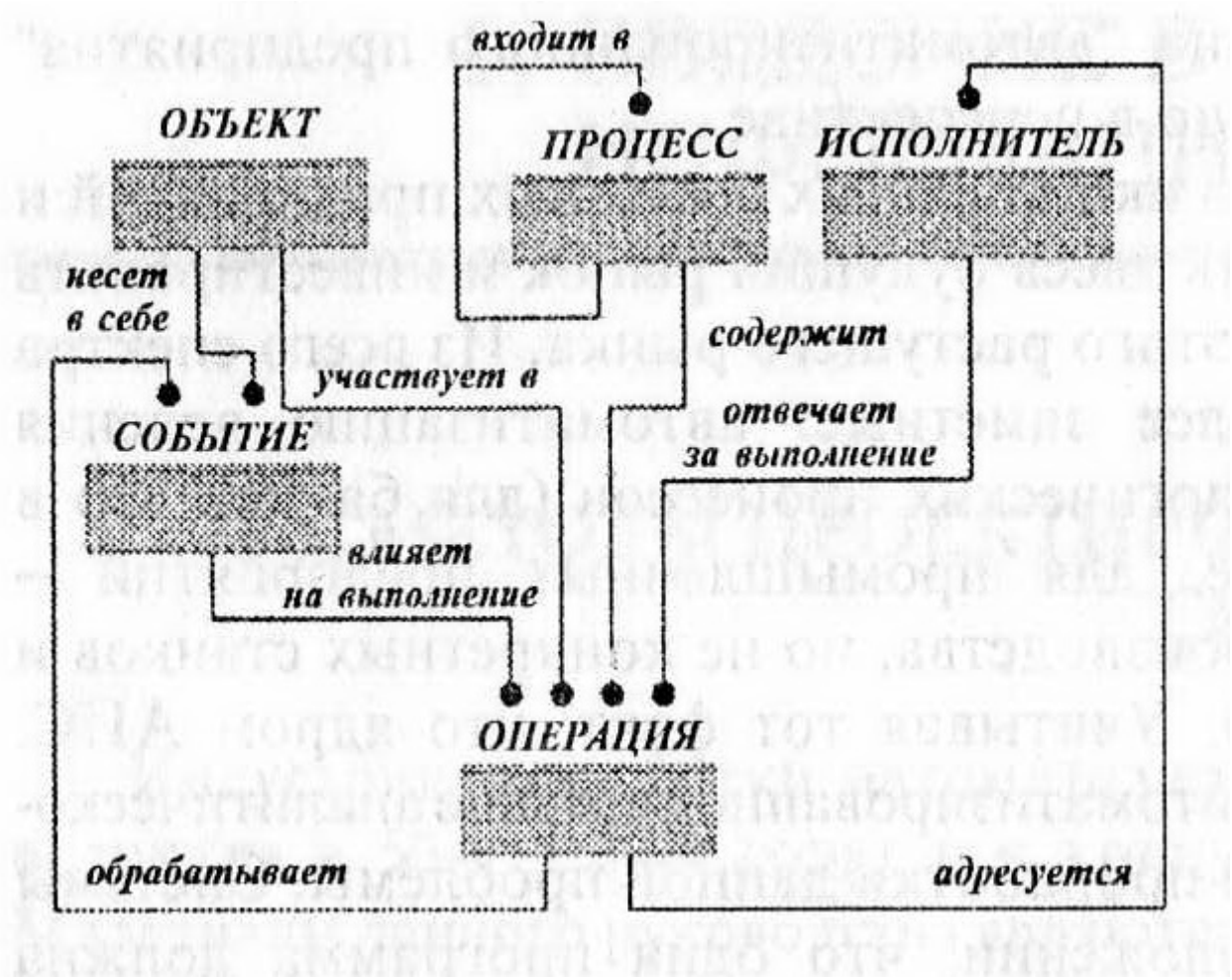


Рис.1. Концептуальная информационная модель технологии Workflow

Двойственный подход к формированию ежедневного баланса лег в основу так называемого "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.

Итогом стал переход от системы автоматизации документооборота (docflow) к системам автоматизации бизнес-процессов (технологии workflow) (рис. 1).

Задача проектирования АИС промышленных предприятий более сложна, так как 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 цифробуквенного обозначения характеризует деталь, сборочную единицу, изделие и их уровень взаимосвязи). Причем конкретная часть кода характеризует технологические, конструкторские, финансовые и другие документы. Все это регламентируется соответствующими ГОСТами (аналог инструкций ЦБ для банков), поэтому может быть

формализовано. Модульный подход к реализации АИС в этом случае еще более важен.

Двойственный подход к формированию ежедневного производственного плана лег в основу принципа дуализма для АИС промышленных предприятий. Реализация этого принципа неизбежно потребовала 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 оправдана. В этих случаях в филиалах возможна эксплуатация программного обеспечения (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 вычислительной сети

и средств связи, экспертно-аналитическое управление (постановка задач), добавляется лишь новая структура - управление развития и разработки АИС, что обычно 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], а именно:

- правильный выбор архитектуры построения вычислительно-коммуникационной сети и ориентация на профессиональные СУБД. По экспертным оценкам собственные разработки АИС базируются на СУБД Oracle (53 %), на Informix (около 15 %), других СУБД (22 %);
- использование при разработке современного инструментария (CASE средства, эффективные средства разработки: Delphi, Designer2000, Developer2000, SQL-Stations и т.п.);
- использование мультизадачной инфраструктуры разработки проекта, когда конкретный модуль АИС ведет группа разработчиков с взаимосвязанным перечнем задач; эта структура построена на принципах полной взаимозаменяемости, т.е. функционирование данного модуля АИС и его развитие не связано с одним конкретным разработчиком;
- 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. 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 недешевое. 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 возникает необходимость, произойдет 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 одного из основных компонентов стратегического развития бизнеса. В связи с этим в недалеком будущем можно ожидать расширения рынка корпоративных информационных систем и последующего его значительного роста. Учитывая тесную интеграцию финансовых и промышленных структур, можно полагать, что основой построения корпоративных систем финансово-промышленных групп будут являться АБС, используемые в финансовых учреждениях.

1.2 ОРИЕНТАЦИЯ НА ПРОФЕССИОНАЛЬНЫЕ СУБД - "ЗА" И "ПРОТИВ"

По материалам периодической печати можно судить, что 1998 год стал годом перехода к внедрению АБС четвертого поколения, основой которых, в свою очередь, является ориентация на профессиональные СУБД. Такая ориентация позволяет:

- использовать оптимизированный многопользовательский режим работы с развитой системой транзакционной обработки, что обеспечивает многочисленным пользователям возможность работы с базой данных, не мешая друг другу;
- применять надежные средства защиты информации (учитывая стандартную трехзвенную архитектуру защиты на уровне сети - на уровне сервера БД - на уровне клиентской ОС);
- использовать эффективные инструменты для разграничения доступа к БД;
- осуществлять поддержку широкого диапазона аппаратно-программных платформ;
- реализовывать распределенную обработку данных; осуществлять построение гетерогенных и распределенных сетей;
- использовать развитые средства управления, контроля, мониторинга и администрирования сервера БД;
- осуществлять поддержку 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. Приведем основные "мнимые" стереотипы, не позволяющие пока этим структурам ориентироваться на использования профессиональных СУБД при построении своих АИС [1, 10]:

- **"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 Work Group 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 выпустило свои версии и под Intel-серверы, в том числе и под ОС LINUX, а учитывая что LINUX при всей своей мощности UNIX-системы - практически бесплатная 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 серверы, но и пользовать серверные Intel-платформы;
- **"ПРОТИВ"** - профессиональные АИС сложны и дороги в администрировании;
- "ЗА" - как правило, сложность администрирования зависит от конкретной АИС. Кроме этого, эксплуатация 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, требующий 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 не относится к отечественным системам нового поколения, изначально рассчитанным на изменчивое российское законодательство.

Выводы. Анализ рынка показывает, что в настоящее время 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 Fetrieve все же трудно назвать масштабируемыми, а саму Betrieve - профессиональной СУБД, пригодной для построения корпоративной информационной системы. Betrieve-cucTeMbi унаследовали свою архитектуру и большую часть кода от своих предшественников, разработанных на Clipper и Clarion, что во многом объясняет столь большую популярность Betrieve среди фирм, разрабатывающих ранее под эти платформы. Действительно, механически перенести Clarion систему под использование менеджера записей Betrieve 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. 2.

Таблица 1

Этапы проектирования АИС и их характеристики

Наименование этапа	Основные характеристики
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**: • концептуальная модель АИС, состоящая из описания предметной области, ресурсов и потоков данных; перечень требований и ограничений к технической реализации АИС; • аппаратно-технический состав создаваемой АИС.
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, сгенерированны скрипты для создания объектов БД.
Выбор лингвистического обеспечения, разработка программного обеспечения АИС	Разработка АИС: выбирается лингвистическое обеспечение (среда разработки – инструментарий), разрабатываются программное и методическое обеспечение. 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.
Тестирование и отладка АИС	На данном этапе осуществляется корректировка информационного аппаратного, программного обеспечений, разрабатывается методическое обеспечение (документации разработчика, пользователя) и т.п. **Результат**: оптимальный состав и эффективное функционирование АИС; комплект документации разработчика, администратора, пользователя.
Эксплуатация и контроль версий	Особенностью 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, как показала практика, БД АИС за год эксплуатации может насчитывать более 1000 таблиц, из которых эффективно будут использоваться лишь 20–30 %. **Результат**: наращиваемость и безызбыточный состав гибкой, масштабируемой АИС.

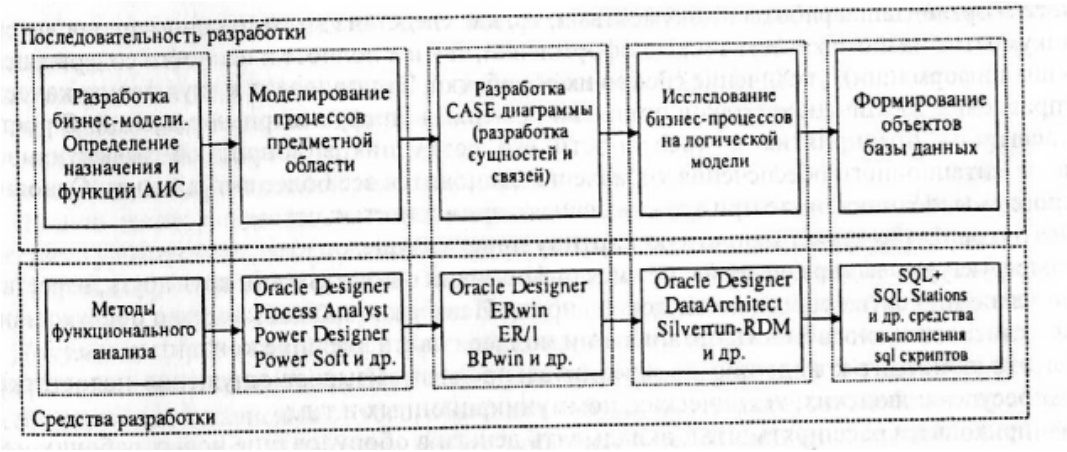


Рис. 2 Последовательность трансформации бизнес-модели в объекты БД и клиентские приложения

1.4. РАЗРАБОТКА И АНАЛИЗ БИЗНЕС-МОДЕЛИ

При построении эффективной автоматизированной системы первым этапом является исследование и формализация бизнес-процессов деятельности банка пли предприятия, т.е. описание системы ведения делопроизводства в целях эффективного использования информации для достижения поставленных задач и решения проблем, стоящих перед организацией. На рис. 3 в качестве примера представлена бизнес-модель заключения договора на обслуживание. 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;
- переработки 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.

Рис. 3. Бизнес-модель процесса заключения "договоров на обслуживание"

Это приводит к ухудшению качества управления, 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 уменьшения плановой прибыли.

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.**

Основные понятия электронного документооборота

Документ. Например, 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 оттого, обычная это бумага или электронный бланк, проблем корпорации не решает - первичны бизнес-процессы и четкий контроль за выполнением проекта.

Собственно документооборот может быть двух типов:

- *универсальный* - автоматизирующий существующие информационные потоки слабоструктурированной информации. Справедливо было бы его назвать 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].

Модели информационного пространства предприятия

Комплексная автоматизации перечисленных выше функций 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 желанным без потерь в производительности и независимо от своего местоположения в сети;
- использования средств коммуникации (например, электронная почта, факс, печать документов);
- сохранения целостности данных в общей базе данных;
- полнотекстового и реквизитного поиска информации;
- открытости системы, когда пользователи должны иметь доступ к привычным средствам создания документов и к уже существующим документам, созданным в других системах;
- защищенности 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.

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.

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, приемно-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.

Вторая ось (D) - полнотекстовые документы - отражает необходимость организации взаимодействия по формированию и передаче товаров, услуг или информации как внутри корпорации, так и вне ее. В этих документах наряду с фактографической информацией содержится слабоструктурированная информация, не подлежащая автоматизированной аналитической обработке, такая, например, как форс-мажорные 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 (FD, 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 заботиться 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. Однако 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 под новым качеством ошибочно 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.

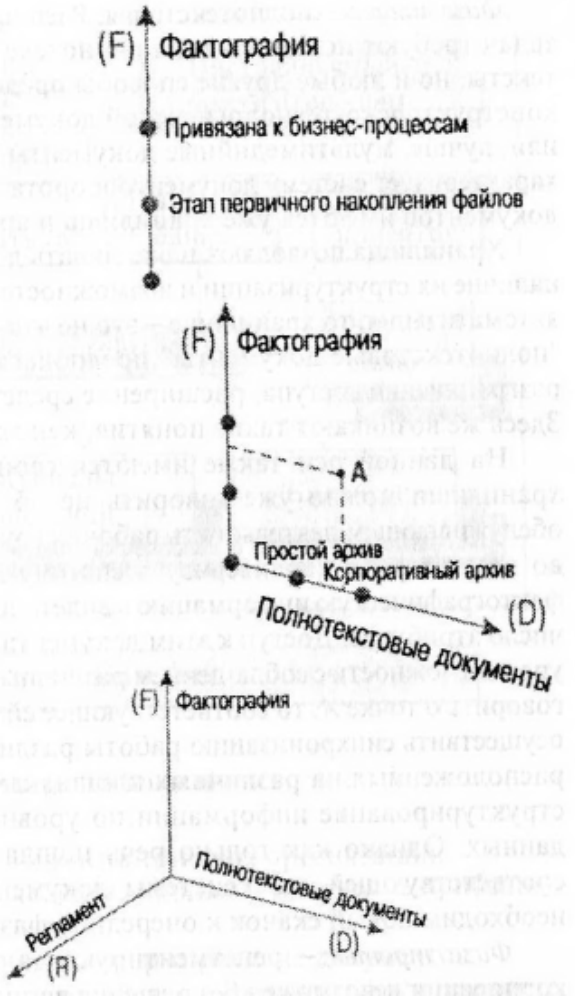


Рис. 4. Эволюция бизнес-моделей документооборота

Эволюции 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. 4.

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.

Точка на этой оси - это текущее состояние системы документооборота организации. Движение по оси вверх характеризует накопление фактографической

информации и, начиная с определенного момента, 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 число 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 осуществить 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), т.е. каждый сотрудник имел доступ к архиву или его части, либо в папку каждому работнику помещалось индивидуальное задание (иначе говоря, было известно только то, 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 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.

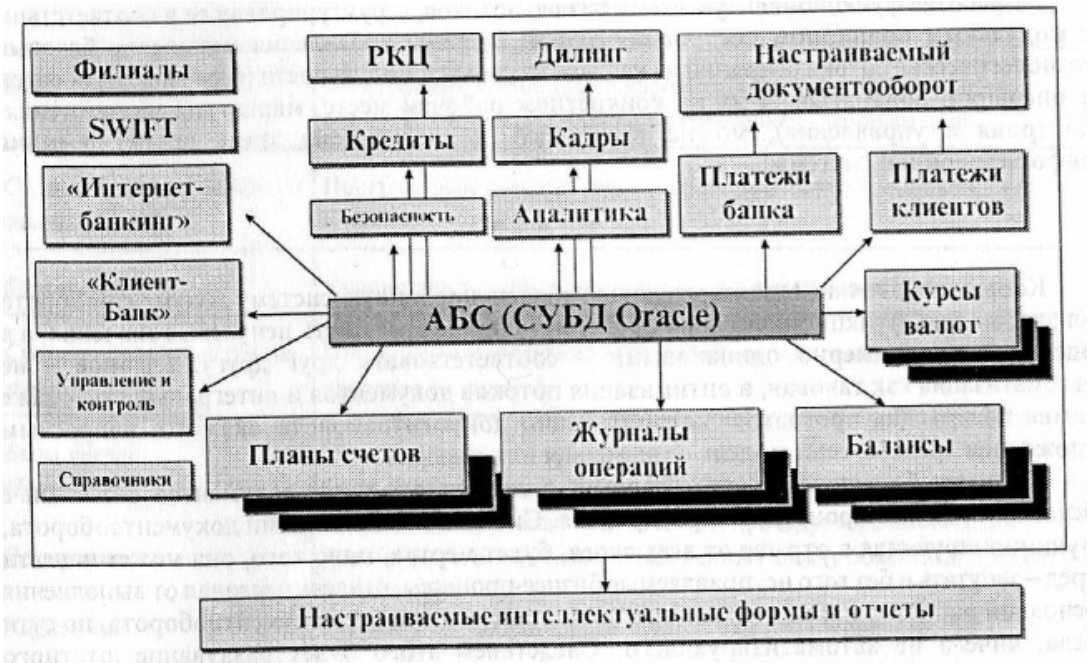


Рис. 5. Функциональная схема типовой АБС четвертого поколения

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 разработке функциональной модели типовой АБС. Среди архитектур реализации АБС в настоящее время выделяют АБС пяти поколений: первые были предназначены для решения задач автоматизации бухгалтерских операций, АБС второго и третьего поколений также реализовывали элементы автоматизированного документооборота, АБС четвертого поколения строились

по классической схеме трехзвенной клиент-серверной архитектуры и имели модульную структуру, реализующую концепции docflow. АБС пятого поколения являются информационными системами, реализующими полнофункциональную модель workflow-автоматизации бизнес-процессов.

Схема функциональных бизнес-потоков типовой АБС четвертого поколения представлена на рис. 5.

Назначения большинства модулей АБС (рис. 5) достаточно понятны, следует лишь уточнить, что под интернет-банкингом понимается автоматизированная система, реализующая технологии busines-to-customers (клиентское обслуживание, включая электронные платежи физических лиц) и bisnes-to-bisnes (обеспечение работы дистрибьютерско-дилерской сети корпораций в режиме on-line коммерции).

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

* * *

Координаты точки, характеризующей сбалансированную систему документооборота (бизнес-модель функционирования предприятия), должны иметь ненулевые значения, а в идеале быть примерно одинаковыми - 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.

Не имеет 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. Следствием этого будет 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 и 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.

2. ТЕХНОЛОГИИ СОЗДАНИЯ РАСПРЕДЕЛЕННЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

Построение современных распределенных информационных систем сегодня напрямую связано с реляционными 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 СУБД и других программ, выполняющих сходные функции. Инструментальные средства Oracle - одни из лучших и наиболее мощных имеющихся инструментов разработки профессионального класса.

2.1 МОДЕЛИ БАЗ ДАННЫХ И ИХ СРАВНИТЕЛЬНЫЕ ХАРАКТЕРИСТИКИ

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 СУБД, основанные на трех базовых моделях данных: иерархической, сетевой и реляционной. Основные определения теории баз знаний и баз данных представлены в табл. 2.

Таблица 2

База данных (БД)	Электронные хранилища информации, доступ к которым осуществляется с помощью одного или нескольких компьютеров
Системы управления базами данных (СУБД)	Программные средства для создания, наполнения, обновления и удаления баз данных
База знаний (БЗ)	Хранилища знаний, представленных в формализованном виде
Система управления базами знаний (СУБЗ)	Программные средства для создания, наполнения, обновления и удаления баз знаний
Виды знаний: *Процедурные* *Декларативные* *Каузальные* *Неточные*	 Знания, отвечающие на вопрос "Как решать поставленную задачу?" Знания, не содержащие в явном виде процедуры решения задач. Знания о причинно-следственных связях между объектами предметной области. 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 взаимосвязанных 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.
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 COBRA	COBRA – 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 Sum 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.

Иерархическая модель

Первые иерархические и сетевые СУБД были созданы в начале 60-х годов. Причиной создания послужила необходимость управления миллионами записей, связанных друг с другом иерархическим образом (например, при информационной поддержке космического 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 и др.

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.

К ограничениям иерархической модели данных можно отнести:

- отсутствие явного разделения логических и физических характеристик модели;
- необходимость дополнительных манипуляций для представления неиерархических отношений данных;
- необходимость реорганизации базы данных из-за непредвиденных запросов.

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 - как 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 связанных отношениями мощности один - к одному или один - ко многим. В конце 60-х годов 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 использованная в одной из самых первых СУБД архитектура Iategrated Data Store (IDS), созданная ранее компанией General Electric. Это привело к тому, что была рекомендована сетевая модель.

Документы DTBG от 1971 г. остаются 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 известные коммерческие реализации.

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 Е.Ф. 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. 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.***

Существуют 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.

Основные определения реляционных СУБД приведены в табл. 3.

Таблица 3

Основные определения реляционных СУБД

Реляционная модель данных	Организует и представляет данные в виде таблиц или реляций.
Реляционная база данных (РБД, RDBMS)	База данных, построенная на реляционной модели.
Реляция (таблица – элементарная информационная единица)	Двумерная таблица, содержащая строки и столбцы данных.
Степень реляции	Число атрибутов реляции (необходимо помнить, что никакие два атрибута реляции не могут иметь одинаковых имен).
Кортежи	Строки реляции (таблицы), соответствуют объекту, 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 атрибутов строки пока неизвестны, то строка запишется в базу данных с пустыми значениями этих 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 таблице соответствует одна запись в дочерней таблице.
Отношение "один – ко многим"	Когда одной записи в родительской таблице соответствует несколько записей в дочерней таблице.
Отношение "многие – ко многим"	Когда многим записям в родительской таблице 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 или представлениям.
Ссылочная целлостность	Адекватное воспроизведение записей в ссылочных полях таблиц.
Индекс	Механизмы быстрого доступа к хранящимся в таблицах данным путем их предварительной сортировки.
Операция соединения (Join)	Процесс, позволяющий объединять данные из двух таблиц посредством сопоставления содержимого двух аналогичных столбцов.
SQL (Structured query language)	Язык структурированных запросов, язык SOL. 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, или иначе: такое воздействие на СУБД, которое переводит ее из одного целостного состояния в другое. 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.
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, т.е. они являются средством обеспечения ссылочной целостности на основе механизма каскадных изменений.

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, что делает информацию в базе данных недостоверной. Для предотвращения потери ссылочной целостности используется механизм каскадных изменений (который чаще всего реализуется специальными объектами СУБД - триггерами). Данный механизм состоит в следующей последовательности 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 таблице следует удалить соответствующие записи и в дочерней таблице.

Процесс нормализации. *Нормализация* - процесс приведения реляционных таблиц к стандартному виду. В базе данных могут присутствовать следующие проблемы:

- *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 *разбиение таблиц* - разделение одной таблицы на несколько. 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 Е.Ф. 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 от части ключа. Таким образом, 2НФ может оказаться нарушена только в том случае, когда ключ составной.

Третья нормальная форма. Реляционная таблица имеет *третью нормальную форму* (ЗНФ), если для любой функциональной зависимости X - YX является ключом. Заметим, что любая таблица, удовлетворяющая ЗНФ, также удовлетворяет и 2НФ. Однако обратное неверно.

Критерий нормальной 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, т.е. этот критерий учитывает следующие два случая:

- неключевой атрибут зависит от ключевого атрибута, входящего в составной ключ (критерий нарушения 2НФ);
- 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. Эта 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.

Преобразование функциональной модели в реляционную. В разделе 1.3 были описаны основные этапы разработки автоматизированной информационной системы, в разделе 1.3.1 - функциональная модель АИС. Результатом первого этапа проектирования АИС является функциональная модель системы, содержащая множество объектов (процессов, операций), их атрибутов. Теперь, после того как мы рассмотрели основные положения теории баз данных, пришло время заняться непосредственно формализацией выделенных бизнес- процессов, операций и т.п.

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.

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. 6. (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).

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 дадим 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 CLS, то все таблицы, входящие в данную систему, было бы целесообразно называть CLS_<имя модуля>_<имя таблицы>;
- для столбцов:

<Псевдоним таблицы>_<имя столбца>.

Например, как показано на рис. 6, регистрационный номер пациента хранится в поле PT_REG_NUMBER таблицы PATIENTS, имеющем псевдоним PT.

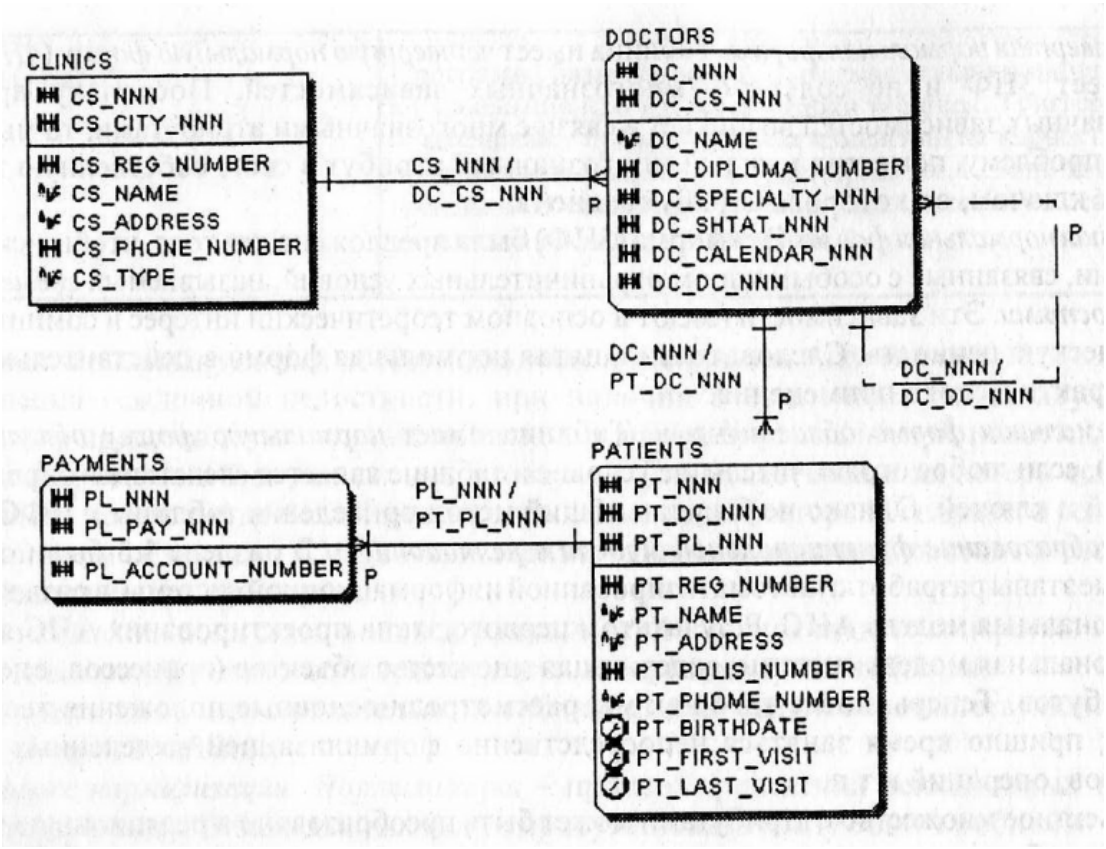


Рис. 6. Пример реализации логических отношении данных "ДОКТОР - ПАЦИЕНТ"

Конечно, использование этих нехитрых правил не является обязательным, но позволяет значительно облегчить читаемость разработанной информационной структуры. Предположите, как было бы все, если бы поля таблиц назывались R111, R112 и т.п., а ведь такие вещи встречаются практически очень часто, например в FoxPro 2.6.

Рис. 7. Типы логических отношений

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. 3. Рассмотрим преобразование отношений на примере АИС "ДОКТОР - ПАЦИЕНТ" (рис. 6).

Отношение **"один-к-одному"** представляет собой такое отношение, при котором каждой записи в одной таблице соответствует единственная запись в другой таблице (рис. 7, а). 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. 6 представлено, как с использованием отношения "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.

Отношение **"один-ко-многим"** можно без преувеличения назвать основным типом отношений, использующимся при проектировании современных БД, так как позволяет представлять иерархические структуры данных. Под данным отношением 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. 7,б). Отношения “один-ко-многим" могут быть жесткими и нежесткими. Для жестких отношений необходимо выполнение требований, что каждой записи в родительской таблице должна соответствовать хотя бы одна запись в дочерней таблице.

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 таблице (рис. 7, в). Использование такого типа отношений крайне ограничено, что объясняется не только тем, 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".

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. 7, г, 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 что Сидоров В.Н. является по отношению к ним начальником и т.п. Данный тип отношений позволяет реализовать древовидную структуру функциональных отношений, например структуру организации.

Учитывая требования ссылочной целостности и нормализации на основе применения рассмотренных выше типов отношений, можно осуществить преобразование функциональной модели бизнес-процессов в реляционную модель. Итогом этапа является диаграмма "Сущность - связь" (часто называемая CASE-диаграмма, ER-диаграмма).

* * *

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 Internet/Intranet. 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.

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, причем ресурсы сервера используются 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, 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.

В последнее время все большее развитие получает архитектура Internet/Intranet. 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 апплетов 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);
- доступность прикладных программных средств 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 реализовать системы с тысячами пользователей (причем многие из

них могут работать за удаленными терминалами).

В общем случае АИС, реализованная с использованием данной архитектуры, включает Web-узлы с интерактивным информационным наполнением. Эти узлы реализованы с помощью технологий Java, JavaBeans, взаимодействующих с предметной базой данных, с одной стороны, и с клиентским местом - с другой.

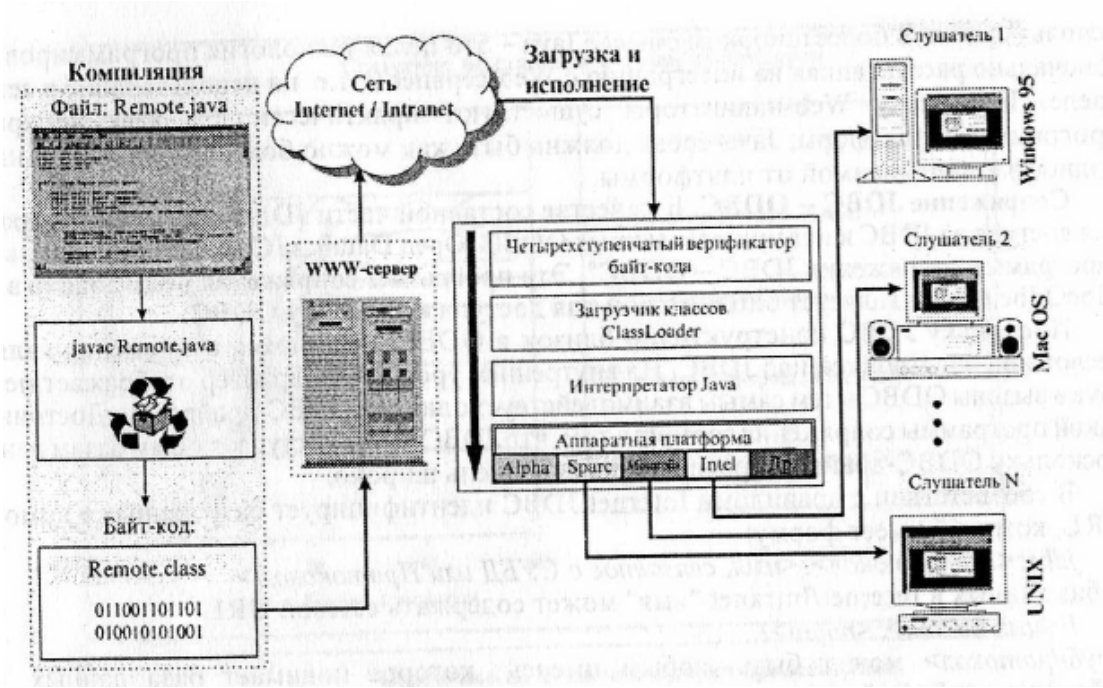


Рис. 8. JAVA-технологии при реализации АИС

На сегодняшний день известны и широко применяются три основные технологии создания интерактивного взаимодействия с пользователем в Web. Первый способ заключается в использовании Стандартного Интерфейса Шлюза (Common Gateway Interface) - CGI, второй - во включении JavaScript-сценариев в тело Web-страниц. И, наконец, самый мощный, предоставляющий практически неограниченные возможности способ - применение технологии Java (использование Java-апплетов).

CCI - 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. Взаимодействие между броузером сервером и CGT осуществляется по алгоритму:

1. Отсылка содержания формы на сервер.

2. Переправка содержимого процессу Web-сервер.

3, 4. Пересылка данных формы через CGI.

5, 6. Анализ и обработка данных, динамическое формирование Web-страницы,

переправка ее Web-серверу.

7, 8. Возвращение результатов клиенту в виде готового HTML-документа.

Имеются в виду узлы, предоставляющие доступ к базам данных, средствам поиска, и даже информационные системы, передающие 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 осуществить. Данный интерфейс можно считать посредником между броузером, сервером и любой информацией, которая должна передаваться между ними.

Наибольшую популярность CGI-сценарии нашли при использовании в качестве обработчиков форм, средства доступа к базам данных, средства осуществления локального и глобального поисков, шлюзовых протоколов.

Наибольшей мощью в реализации клиентского программного обеспечения обладают апплеты-программы, написанные на языке JAVA (рис. 8). 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, т.е. 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.

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 в 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 с помощью 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" зарезервировано 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

Чтобы отработать информацию из базы данных, информационно-обучающая система на языке Java выполняет ряд шагов. На рис. 15 показаны основные объекты JDBC, методы и последовательность выполнения. Сначала 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 число рядов, затронутых оператором SQL.

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

- отсутствие многих популярных приложений и средств разработки, реализованных в виде JAVA-апплетов;
- относительное высокое время компиляции апплетов на клиентских местах (временно);
- вопросы безопасной работы в сети.

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, приведем здесь лишь некоторые обобщения материалов тестирования.

При проведении испытаний (рис. 9) 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).

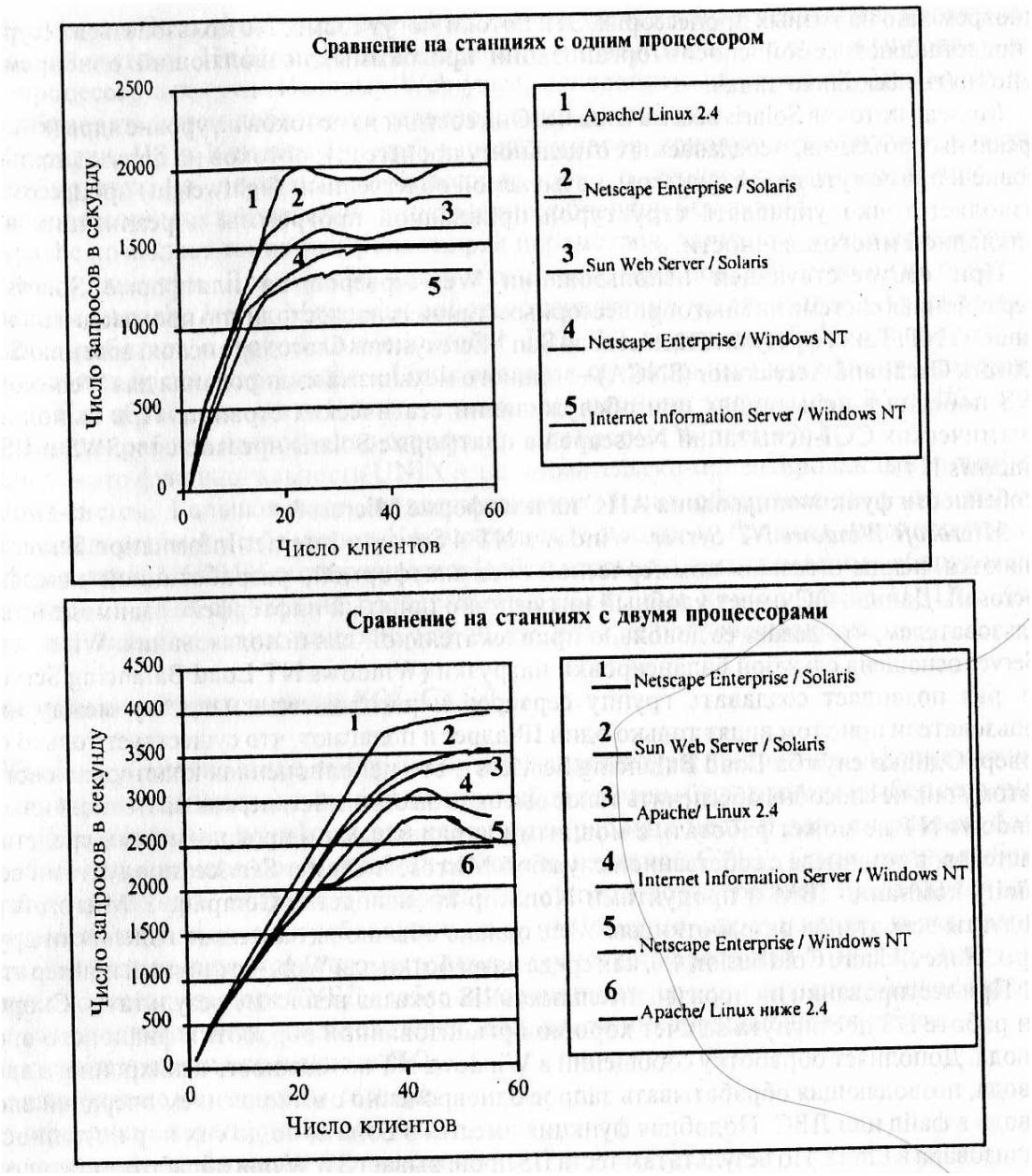


Рис. 9. Сравнительные характеристики серверных платформ

Все платформы были испытаны с помощью новой версии эталонного теста WebBench отделения Ziff-Davis Benchmark Operation.

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.

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.

При соответствующем использовании Web-серверов на платформе Solaris эта операционная система на многопроцессорных станциях превосходит по производительности WindowsNT. 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.

Особенности функционирования АИС на платформе Microsoft.

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 понятый интерфейс взаимодействия с пользователем, что делает ее 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.

При тестировании на производительность IIS показал неплохие результаты.

Скорость при работе IIS достигнута за счет хорошо организованной обработки 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.

Особенности функционирования АИС на основе Linux. Все больше растет популярность Linux и ее респектабельность как платформы разработки для Web и корпоративных сред. Linux характеризуется рядом преимуществ, таких как широкое сообщество разработчиков открытого кода и поддержка многих моделей комплектующих. 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, а следовательно, и 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, таких как число процессоров, доступных клиентам. Для Apache, как и для других серверов, есть механизм работы с сервлетами (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 платформе Windows 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 совместим с оборудованием, которое в ряде случаев не поддерживают некоторые коммерческие UNIX). Хотя некоторые достаточно экзотические устройства пока не поддерживаются.

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. Очень важно выбрать такую базу данных, которая в наибольшей степени соответствует 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:

- максимальное число пользователей, одновременно обращающихся к базе;
- характеристики клиентского ПО;
- аппаратные компоненты сервера;
- серверная операционная система;
- уровень квалификации персонала.

На сегодня известно большое число различных серверов баз данных SQL. Остановимся более подробно 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:

- конфигурирование системы;
- мониторинг;
- настройка;
- обработка запросов;
- 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- гигабайтным 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 наделен самым развитым набором функций для работы с языком Java, доступа к данным через Internet и системой оптимизации одновременного доступа. Единственным недостатком данной СУБД является сложность администрирования, однако все затраты на ее внедрение и освоение впоследствии 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. Так, 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:

- высочайшая надежность;
- возможность разбиения крупных баз данных на разделе (large-database partition), что дает возможность эффективно управлять гигантскими гигабайтными базами;
- наличие универсальных средств защиты информации;
- эффективные методы максимального повышения скорости обработки запросов;
- индексация по битовому отображению;
- свободные таблицы (в других СУБД все таблицы заполняются сразу при создании);
- распараллеливание операций в запросе;
- наличие широкого спектра средств разработки, мониторинга и администрирования;
- ориентация на Internet-технологии.

Решения, не уступающие разработкам Oracle, можно найти только в DB2 фирмы IBM. Ориентация на Internet-технологии - основной девиз современных продуктов Oracle.

В этой связи можно отметить пакеты InterMedia, обеспечивающие 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 не только составлять на языке Java внутренние программы для баз данных (хранимые процедуры и триггеры), но и разрабатывать компоненты Enterprise JavaBeans и даже запускать их на сервере). Компоненты Enterprise JavaBeans представляют собой базовые модули, из которых складываются Internet-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, а именно - очень развитые средства для обработки:

- аудиоклипов;
- неподвижных изображений;
- видеофрагментов;
- географических данных (с целым набором функций, связанных с определением местонахождения).

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 позволяет 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 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:

- простота администрирования;
- возможность подключения к Web;
- быстродействие и функциональные возможности механизма сервера СУБД;
- наличие средств удаленного доступа.

В комплект средств административного управления данной СУБД входит целый набор специальных мастеров и средств автоматической настройки параметров конфигурации. 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 базу данных, которая идеально подходит для малых и средних организаций. Необходимо заметить, что SQL Server уступает другим рассматриваемым СУБД по двум важным показателям: программируемости и средствам работы. При разработке клиентских БД-приложений на основе языков Java, HTML часто возникает проблема недостаточности программных средств SQL Server, поэтому пользоваться этой СУБД 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. В связи с этим 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 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 полным набором мультимедиа расширений, позволяющих сохранять текст, звук, видеофрагменты, 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 приложения, так как

существует средство Stored Procedure Builder - автоматически преобразующее 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.

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 Internet-систем. Предположительно данная СУБД будет обладать гибкой средой разработки, обладающей наращиваемостью, соответствующей характерным для Internet 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.

* * *

Рассмотрены основные характеристики архитектур построения АИС, серверных операционных систем и СУБД в дальнейшем в качестве архитектуры АИС мы выберем архитектуру Internet/intranet в качестве серверной ОС - Linux, в качестве СУБД - Oracle8i. В сводной табл. 4 представлены сравнительные характеристики двух наиболее распространенных на сегодня решений на базе Microsoft SQL Server 7.0 (на NT) и Oracle8i (на Unix, Linux).

Таблица 4

Сравнительные характеристики Microsoft SQL Server и Oracle8i

Характеристика	*Microsoft SQL Server*	*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-данных	Плохо	Отлично
Подключение к Web	Плохо	Отлично
Поиск по всему тексту	Хорошо	Отлично
Функциональная совместимость	Хорошо	Приемлемо
Сопряжение с другими БД	Хорошо	Плохо
Работа под управлением различных ОС	Приемлемо	Хорошо
Возможности программирования	Приемлемо	Отлично

Характеристика	*Microsoft SQL Server*	*Oracle8i*
Хранимые процедуры и триггеры	Хорошо	Отлично
Внутренний язык программирования	Плохо	Отлично
Объектно-ориентированные системы	Плохо	Отлично
Тиражирование	Отлично	Отлично
Распределенная обработка транзакций	Отлично	Отлично
Дистанционное администрирование	Хорошо	Отлично
Средства загрузки	Отлично	Отлично
Средства анализа	Отлично	Хорошо

Клиентские места при этом могут функционировать практически на любой платформе, средством доступа WEB-клиентов к СУБД является либо CGI (Perl) либо JAVA-приложения.

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: 24 ч*60 мин*60 с=86400 с в сутки, если каждый посетитель берет с сервера по 10 документов (*.html+графика), то при равномерном распределении загрузки получается три 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 обращений в секунду. Для нормальной работы такому серверу необходимо около 400 Мбайт RAM, при хорошей настройке - не менее 200.

При правильной конфигурации сервера не рекомендуется использовать swap. Все должно помещаться в оперативной памяти (имеется в виду, что у сервера swap-область быть должна, но она обязана быть пустой). Для предотвращения перегрузок рекомендуется пользоваться несколькими правилами, снижающими нагрузку сервера (придерживаясь этих правил, можно сэкономить около 30 % ресурсов сервера):

- для статической информации всегда ставить last-modified в атрибут выдачи CGI- скриптов - документ без временного штампа не сохраняется в локальном кэше и постоянно перезаписывается при просмотре;
- CGI-программы хранить в любом каталоге кроме/CGI-BIN/, так как proxy-серверы не кэшируют файлы, находящиеся в этих каталогах, и каждый раз вынуждены обращаться к вам на сервер;
- не использовать анимированные *.gif, так как некоторые NT обращаются к серверу перед каждым циклом;
- не ставить баннеры сверху страницы, так как баннер сверху отнимает 1-2 реквеста из четырех и в итоге грузится вперед, тормозя ваши сайтовые картинки;
- при вызове баннера не обращаться каждый раз к CGI, а подставлять вместо случайного числа любое число, что можно сделать, например, получив дату на JavaScript;
- вызывать баннеры программами на Си, так как Perl работает медленнее;
- на одной машине должен размещаться только информационный сервер.

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.

2.3 РЕЛЯЦИОННАЯ МОДЕЛЬ КАК ПЛАТФОРМА ДЛЯ РАЗРАБОТКИ СОВРЕМЕННЫХ ИНФОРМАЦИОННЫХ СИСТЕМ НА ПРИМЕРЕ ИНТЕРАКТИВНОЙ СИСТЕМЫ ПАТЕНТНОГО ОБЕСПЕЧЕНИЯ ТЕХНОЛОГИЧЕСКОГО ПРОЕКТИРОВАНИЯ

Мы рассмотрели различные подходы к внутренней организации баз данных и пришли к выводу о необходимости использования реляционной модели, так как она решает одну из основных проблем - внесения изменений в БД в процессе ее использования. Ведь в реляционной БД проблемы синхронизации данных не возникает вовсе, так как данные хранятся в одном экземпляре (ключевой проблемой является синхронизация транзакций).

Основные черты реляционных баз данных:

- структура реляционной БД определяется хранящимися в них данными и не фиксируется в момент завершения разработки, т.е. является гибкой и масштабируемой;
- структурам данных можно давать весьма информативные названия;
- данные хранятся в единственном экземпляре; все опции чтения и модификации данных выполняются только с этим экземпляром данных, что качественно облегчает синхронизацию данных между многими приложениями и пользователями;
- данные хранятся в соответствии с четко определенными и строго соблюдаемыми правилами.

Исследование моделей информационного представления данных в современных СУБД

Исследование различных методов и средств представления управления данными в информационных системах проведем на примере интерактивной БД патентного обеспечения конструкторско-технологического проектирования. 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 МКИ, которая, являясь средством для единообразной в международном масштабе классификации патентных документов, представляет собой эффективный инструмент для патентных ведомств и других потребителей, осуществляющих поиск патентных документов в целях установления новизны и оценки вклада изобретения в заявленное техническое решение (включая оценку технической прогрессивности и полезности или результата).

МКИ состоит из следующих отделов: раздел, класс, подкласс, группа, подгруппа.

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. 10). Реализацией данной логической модели является совокупность 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 МКИ в информационной системе патентного обеспечения технологического проектирования.

Таблица P1B_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).

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-GUI).

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.

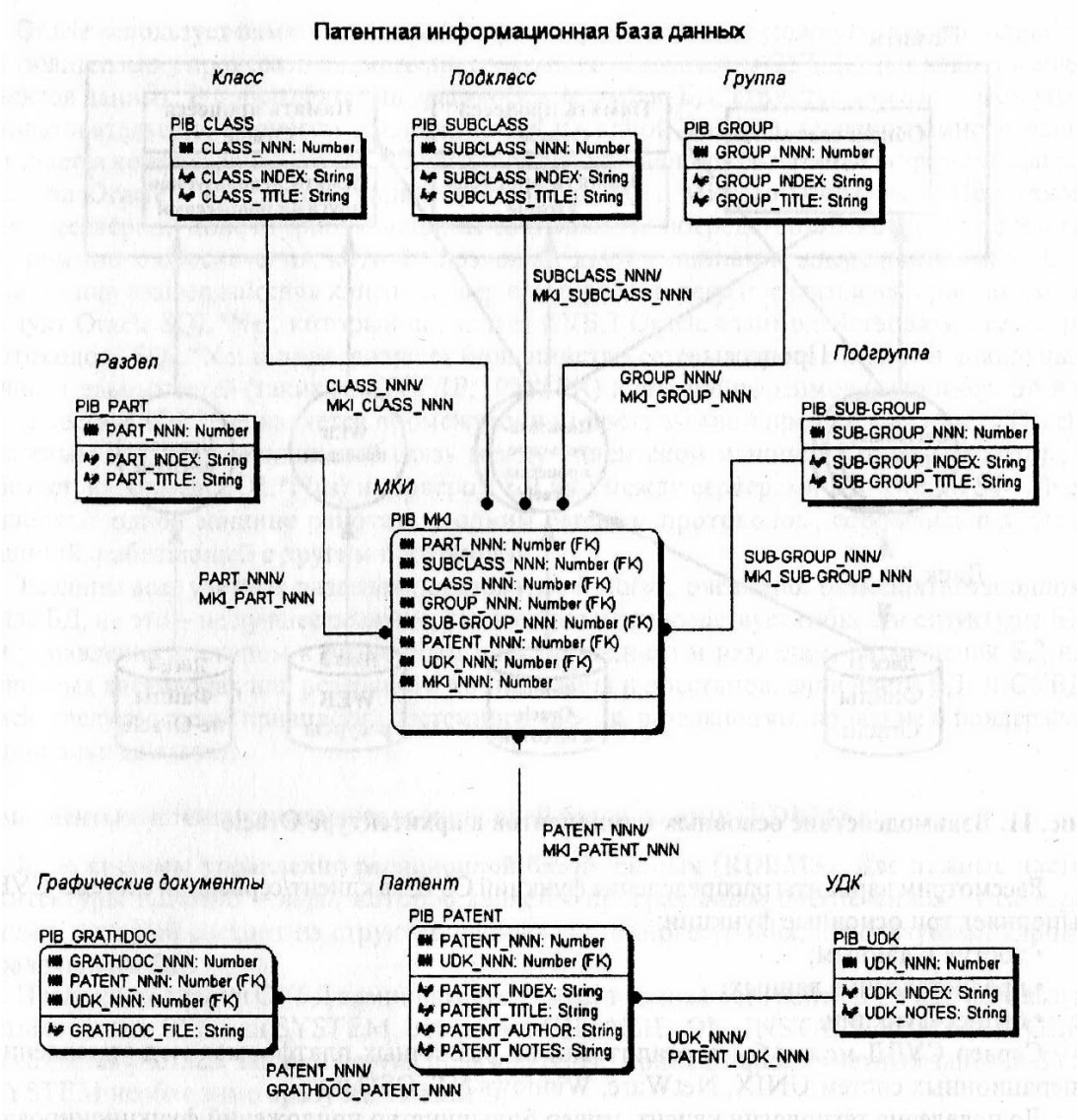


Рис. 10. Логическая модель АИС патентного обеспечения конструкторско-технологического проектирования

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

- доступ к данным;
- предоставление данных;
- 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 БД и логику приложений.

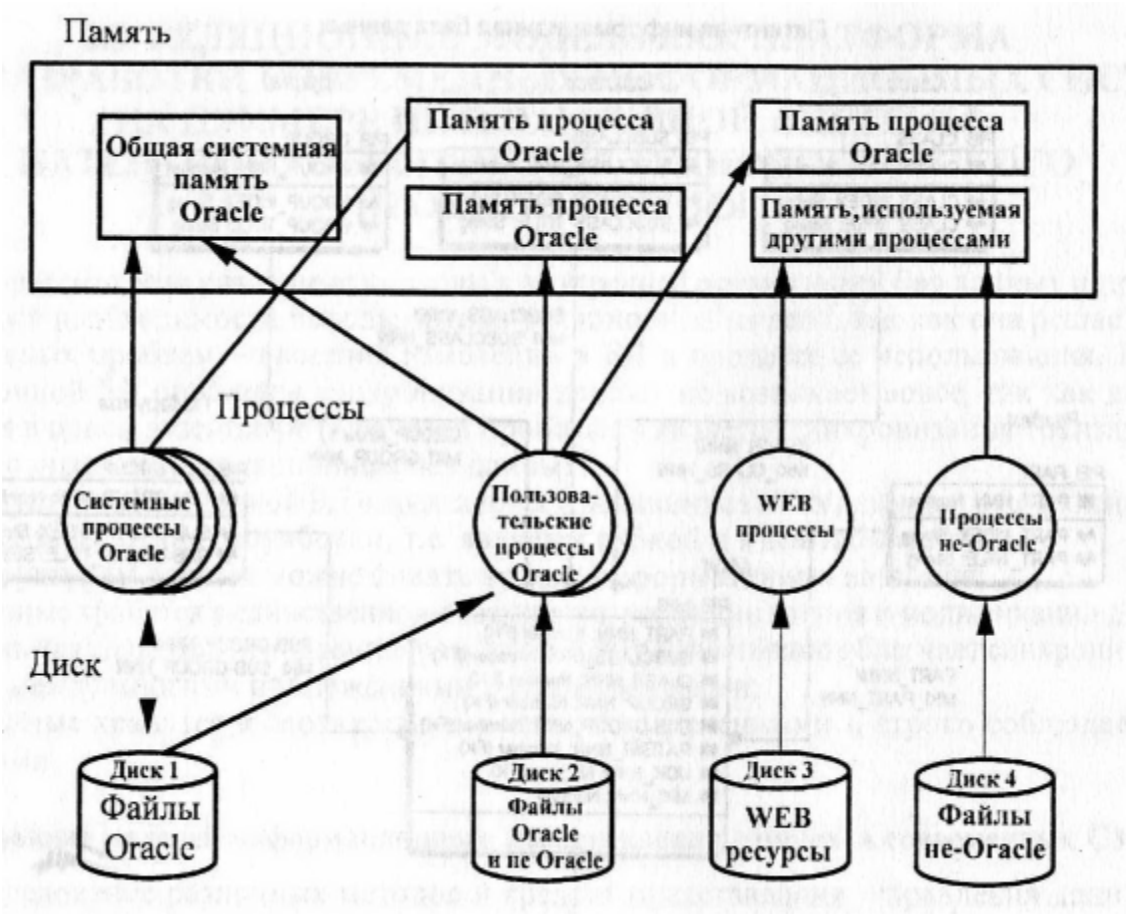


Рис. 11. Взаимодействие основных компонентов в архитектуре Oracle

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

Программные продукты Oracle охватывают все основные компоненты архитектуры клиент/сервер, показанной на рис. 11 [5-10]:

- полнофункциональный высокопроизводительный сервер RDBMS (система управления реляционной БД), масштабируемый от портативных ЭВМ до мэйнфреймов;
- средства для разработки и запуска клиентских приложений, поддерживающие несколько сред GUI;
- программный компонент для организации связи между БД на различных ЭВМ, который обеспечивает эффективную и безопасную связь с помощью широкого набора сетевых протоколов.

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 БД, структуры памяти, фоновые и пользовательские процессы располагаются на одной машине. 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 предназначен 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 БД или между серверами БД. Опции SQL*Net позволяют одной машине работать с одним сетевым протоколом, сообщаясь с другой машиной, работающей с другим протоколом.

Таблицы всех учетных разделов пользователей могут, очевидно, размещаться в одном файле БД, но это - не лучшее решение, так как оно не способствует гибкости структуры БД для управления доступом к различным пользовательским разделам, размещения БД на различных дисководах или резервного копирования и восстановления части БД. В СУБД Oracle предусмотрены привилегии системного уровня, резервное копирование и поддержка национальных языков.

Компоненты системы управления реляционной базой данных (RDBMS)

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 БД.

После инсталляции СУБД администратор имеет возможность войти в СУБД, используя учетные записи SYS или SYSTEM, с паролем CHANGE_ON_INSTALL или MANAGER для создания учетных записей других пользователей, причем пароль учетных записей SYS и SYSTEM необходимо сразу же изменить.

Типы обрабатываемых данных

Типы данных обрабатываемых СУБД Oracle представлены в табл. 5.

Таблица 5

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 для 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 Гбайт
LING RAW	Двоичные данные переменной длины вплоть до 2 Гбайт или 231-1

Тип данных	Описание
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 *me* для RAW обязателен
RAW MLSLABEL	Используется в Trusted ORACLE
ROWID	Значения псевдостолбца ROWID
VARCHAR2 (size)	Символьная строка переменной длины, имеющая максимальную длину *size* символ. Длина по умолчанию 1, максимальная 2000
VARCHAR (size)	То же что и VARCHAR2

Извлекать данные можно также и из псевдостолбцов (табл. 6), которые похожи на столбцы таблиц, но их значения нельзя изменять с помощью операторов DML.

Таблица 6

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 т.д. 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 байт, за исключением имен БД, длина которых ограничена 8 байт;
- 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 в двойные кавычки.

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 БД. 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 [7].

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 той же строке, имеют одинаковые приоритеты.

Замечание. В выражениях можно использовать круглые скобки, чтобы изменять последовательность выполнения операций, предписываемую приоритетом. Выражения, заключенные в скобки, ORACLE вычисляет 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, AND, 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 БД. 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 всю элегантность и мощь SQL. Все примеры, приведенные ниже, даны применительно к ER-диаграмме ДОКТОР-ПАЦИЕНТ, приведенной на рис. 6.

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 набор символов --.

Перед тем как перейти непосредственно к рассмотрению использования основных SQL-операторов, скажем еще несколько слов об организации проектирования БД. Процесс создания БД - это сложный многоэтапный процесс, 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 "РАЗДЕЛЕНИЕ 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", проводить миграцию созданной БД на другой сервер (достаточно на новом сервере выполнить все задачи в порядке 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 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, оформите 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 целесообразность оформить один дополнительный запускающий скрипт, например start.sql, и поместить в него запуск всех остальных скриптов. Поверьте на слово, это значительно сэкономит ваше время в дальнейшем. Например, если в задачу 000001/VER001/входят файлы: db1.sql., db2.sq1, db3.sql, то файл start.sql может быть представлен следующим образом:

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

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 БД, все операторы данной группы начинаются с ключевого слова CREATE.

CREATE DATABASE

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

```
-- 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"

darafile

'/oracle/db/systemOl.dbf' size 100 M

logfile

'/oracle/db/logl.dbf’ size 1 M,

'/oracle/db/log2.dbf' size 1 M;

disconnect

spool off
```

CREATE TABLESPASE

Создает в БД область для хранения таблиц, сегментов и индексов, определяет файлы БД, параметр хранения по умолчанию и режим табличного пространства (автономный или оперативный).

CREATE USER

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

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_D1PL0MA_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

) tablespase 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

Вставляет строки в таблицу или в базовую таблицу представления.

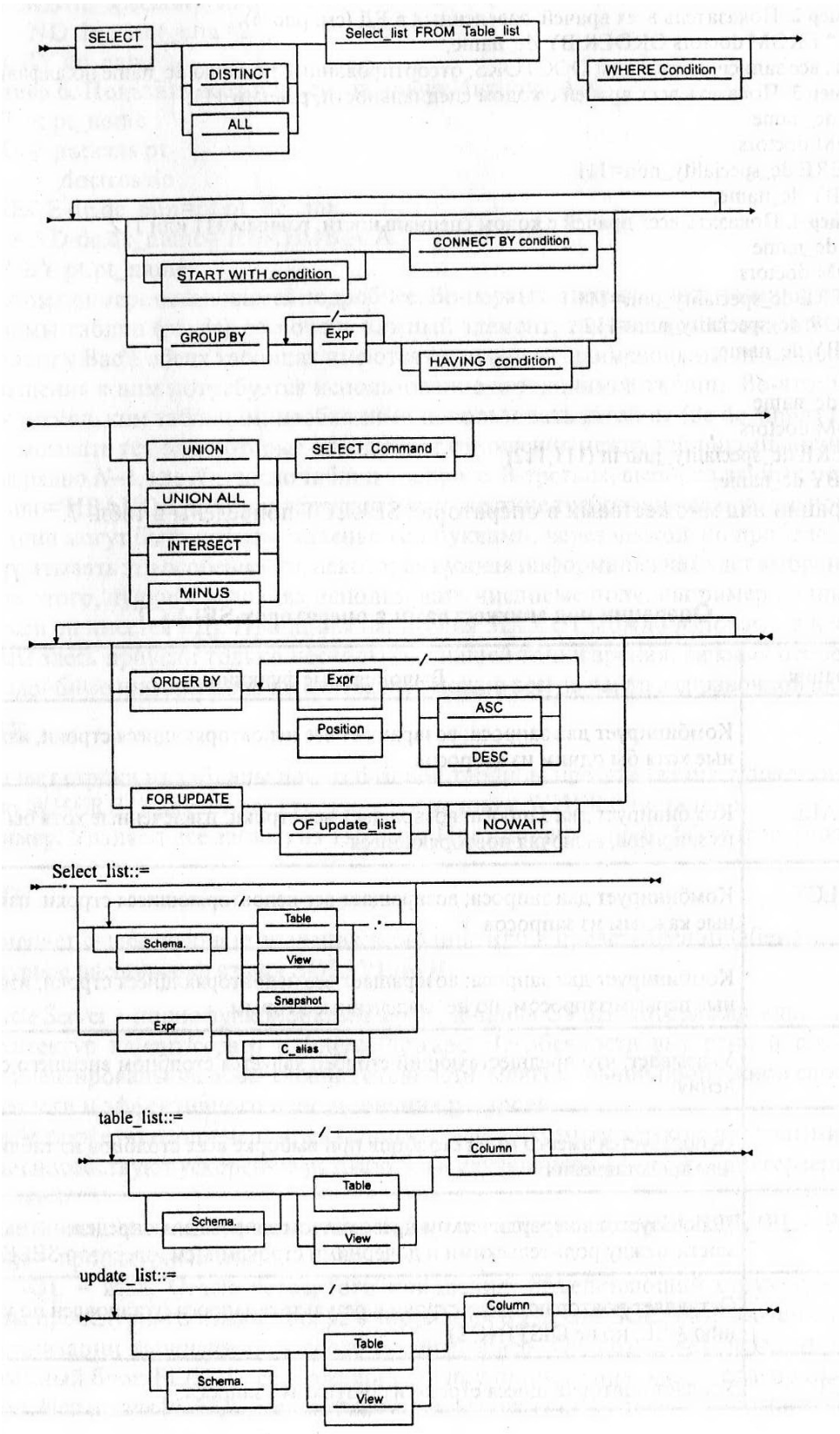
Пример. В качестве примера рассмотрим вставку данных в таблицу "Праздничные дни" (PRAZDNIKI)

```
insert into pra d values ('понедельник');
```

```
insert into pra d values ('вторник');

insert into pra d values ('среда');

insert into pra d values ('четверг');

insert into pra d values ('пятница');

insert into pra d values ('суббота');

insert into pra d values ('воскресенье');
```

SELECT

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



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. Показатель всех врачей, заведенных в БД (см. рис. 6).

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=l11

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=l11

OR dc_speciality_nnn=l12

ORDER BY dc_name;

2-й способ

SELECT dc_name

FROM doctors ,

WHERE dc_speciality_nnn in (111,112)

ORDER BY dc_name;

Операции над множествами в операторах SELECT приведены в табл. 7.

Таблица 7

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
(+)	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 (см. 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 ,doctros 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. Во-первых, здесь впервые появились в запросе псевдонимы таблиц (pt, dc), это очень важный элемент, так как может оказаться, что по нерадивости у Вас в обеих таблицах имеются одинаковые наименования столбцов, и тогда для обращения к ним потребуется использование псевдонимов таблиц. Во-вторых, 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 равно N-1, где N- число таблиц в запросе. В-третьих, 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.) и если не учитывать эти особенности, некоторая нужная информация не будет выбрана. Чтобы избежать этого, лучше в условиях использовать числовые поля, например личный номер врача (если он имеется БД). Принципы написания SELECT можно изложить в нескольких томах, мы здесь привели 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).

Процедурное расширение языка SQL-PL/SQL

Oracle Server - полнофункциональная реляционная СУБД, которая идеально подходит для архитектур клиент/сервер и Internet/Intranet. Особенности внутренней архитектуры Oracle ориентированы на обеспечение готовности, максимальной пропускной способности, безопасности и эффективного использования ресурсов.

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, разработан специально для организации вычислений в среде клиент/сервер. Он позволяет передать на сервер программный блок PL/SQL, содержащий 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.

Вся логика приложений делится на клиентскую и серверную части. Серверная часть может быть реализована 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 БД.

Хранимые процедуры - часть логики приложения, особенно нуждающаяся в доступе к БД, 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 БД.

Пакеты - часть логики приложений БД функций и пакетов, предназначенных для решения задач в рамках одного модуля (подсистемы) АИС.

Триггеры базы данных - триггеры для организации сложного контроля целостности, выполнения протоколирования (аудита) и других функций безопасности, реализации в приложениях выдачи предупреждений и мониторинг.

Декларативная целостность. Ограничения активизируются сервером всякий раз, когда записи вставляются, обновляют 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 БД.

Список зарезервированных слов PL/SQL

Язык PL/SQL также включает зарезервированные слова, имеющие определенное значение в операторах PL/SQL [7].

ФУНКЦИИ

Числовые функции

Функция	*Возвращаемое значение*
ABS(n)	Абсолютное значение величины *n*
CEIL(n)	Наименьшее целое, большее или равное *n*
COS(n)	Косинус *n* (угла, выраженного в радианах)
COSH(n)	Гиперболический косинус *n*
EXP(n)	**e** в степени *n*
FLOOR(n)	Наибольшее целое, меньшее или равное *n*
LN(n)	Натуральный логарифм *n*, где *n* > 0
LOG(m, n)	Логарифм *m* по основанию *m*

Функция	*Возвращаемое значение*
MOD(m.n)	Остаток от деления *m* на *n*
POWER(w.n)	*m* в степени *n*
ROUND(n[,m])	*n*, округленное до *m* позиций после десятичной точки. По умолчанию *m* равно нулю
SIGN(n)	Если *n* < 0, –1; если *n* = 0, 0; если *n* > 0, 1
SIN(n)	Синус *n* (угла, выраженного в радианах)
SINHM	Гиперболический синус
SQRT(n)	Квадратный корень от *n*. Если *n* < 0, возвращает значение NULL
TAN(n)	Тангенс *n* (угла, выраженного в радианах)
TANH(n)•	Гиперболический тангенс *n*
TRUNC(n[,m])	*n*, усеченное до *m* позиций после от десятичной точки. По умолчанию *m* равно нулю

Символьные функции

Символьные функции, возвращающие символьные значения:

Функция 1	*Возвращаемое значение*
CHR(n)	Символ с кодом *n*
CONCAT(char1, char2)	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*, все буквы которой преобразованы в строчные
LPAD(char1.n [,char2})	Символьная строка *char1*, которая дополняется слева последовательностью символов из *char2* так, чтобы общая длина строки стала равна *n*. Значение *char2* 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 *nls_sort* определяет последовательность сортировки

Функция 1	*Возвращаемое значение*
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 *nls_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 *char1*, которая дополнена справа последовательностью символов из *char2* 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 "–" (один пробел)
SOUNDEX(char)	Символьная строка, содержащая фонетическое представление для *char*, на английском языке
SUBSTR(char,m[,n])	Фрагмент символьной строки *char*, начинающий с символа *m*, длиной *n* символов (до конца строки, если параметр *n* не указан)
SUBSTRB(char,m[,n])	Фрагмент символьной строки *char*, начинающийся с символа *m*, длиной *n* байтов (до конца строки, если параметр *n* 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*
ASCII(char)	Возвращает десятичный код первого символа строки *char* в кодировке, принятой в БД (Код ASCII в системах, использующих кодировку ASCII). Возвращает значение первого байта многобайтового символа
INSTR(char1. char2[,n[,m]])	Позиция первого символа *m*-го фрагмента строки *char1*, совпадающего со строкой *char2*, начиная с *n*-го символа. По умолчанию *n* и *m* равны 1. Номер символа отсчитывается от первого символа строки *char1*, даже когда $n > 1$

Функция	*Возвращаемое значение*
INSTRB(char1. char2[,n[,m]])	Позиция первого символа *m*-го фрагмента строки *char1*, совпадающего со строкой *char2*, начиная с *m*-го байта. По умолчанию *n* и *m* равны 1. Номер байта отсчитывается от первого символа строки *char1*, даже когда *n* > 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]expr)	Число строк, для которых expr принимает не пустое значение
MAX([DISTINCT|ALL]expr)	Максимальное значение выражения *expr*
MIN(DISTINCT|ALL]expr)	Минимальное значение выражения *expr*
STDDEV([DISTINCT|ALL|n)	Стандартное отклонение величины *n*, нулевые значения опускаются
SUM([DISTINCT|ALL]n)	Сумма значений *n*
VARIANCE([DISTINCTIALL]n)	Дисперсия величины *n*, 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 *d2*
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 char)
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 (округления 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

Функция 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 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 формата	*Возвращаемое значение*
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, или 1 цифра года
IYYY	4 цифры года по стандарту ISO. IYY или IY или I. Последние 3, 2 или 1 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 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 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 воспроизводится в возвращенном значении

Префиксы и суффиксы элементов формата даты. К элементам формата даты можно добавлять следующие префиксы:

FM	"Режим заполнения". Подавляет заполнение проблемами, 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 добавлять 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 формата, 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

Элемент формата	*Пример*	*Описание*
$	'$9999'	Добавляет знак доллара перед числом
B	'B9999'	Заменяет нулевые значения пробелами
MI	'99999MI'	Возвращает знак '-' после отрицательных значений
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, search 1, return 1, [search 2, return 2,]...[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 (expr 1, expr 2)	Возвращает *expr 2*, если *expr 1* имеет пустое значение, в противном случае возвращает *expr 1*
UID	Целое число, которое уникально идентифицирует текущего пользователя
USER	Имя текущего пользователя ORACLE

Функция	*Возвращаемое значение*
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 представления для *expr*

* * *

Подведем некоторые итоги: гибкость СУБД Oracle во многом определяется тем, что отдельные блоки кода PL/SQL программ можно хранить как объекты БД в формате хранимых процедур, функций и пакетов, т.е. часть кода программы хранится 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 БД, обеспечивающий выполнение конкретных действий с передаваемыми параметрами процедуры.

Функция - объект БД, обеспечивающий выполнение конкретных действий над параметрами функции (результат обработки возвращается).

Команды создания функций, процедур и пакетов. Для создания функций, процедур, пакетов БД используются следующие команды.

Create function. Создает автономную хранимую функцию.

Create package. Создает спецификацию для хранимого пакета.

Create package body. Создает тело хранимого пакета.

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 привилегии */
grant all on password to admin;
```

Системные объекты базы данных

К основным системным объектам БД относятся; словарь данных, сегменты отката, временные сегменты и сегмент начальной загрузки [7].

Представление словаря данных. Словарь данных содержит информацию об объектах БД, 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 пользователю
ALL_TAB_COMMENTS	Комментарии к таблицам и представлениям, доступным пользователю
....	....

Описание других представлений словаря данных и системных объектов можно найти в [7].

Защита данных

Транзакции, фиксация и откат. Изменения в БД 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 БД, 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 БД. 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 транзакций, 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 усовершенствования 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 БД, 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 БД.

Ограничения целостности устанавливают бизнес-правила на уровне БД, 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.

Привилегии системного уровня

Каждый пользователь Oracle, определяемый в БД, может иметь одну или несколько из более чем 80 привилегий системного уровня. Эти привилегии очень тонко управляют правами выполнения команд SQL. Администратор БД назначает системные привилегии непосредственно пользовательским учетным разделам Oracle или ролям. Роли затем назначаются учетным разделам Oracle.

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 БД, учетный раздел Oracle должен иметь привилегию системного уровня CREATE SESSION.

Привилегии объектного уровня. Привилегии объектного уровня обеспечивают возможность выполнения определенного типа 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 пользователь вставлял и выбирал строки из его таблицы (но не модифицировал и не 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 привилегии можно разделить на системные и объектные. Системные привилегии относятся ко всему классу объектов, а объектные относятся к заданным объектам. Подробная информация приведена в [7].

Поддержка национальных языков

Средство поддержки национальных языков Oracle (National Language Support - NLS) позволяет пользователям использовать БД на их собственных языках.

ПРИЛОЖЕНИЕ

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 БД и пользовательские приложения для работы с ней.

Этапы выполнения работы.

1. Разработка архитектуры и технологических взаимосвязей взаимодействия пользователей с АСУ КТП на предприятии радиопромышленности (предприятие состоит из следующих подразделений: администрация, отдел автоматизации, конструкторский отдел, отдел технологической подготовки производства, производство - цех, в каждом из которых имеется по два автоматизированных рабочих 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 с назначением им имен и привилегий.

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

3. Формализация функциональной модели АСУ КТП (логической модели). Разработка табличной структуры БД АСУ КТП. С использованием CASE-средств провести моделирование спроектированной структуры БД 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.

Пример. Анализ результатов этапа разработки логической модели (создание таблиц БД), т.е. нормализация и оценка возможности оптимизации структуры базы и формирования отчетности проводятся в следующем порядке:

1. Объединить 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 следует включать данные за конкретный год, определяемый по параметру.

2. Для хранения всех чертежей создать отдельную таблицу, в которой будут 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. 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.

Модули 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 БД. 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 проектированием. 2. Модуль загрузки/выгрузки КД (файлы *.dwg и т.п.). 3. Модуль формирования отчетности по конструкторскому проектированию	1. Модуль управления технологическим проектированиванием. 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)

Перечень основных таблиц БД

1. Таблица пользователей ASUKTR_USER

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

2. 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

4. Таблица управления проектами

PROEKT_NNN	Наименование проекта	Описание проекта	Ссылка на руководителя

5. Таблица схемотехнических документов

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

6. Таблица конструкторских документов по сборочным единицам

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

7. Таблица конструкторских документов по деталям

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

8. Таблица графических документов

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

9. 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

Здесь представлены только базовые 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).

СПИСОК ЛИТЕРАТУРЫ

1. **Diasoftlnfo** / Корпоративный журнал компании DIASOFT. М. 1999.
2. **Материалы** аналитической компании СПЛАН.
3. **Голенцова Е.** Три основных вопроса СУД. ОАО "Весть". 1998.
4. **Громов А.** Управление бизнес-процессами на основе технологии Workflow // Открытые системы, 1997. №1.
5. **Майкл Р.** Oracle 7.3: Энциклопедия пользователя: Пер с англ. К.: Диасофт. 1997. 832 с.
6. **Фаронов В.В., Шумаков П.В.** Delphi 4: Руководство разработчика баз данных. М.: Нолидж, 1999. 560 с., ил.
7. **Урман С.** Oracle 8: Программирование на языке PL/SQL. М.: ЛОРИ, 1999. 608 с.
8. **Луни К.** Oracle 8: Настольная книга администратора. М.: ЛОРИ, 1999. 500 с.
9. **Бобровски С.** Oracle 8: Архитектура. М.: ЛОРИ, 1999. 208 с.
10. **Хансен Г., Хансен Д.** Базы данных: разработка и управление: Пер. с англ. М.: ЗАО "Издательство БИНОМ", 1999. 704 с.
11. **Дунаев С.** Доступ к базам данных и техника работы в сети. Практические приемы современного программирования. М.: ДИАЛОГ-МИФИ, 1999. 416 с.
12. **Петерсен Р.** 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.: Учебн. Центр ОАО Газпром, 1999. 108 с.
15. **Овчинников Е.М.** Корпоративные информационные системы и технологии / Конспект лекций. М.: Учеб. Центр ОАО Газпром, 1999. 78 с.
16. http://www.oracle.ru, http://www.diasoft.ru, http://www.vest.msk.ru, http://info.iu4.bmslu.ru, http://cdl.iu4.bsmtu.ru, http://www.microsoft.com, http://www.ibm.com, http://www.citforum.ru
17. **Материалы** выставок Comtec, NetCom, UnixExpo, Информатика и др. (CDROM).
18. **Материалы** 3-го, 4-го и пятого форумов разработчиков АБС (CDROM).

СОДЕРЖАНИЕ

Андрей Игоревич ВЛАСОВ, Сергей Леонидович ЛЫТКИН, Владимир Леонидович ЯКОВЛЕВ

КРАТКОЕ ПРАКТИЧЕСКОЕ РУКОВОДСТВО РАЗРАБОТЧИКА ПО ЯЗЫКУ PL/SQL

Ордена Трудового Красного Знамени издательство "Машиностроение".

107076, Москва, Стромынский пер., 4. Телефон редакции журнала (095) 269-5510

Художник *В. Н. Погорелое*. Художественный редактор *Т. Н. Погорелова*. Технический редактор *Т.П. Андреева.* Корректор *Л.И. Сажина*

Сдано в набор 31.01.00. Подписано в печать 15.03.00. Формат 60x88 1/16. Бумага офсетная. Печать офсетная. Усл.-печ. л. 3,92. Уч.-изд. л. 6,04. Заказ 214.

Журнал зарегистрирован в Комитете Российской Федерации по печати. Свидетельство о регистрации

№ 013895 от 05.05.98.

Отпечатано в Подольской типографии Чеховского полиграфического комбината. 142100, г. Подольск, ул. Кирова, 25