ЛЕКЦИЯ № 3.1. НАЗНАЧЕНИЕ И СОСТАВ СИСТЕМНЫХ СРЕД САПР

1. Системные среды автоматизированных систем.
2. Подходы к интеграции ПО в САПР.
3. Технологии интеграции ПО типа DDE и OLE.
4. Управление данными в САПР.
5. Интеллектуальные серверы БД.

1. Системные среды автоматизированных систем.

Системы автоматизированного проектирования относятся к числу наиболее сложных и наукоемких автоматизированных систем (АС). Наряду с выполнением собственно проектных процедур необходимо автоматизировать также управление проектированием, поскольку сам процесс проектирования становится все более сложным и зачастую приобретает распределенный характер. На крупных и средних предприятиях заметна тенденция к интеграции САПР с системами управления предприятием и документооборота. Для управления столь сложными интегрированными системами в их составе имеется специальное ПО — системная среда САПР или АС.

Первые системные среды САПР, называвшиеся мониторными подсистемами или Framework (FW), появились на рубеже 70...80-х г.г. В настоящее время основными функциями системных сред САПР являются управление данными, управление проектированием, интеграция ПО, реализация интерфейса с пользователем САПР, помощь в разработке и сопровождении ПО САПР.

ХерминFramework применительно к системным средам САПР был введен в 1980 г. фирмой Cadence — одним из пионеров в создании системных сред САПР. Кроме Cadence, тематикой Frameworks для САПР электронной промышленности занималось несколько ведущих в этой области фирм (MentorGraphics, IBM, DEC, SunMicrosystems и др.), создавших международную ассоциацию CFI (CADFrameworkInitiative). Широкую известность получили такие системные среды, как FalconFramework фирмы MentorGraphics, DesignFramework-2 фирмы Cadence и JCF (Jessy-CommonFramework) европейской программы ESPRIT.

Важно отметить, что проблема системных сред САПР, зародившаяся в процессе становления САПР электронной промышленности, получила развитие при реализации CALS-технологии в различных отраслях машиностроения.

В типичной структуре ПО системных сред современных САПР можно выделить следующие подсистемы.

Ядро отвечает за взаимодействие компонентов системной среды, доступ к ресурсам ОС и сети, возможность работы в гетерогенной среде, настройку на конкретную САПР (конфигурирование) с помощью специальных языков расширения.

Подсистема управления проектом, называемая также подсистемой сквозного параллельного проектирования CAPE (ConcurrentArt-to-ProductEnvironment), выполняет функции слежения за состоянием проекта, координации и синхронизации параллельно выполняемых процедур разными исполнителями. Примерами подсистем управления проектами в машиностроении могут служить Design Manager в САПР Euclid, UG/Manager в Uni graphics. Иногда в отдельную подсистему выделяют управление методологией проектирования. При этом под *методологией* понимают совокупность методов и средств образования *маршрутов* проектирования — последовательностей проектных операций и процедур, ведущих к цели проектирования.

Методы построения маршрутов проектирования (workflow) зависят от типа проектных задач. Различают простые задачи, выполняемые одной программой, линейные, в которых нет разветвлений в межпрограммных связях, и комплексные. Методы построения маршрутов могут быть основаны на предварительном описании задач или на предварительном описании правил конструирования задач. В описании задач фигурируют порты, с которыми соотнесены данные. Порты могут быть обязательными и необязательными, порождающими дополнительные данные или данные нового объекта. Описания задач даются в виде графов или на языках расширения.

Подсистема управления методологией проектирования представлена в виде базы знаний. В этой базе содержатся такие сведения о предметной области, как информационная модель (например, в виде диаграмм сущность-отношение), иерархическая структура проектируемых объектов (например, в виде И-ИЛИ-дерева), описания типовых проектных процедур, типовые фрагменты маршрутов проектирования — так называемые потоки процедур, соответствие между процедурами и имеющимися пакетами прикладных программ, ограничения на их применение и т.п. Часто такую БЗ дополняют обучающей подсистемой, используемой для подготовки специалистов к использованию САПР.

Современные ***системы управления проектными данными*** называют PDM(ProductDataManager), иногда применительно к АСУ используют название EDM (EnterpriseDataManager). PDM предназначены для информационного обеспечения проектирования и выполняют следующие функции:

— хранение проектных данных и доступ к ним, в том числе ведение распределенных архивов документов, их поиск, редактирование, маршрутизация и визуализация;

— управление конфигурацией изделия, т.е. ведение версий проекта, управление внесением изменений;
— создание спецификаций;
— защита информации;
— интеграция данных (поддержка типовых форматов, конвертирование данных).

Основной компонент PDM — *банк данных* (БнД). Он состоит из системы управления базами данных и баз данных (БД). Межпрограммный интерфейс в значительной мере реализуется через информационный обмен с помощью банка данных. PDM отличает легкость доступа к иерархически организованным данным, обслуживание запросов, выдача ответов не только в текстовой, но и в графической форме, привязанной к конструкции изделия. Поскольку взаимодействие внутри группы проектировщиков в основном осуществляется через обмен данными, то в системе PDM часто совмещают функции управления данными и управления параллельным проектированием.

Подсистема интеграции ПО предназначена для организации взаимодействия программ в маршрутах проектирования. Она состоит из ядра, отвечающего за интерфейс на уровне подсистем, и оболочек процедур, согласующих конкретные программные модули, программы и/или программно-методические комплексы (ПМК) со средой проектирования.

Интеграция ПО базируется на идеях объектно-ориентированного программирования. Следует различать синтаксический и семантический аспекты интеграции. Синтаксическая интеграция реализуется с помощью унифицированных языков и форматов данных, технологий типа ODBC для доступа к общему банку данных или компонентно-ориентированных (CBD — Component-Based Development) технологий. Пример унифицированного формата — TES (Tool Encapsultion Specification), предложенного консорциумом CFI. Информация из TES используется для создания оболочек модулей при инкапсуляции. Семантическая интеграция подразумевает автоматическое распознавание разными системами смысла передаваемых между ними данных и достигается значительно труднее.

Подсистема пользовательского интерфейса включает в себя текстовый и графический редакторы и поддерживается системами многооконного интерфейса типа X WindowSystem или OpenLook.

Подсистема CASE предназначена для адаптации САПР к нуждам конкретных пользователей, разработки и сопровождения прикладного ПО. Ее можно рассматривать как специализированную САПР, в которой объектом проектирования являются новые версии подсистем САПР, в частности, версии, адаптированные к требованиям конкретного заказчика. Другими словами, такие CASE-подсистемы позволяют пользователям формировать сравнительно с малыми затратами усилий варианты прикладных ПМК из имеющегося базового набора модулей под заданный узкий диапазон конкретных условий проектирования. В таких случаях CASE-подсистемы называют *инструментальными средами.*

CASE-система, как система проектирования ПО, содержит компоненты для разработки структурных схем алгоритмов и “экранов” для взаимодействия с пользователем в интерактивных процедурах, средства для инфологического проектирования БД, отладки программ, документирования, сохранения “истории” проектирования и т.п. Наряду с этим, в CASE-подсистему САПР входят и компоненты с специфическими для САПР функциями.

Так, в состав САПР Microstation (фирма Bentley Systems) включена инструментальная среда Micro station Basic и язык MDL (Microstation Development Language) с соответствующей программной поддержкой. Язык MDL — С-подобный, с его помощью можно лаконично выразить обращения к проектным операциям и процедурам. В целом среда Microstation Basic близка по своим функциям к среде MS Visual Basic, в ней имеются генератор форм, редактор, конструктор диалога, отладчик.

САПР Спрут (российская фирма Sprut Technologies) вообще создана как инструментальная среда для разработки пользователем потоков задач конструкторского и технологического проектирования в машиностроении с последующим возможным оформлением потоков в виде пользовательских версий САПР. Сконструированный поток поддерживается компонентами системы, в число которых входят графические 2D и 3D подсистемы, СУБД, продукционная экспертная система, документатор, технологический процессор создания программ для станков с ЧПУ, постпроцессоры.

Наиболее известной CASE-системой в составе САПР в настоящее время является описываемая ниже система CAS.CADE фирмы Matra Datavision, с ее помощью фирма разработала очередную версию EuclidQuantum своей САПР Euclid.

2. Подходы к интеграции ПО в САПР.

Для создания ПО САПР так же, как и других сложных автоматизированных информационных систем, определяющее значение имеют вопросы интеграции ПО. Теоретической базой для создания технологий интеграции ПО в САПР являются:

1) методология автоматизированного проектирования, в соответствии с которой осуществляются типизация проектных процедур и маршрутов проектирования в различных предметных областях, выявление типичных входных и выходных данных процедур, построение информационных моделей приложений и их обобщение, сравнительный анализ альтернативных методов и алгоритмов выполнения типовых процедур;

2) объектно-ориентированная методология, в соответствии с которой множества сущностей, фигурирующих в процессах проектирования, подразделяются на классы, в классах появляются свои процедуры и типы данных с отношениями наследования. Эти классы могут быть инвариантными и прикладными. Их обобщение и унификация приводят к появлению таких понятий и средств, как интегрированные ресурсы и

прикладные протоколы, фигурирующие в стандартах STEP, или унифицированные программные компоненты типа графических ядер конструкторских САПР. Именно наличие типовых процедур и единообразное толкование атрибутов объектов в рамках конкретных протоколов позволяют разным программным системам “понимать” друг друга при взаимодействии.

Наряду с типовыми графическими ядрами, известны типовые ПМК имитационного моделирования, конструирования деталей и механизмов, технологической подготовки производства и др. Возможность использования типовых программ в составе программных комплексов обусловлена именно унификацией интерфейсов при обменах данными.

В некоторых маршрутах проектирования обмены данными должны происходить с высокой частотой, что обусловливает специфические требования к интерфейсам. Примером могут служить задачи имитационного моделирования, в которых требуется имитировать взаимодействие процессов, описываемых с помощью различного МО (например, на сосредоточенном и распределенном иерархических уровнях, или с помощью аналоговых и дискретных моделей). Для таких задач при моделировании характерно воспроизведение временной последовательности событий, происходящих в анализируемых взаимодействующих системах. Соответственно взаимодействие программ моделирования может происходить через фиксированное число временных шагов или по мере совершения тех или иных событий в моделируемых системах.

Так, в программах смешанного аналого-дискретного моделирования электронных устройств аналоговая часть моделируется с помощью программы анализа электронных схем, а дискретная часть — с помощью программы логического моделирования. Влияние аналоговой части на дискретную отображается в математических моделях путем преобразования непрерывных фазовых переменных в логические переменные в местах сопряжения частей модели, обратное влияние выражается в преобразовании идеализированных логических сигналов в заданные функции времени, соответствующие электрическим сигналам заданной формы. Очевидно, что в содержательной части сообщений, передаваемых из одной части в другую, должны быть сведения либо о состояниях, выражаемых значениями фазовых переменных в интерфейсных узлах, либо о событиях — изменениях фазовых переменных. Обмен сообщениями может происходить многократно в течение акта одновариантного анализа.

В программно-методических комплексах конструирования происходит обработка графической информации. Содержательная часть сообщений относится к геометрическим элементам, их размерам и положению в пространстве. В программах технологической подготовки механической обработки деталей наряду с геометрической информацией о конструкциях заготовок в передаваемые сообщения могут входить сведения об инструменте, технологической оснастке, оборудовании, режимах обработки, нормах времени, траекториях движения инструмента и рабочих органов оборудования и т.п.

Другими словами, в каждом приложении совокупность используемых при обменах понятий, предметных переменных и числовых параметров существенно ограничена и достаточно определена для того, чтобы можно было ставить вопрос о типизации моделей и языка взаимодействия. Такие вопросы решаются в рамках технологий STEP/CALS. Число приложений, нашедших свое описание в прикладных протоколах STEP ограничено, но совокупность таких протоколов может расширяться.

Прикладные протоколы STEP представляют семантическую сторону интеграционных технологий. Для интеграции нужна не только унификация моделей приложений, но и унификация механизмов взаимодействия, примерами которых являются технологии OLE, DDE, а также компонентно-ориентированные технологии.

3. Технологии интеграции ПО типа DDE и OLE.

Современные ОС позволяют работать одновременно с несколькими задачами с выделением каждой задаче своего окна на экране дисплея. Межпрограммные взаимодействия осуществляются путем посылки сообщений, как это принято в объектно-ориентированном программировании. Используются специальные средства организации взаимодействий.

Так, ОС Unix поддерживает взаимодействие асинхронных параллельных процессов, в том числе в разных узлах сети. Каждый клиент должен предварительно зафиксировать свои потребности в виде имен используемых сообщений. Сообщения имеют структуру фрейма. Получатель сообщения определяет, что сообщение относится к нему, вызывает обработчик сообщения и использует полученные данные в соответствии с своими функциями.

В операционных системах Microsoft для организации межпрограммных взаимодействий были предложены средства Clipboard, DDE, OLE и в дальнейшем технология ActiveX.

Работа Clipboard основана на традиционном способе обменных зон — выделении кармана (некоторой области оперативной памяти, разделяемой взаимодействующими программами). При обменах одна программа посылает сообщение в карман, а другая извлекает, интерпретирует и использует это сообщение. Аналогичный режим работы осуществляется с помощью технологии формирования составных документов OLE, но здесь расширены возможности комбинирования данных различных типов в передаваемых документах.

Различают два способа взаимодействия: связь (linking) и внедрение (embedding). При связи в создаваемый документ включается не сам текст из источника, а лишь ссылка на него. Очевидно, что здесь меньше затраты памяти, изменения в источнике автоматически переходят в документ. При внедрении текст из источника физически переносится в документ. После чего документ можно редактировать независимо от источника. Оба этих способа

реализованы в технологии OLE, что и зафиксировано в ее названии (OLE — ObjectLinkingandEmbedding).

При обмене с помощью DDE (DynamicDataExchange) программа-клиент запрашивает режим диалога с программой-сервером. В сообщении указывается имя сервера, имя раздела (обычно раздел
— это файл), имя элемента (обмениваемая порция информации). Предварительно такой элемент (атом) должен быть создан, его адрес зафиксирован в таблице атомов. В ответ на запрос создается канал, по которому сервер передает данные или, что реализуется чаще, пересылает адрес нужного атома. По этому адресу клиент дополнительной командой может получить данные.

Подход к реализации интероперабельности, имеющийся в DDE и OLE, получил развитие в современных компонентно-ориентированных технологиях разработки ПО, рассматриваемых ниже.

4. Управление данными в САПР.

В большинстве автоматизированных информационных систем применяют СУБД, поддерживающие реляционные модели данных.

Среди общих требований к СУБД можно отметить: 1) обеспечение целостности данных (их полноты и достоверности); 2) защита данных от несанкционированного доступа и от искажений из-за сбоев аппаратуры; 3) удобство пользовательского интерфейса; 4) в большинстве случаев важна возможность распределенной обработки в сетях ЭВМ.

Первые два требования обеспечиваются ограничением прав доступа, запрещением одновременного использования одних и тех же обрабатываемых данных (при возможности их модификации), введением контрольных точек (checkpoints) для защиты от сбоев и т.п.

Банк данных в САПР является важной обслуживающей подсистемой, он выполняет функции информационного обеспечения и имеет ряд особенностей. В нем хранятся как редко изменяемые данные (архивы, справочные данные, типовые проектные решения), так и сведения о текущем состоянии различных версий выполняемых проектов. Как правило, БнД работает в многопользовательском режиме, с его помощью осуществляется информационный интерфейс (взаимодействие) различных подсистем САПР. Построение БнД САПР — сложная задача, что обусловлено следующими особенностями САПР:

1. Разнообразие проектных данных, фигурирующих в процессах обмена как по своей семантике (многоаспектность), так и по формам представления. В частности, значительна доля графических данных.
2. Нередко обмены должны производиться с высокой частотой, что предъявляет жесткие требования к быстродействию средств обмена (полагают, что СУБД должна работать со скоростью обработки тысяч сущностей в секунду).

3. В САПР проблема целостности данных оказывается более трудной для решения, чем в большинстве других систем, поскольку проектирование является процессом взаимодействия многих проектировщиков, которые не только считывают данные, но и изменяют их, причем в значительной мере работают параллельно. Из этого факта вытекают следствия: во-первых, итерационный характер проектирования обычно приводит к наличию по каждой части проекта нескольких версий, любая из них может быть принята в дальнейшем в качестве основной, поэтому нужно хранить все версии с возможностью возврата к любой из них; во-вторых, нельзя допускать использования неутвержденных данных, поэтому проектировщики должны иметь свое рабочее пространство в памяти и работать в нем автономно, а моменты внесения изменений в общую БД должны быть согласованными и не порождать для других пользователей неопределенности данных.
4. Транзакции могут быть длительными и трудоемкими. *Транзакцией* называют последовательность операций по удовлетворению запроса. В САПР внесение изменений в некоторую часть проекта может вызвать довольно длинную и разветвленную сеть изменений в других его частях из-за существенной взаимозависимости компонентов проекта (многошаговость реализации запросов). В частности, транзакции могут включать в себя такие трудоемкие операции, как верификация проектного решения с помощью математического моделирования. В результате транзакции могут длиться даже несколько часов и более. Одна из трудностей заключается в отображении взаимозависимости (ассоциативности) данных. При хранении компонентов проекта во внешней памяти затраты времени на обработку запросов оказываются значительно выше, чем в большинстве других автоматизированных систем, с менее выраженными взаимозависимостями данных.
5. Иерархическая структура проектных данных и, следовательно, отражение наследования в целях сокращения объема базы данных.

В определенной мере названные особенности учитываются в СУБД третьего поколения, в которых стали применяться черты объектно-ориентированных (объектных) СУБД. В них наборы данных, характеризующих состояние предметной области (состояние проекта в случае САПР), помещаются в отдельные файлы. Интерпретация семантики данных осуществляется с помощью специальных процедур (методов), сопровождающих наборы. Наследование свойств объектов предметной области выражается с помощью введения категорий класса, надкласса, подкласса. Информационные модели приложений для таких СУБД разрабатываются на основе методик типа IDEF1X.

Объектные БД выгодны, во-первых, тем, что данные по конкретным объектам проектирования не разбросаны по множеству таблиц, как это имеет место в реляционных БД, а сосредоточены в определенных местах. Во-вторых, для каждого объекта могут быть назначены свои типы данных. В результате проще решаются задачи управления и удовлетворения запросов.

Наряду с чисто объектными СУБД (pureODBMS), применяют СУБД объектно-реляционные. В последних происходит объединение свойств реляционных и объектно-ориентированных СУБД: объектно-ориентированная СУБД снабжается непроцедурным языком запросов или в реляционную СУБД вводятся наследование свойств и классы. Непроцедурность входного языка обеспечивается использованием языка SQL. Его операторы непосредственно включаются в программы на языке С. Возможно написание дополнительных программ, интерпретирующих SQL-запросы.

Отличительные особенности СУБД третьего поколения: расширенный набор возможных типов данных (это абстрактные типы, массивы, множества, записи, композиции разных типов, отображение величин с значениями разных типов), открытость (доступность из разных языков программирования, возможность обращения к прикладным системам из СУБД), непроцедурность языка (общепринятым становится язык запросов SQL), управление асинхронными параллельными процессами, состояние которых отражает БД. Последнее свойство позволяет говорить о тесной взаимосвязи СУБД и подсистемы управления проектами DesPM.

Названные особенности управления данными в САПР нашли свое выражение в современных подсистемах управления проектными данными PDM.

В PDM разнообразие типов проектных данных поддерживается их классификацией и соответствующим выделением групп с характерными множествами атрибутов. Такими группами данных являются описания изделий с различных точек зрения (аспекты). Для большинства САПР машиностроения характерными аспектами являются свойства компонентов и сборок (эти сведения называют Billofmaterials — BOM), модели и их документальное выражение (основными примерами могут служить чертежи, *3D* модели визуализации, сеточные представления для конечно-элементого анализа, текстовые описания), структура изделий, отражающая взаимосвязи между компонентами и сборками и их описаниями в разных группах.

Вследствие большого объема проектных данных и наличия ряда версий проектов PDM должна обладать развитой системой поиска нужных данных по различным критериям.

Рассмотренные особенности банков данных в САПР позволяют квалифицировать их как системы DataWarehouse (DW), т.е. хранилища данных. Для хранилищ данных характерен ряд особенностей, совпадающих с названными выше особенностями банков данных САПР: 1) длительное хранение информации, отражающей историю разработок; 2) частота операций чтения данных выше частоты операций обновления данных; 3) использование единых форматов для однотипных данных, полученных из различных источников (например, от разных программно-методических комплексов). Эти особенности позволяют управлять конфигурацией проектов, что, в частности, означает хранение в САПР всех версий проекта и, возможно, данных по проектам предыдущих разработок, удовлетворение

сложных запросов, для ответа на которые требуется извлечение и обработка данных из различных частей хранилища (так называемая многомерная обработка). Модели данных в DW отличаются от реляционных моделей (RM): в RM использованием нормальных форм стремятся максимально уменьшить избыточность данных, что приводит к увеличению числа таблиц, но уменьшенных размеров, однако многомерный поиск, требующийся в DW, в множестве таблиц затруднен. Поэтому в DW чаще используется модель данных “звезда”, в которой имеется общая таблица фактов (FactTable) и каждому факту ставится в соответствие несколько таблиц с необходимыми атрибутами. Целостность данных в DW обеспечивается проверкой и трансформацией данных (datacleaning), вводимых из внешних источников, наличием дисциплины обновления данных, централизованным хранением основной базы, при этом достаточное быстродействие поддерживается передачей копий определенных частей базы в локальные базы, называемые киосками данных (DataMart) и ориентированные на отдельные группы пользователей.

Примером СУБД, учитывающей требования, предъявляемые со стороны САПР, является система IMAN фирмы EDS Unigraphics. Это система управления объектно-ориентированными базами данных, ее можно также назвать системой интеграции данных. Она выполняет функции подсистемы PDM, которые являются функциями хранения данных, управления доступом к ним, контроля вносимых изменений, создания спецификаций изделий, интегрирования прикладных подсистем. Внутри IMAN используется реляционная модель данных, а на интерфейсном уровне — объектно-ориентированная информационная модель. Для синхронизации изменений предусматривается блокировка доступа пользователей, если с БД уже начал работу некоторый пользователь.

Другими известными примерами подсистем управления проектными данными могут служить системы Optegra (фирма Computervision), EuclidDesignManager (MatraDatavision), ProPDM в составе САПР Pro/Engineer (PTC), TechnoDOCS (Российская фирма “Весть”).

Ряд фирм разрабатывает системы PDM, которые можно использовать как самостоятельные продукты и как подсистемы в автоматизированных системах проектирования и управления.

Примером может служить система PartY (фирма Лоция Софт), в которой предусмотрены функции управления конфигурацией изделий, управления проектными данными и документооборотом, графический пользовательский интерфейс, реализация архитектуры клиент-сервер.

5. Интеллектуальные серверы БД.

5.1. Особенности СУБД В САПР

Особенности СУБД в таких сложных системах, как САПР, определяют их квалификацию, как *интеллектуальных* (их еще называют СУБД третьего поколения). К числу признаков интеллектуальной СУБД (дополнительно к охарактеризованным выше) относятся реализация в СУБД части прикладных процедур, что характерно для структуры DBS, *оповещение* пользователей (прикладных программ) об интересующих их изменениях состояния БД, *синхронизация событий* в БД, способность обслуживать прикладные программы, первоначально ориентированные на разные типы СУБД (назовем это свойство *многопротокольностью*).

Оповещение заключается в информировании программы*A* о совершении события, вызванного программой *B* и влияющего на работу программы *A* (рис. 5.4). Примером события может быть выход значения некоторого параметра в БД за допустимые пределы. Наиболее просто информирование можно организовать периодическим опросом состояния БД со стороны *A*. Однако это усложняет ПО и не эффективно по затратам времени и загрузке сети. Лучше возложить функцию оповещения на СУБД, что и присуще интеллектуальным СУБД. Но для этого нужно иметь обратные ссылки на программы, обращающиеся к БД, *правила* (иначе называемые *триггерами*), фиксирующие наступления событий, и процедуры обработки событий (см. рис. 5.4). Удобный вариант оповещения — информирование программы*A* о происшедших событиях во время ее активизации.

Для реализации многопротоколь-ности разрабатывают специальные технологии. Наиболее известной среди них является технология ODBC (OpenDataBaseConnection) фирмы Microsoft.

СУБД

(^Правило J)

Рис 5.4. Схема оповещения

Фактически ODBC представляет собой библиотеку функций для обращенийприкладных программ (1111) к различным СУБД на основе языка SQL. Из 1111 обращение происходит к виртуальной СУБД, в которой с помощью драйверов осуществляется переход к реальной СУБД.

5.2. Распределенные базы данных (РБД).

В крупных АС, построенных на основе корпоративных сетей, не всегда удается организовать централизованное размещение всех баз данных и СУБД на одном узле сети. Поэтому появляются *распределенные базы данных.*

При построении РБД приходится решать ряд сложных проблем, связанных с минимизацией трафика, обеспечением интероперабельности обработки данных и целостности данных.

Минимизация трафика нужна в связи с тем, что при обслуживании запроса могут потребоваться данные из многих узлов, пересылаемые по сети. Возможности минимизации видны из примера обработки данных нескольких таблиц из разных узлов. Очевидно, что целесообразна однократная пересылка таблиц (причем таблиц именно меньшего размера) на один узел, на котором и будет обрабатываться запрос.

Интероперабельность выражает способность взаимодействия программ, работающих в гетерогенных сетях (в разных операционных средах или с разными СУБД). Интероперабельность обеспечивается или с помощью программ-шлюзов (конверторов) для каждой пары взаимодействующих сред, или с помощью единого унифицированного языка взаимодействия. Таким языком для доступа к БД является язык SQL, интероперабельность на его основе имеет место в системе ODBC (OpenDataBaseConnectivity), пример реализации которой показан на рис. 5.5. В примере СУБД FoxPro находится в локальном узле, а СУБД Ingres и Informix - в удаленных узлах.

Прикладная программа имеет ODBC-интерфейс, не зависимый от особенностей различных СУБД. Менеджер драйверов реализует на базе унифицированного языка SQL все нюансы доступа к БД, общие для разных СУБД. Драйвер конкретной СУБД преобразует инвариантные к СУБД запросы в форму, принятую в данной СУБД. В трехзвенной структуре менеджер драйверов может быть размещен ан промежуточном сервере.

Обеспечение целостности в РБД намного сложнее, чем в одноузловых БД. Различают два подхода к построению РБД:

1) тиражирование (репликация), при котором на нескольких серверах (узлах) сети расположены копии БД;

2) полномасштабная распределенность, при которой разные части БД находятся на разных серверах сети (классическая распределенность).

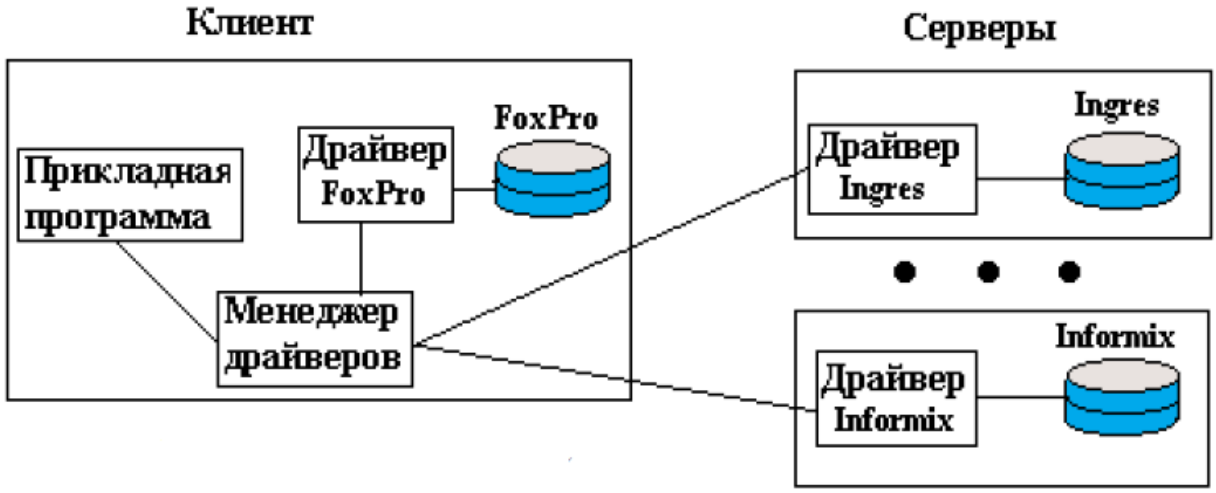

Рис. 5.5. Структура ODBC

Применяют два способа тиражирования.

Способ, называемый репликацией первой копии, основан на выделении среди серверов с копиями БД одного первичного сервера (репликатора). Внесение изменений пользователями возможно только в БД первичного сервера, который в дальнейшем осуществляет тиражирование. *Тиражирование* — это перенос изменений БД из первичного сервера во все вторичные (локальные) серверы, которые используются клиентами только для чтения данных. Репликатор реагирует на события, фиксируемые триггерами, периодически пересылает обновленные данные в копии БД. Недостаток способа - невысокая надежность, присущая любым централизованным структурам.

Надежность повышается при использовании способа голосования. Здесь изменения посылаются не в один первичный, а в некоторые N серверов. При этом любой запрос на чтение направляется к некоторым Mсерверам, причем $N+M>K$, где K — общее число серверов. Принимается последняя по времени обновления версия ответа.

Тиражирование вносит избыточность в хранимые данные, появляются трудности с разрешением конфликтов из-за возможных несогласованных изменений в локальных БД. Однако по сравнению с классическими РБД, в которых данные не дублируются, заметно уменьшается трафик, надежнее и проще работа с локальными БД. Обеспечение надежности и удобства работы особенно актуально в случае ненадежных и медленных каналов связи, что имеет место во многих сетях в России.

В классических распределенных СУБД (РСУБД) необходимо *управлять одновременным доступом,* что должно гарантировать целостность (сериализуемость) БД. Наиболее широко используются алгоритмы управления, основанные на механизме *блокировки.* При этом блокировкой называют ситуацию, при которой некоторая транзакция объявила о желании

получить полномочия на доступ к странице памяти и, следовательно, другие транзакции не имеют права занимать этот ресурс.

Одним из способов управления является централизованное блокирование, при котором на одном из узлов поддерживается единая таблица блокировок. Такой узел устанавливает очередность выполнения транзакций, что исключает конфликты. Однако при централизованном управлении невысока надежность и требуется мощный сервер.

В РСУБД с репликацией нет проблемы согласования при записи действий многих узлов. Собственно тиражирование чаще всего выполняется по правилу полной эквивалентности — обновленные данные сразу же после изменившей их транзакции рассылаются по всем локальным БД. Чтение же выполняется из БД одного конкретного узла, наиболее близкого к пользователю в функциональном или географическом смысле.

Сложнее решать проблемы распределенного управления, что требуется в РСУБД без тиражирования. Одним из распространенных протоколов распределенного управления является протокол двухфазной фиксации транзакций (2PC). На первой фазе инициатор транзакции (координатор) рассылает участникам выполнения транзакции оповещения о блокировке. В ответ узлы сообщают о своей готовности или неготовности. На второй фазе координатор сообщает либо о “глобальной фиксации”, т.е. о выполнении транзакции, либо об откате транзакции. Неприятности возможны при сбоях, которые могут оставить некоторый узел в заблокированном состоянии: он не может ни выполнять транзакцию, ни отменять ее в одностороннем порядке.