МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Обращение к удаленным объектам (RMI)

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Технология RMI

- **Технология RMI** (Remote Method Invocation) – это развитие RPC (его объектная реализация).

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Технология RMI

- **RMI (Remote Method Invocation, т. е. вызов удаленного метода),** которая интегрирована с JDK 1.1, является продуктом компании Jawasoft и реализует распределенную модель вычислений.

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Технология RMI

- **RMI (Remote Method Invocation, т. е. вызов удаленного метода),** которая интегрирована с JDK 1.1, является продуктом компании Jawasoft и реализует распределенную модель вычислений.

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Технология RMI

- **RMI** позволяет клиентским и серверным приложениям через сеть вызывать методы клиентов/серверов, выполняющихся в **Java Virtual Machine**.

Технология RMI

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

КОНЦЕПЦИЯ СОЗДАНИЯ РАСПРЕДЕЛЁННОЙ БАЗЫ ДАННЫХ

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

РАСПРЕДЕЛЕННЫЕ БАЗЫ ДАННЫХ

- Основные принципы создания и функционирования распределенных баз данных
- Трудности в практической реализации распределенных систем
  - технология клиент-сервер
  - технология тиражирования
  - технология объектного связывания

Процессы децентрализации и информационной интеграции

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

1. Много организационно и физически распределенных пользователей работают с общими данными; пользователи с разными именами, полномочиями и задачами, расположены на разных вычислительных системах.
2. Логически и физически распределенные данные, составляющие единое взаимосвязанное целое – общую БД, могут находиться на различных вычислительных установках.

Именно эти две идеи положены в основу создания распределенных ИС и баз данных.

Понятие распределенной БД (DDB)

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

- Под распределенной (Distributed DataBase - DDB) обычно подразумевают базу данных, включающую фрагменты из нескольких баз данных, которые располагаются на различных узлах сети компьютеров, и, возможно, управляются различными СУБД.
- Распределенная база данных выглядит с точки зрения пользователей и прикладных программ как обычная локальная база данных. В этом смысле слово "распределенная" отражает способ организации базы данных, но не внешнюю ее характеристику.

Определение идеальной DDB Криса Дейта

1. Локальная автономия (local autonomy)
2. Независимость узлов (no reliance on central site)
3. Непрерывные операции (continuous operation)
4. Прозрачность расположения (location independence)
5. Прозрачная фрагментация (fragmentation independence)
6. Прозрачное тиражирование (replication independence)
7. Обработка распределенных запросов (distributed query processing)
8. Обработка распределенных транзакций (distributed transaction processing)
9. Независимость от оборудования (hardware independence)
10. Независимость от операционных систем (operationg system independence)
11. Прозрачность сети (network independence)
12. Независимость от баз данных (database independence)

1. Локальная автономия

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

- Это качество означает, что управление данными на каждом из узлов распределенной системы выполняется локально.
- База данных, расположенная на одном из узлов, является неотъемлемым компонентом распределенной системы.
- Будучи фрагментом общего пространства данных, она, в то же время функционирует как полноценная локальная база данных; управление ею выполняется локально и независимо от других узлов системы.

2. Независимость от центрального узла

- В идеальной распределенной системе все узлы сети равноправны и независимы, а расположенные на них базы являются равноправными поставщиками данных в общее пространство данных.
- База данных на каждом из узлов самодостаточна - она включает полный собственный словарь данных и полностью защищена от несанкционированного доступа.

3. Непрерывные операции

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

- Это качество можно трактовать как возможность непрерывного доступа к данным (известное выражение "24 часа в сутки, семь дней в неделю") в рамках DDB вне зависимости от их расположения и вне зависимости от операций, выполняемых на локальных узлах.

4. *Прозрачность расположения*

- Это свойство означает полную прозрачность расположения данных. Пользователь, обращающийся к DDB, ничего не должен знать о реальном, физическом размещении данных в узлах распределенной информационной системы.
- Все операции над данными выполняются без учета их местонахождения.
- Транспортировка запросов к базам данных осуществляется встроенными системными средствами.

5. *Прозрачная фрагментация*

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

- Это свойство трактуется как возможность распределенного (то есть на различных узлах) размещения данных, логически представляющих собой единое целое.
- Существует фрагментация двух типов: горизонтальная и вертикальная.
- Первая означает хранение строк одной таблицы на различных узлах (фактически, хранение строк одной логической таблицы в нескольких идентичных физических таблицах на различных узлах).
- Вторая означает распределение столбцов логической таблицы по нескольким узлам.

6. *Прозрачность тиражирования*

- Тиражирование данных - это асинхронный (в общем случае) процесс переноса изменений объектов исходной базы данных в базы, расположенные на других узлах распределенной системы.
- В данном контексте прозрачность тиражирования означает возможность переноса изменений между базами данных средствами, невидимыми пользователю распределенной системы.
- Данное свойство означает, что тиражирование возможно и достигается внутрисистемными средствами.

7. Обработка распределенных запросов

- Это свойство DDB трактуется как возможность выполнения операций выборки над распределенной базой данных, сформулированных в рамках обычного запроса на языке SQL.
- То есть операцию выборки из DDB можно сформулировать с помощью тех же языковых средств, что и операцию над локальной базой данных.

8. *Обработка распределенных транзакци*

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

- Это качество DDB можно трактовать как возможность выполнения операций обновления распределенной базы данных (INSERT, UPDATE, DELETE), не разрушающее целостность и согласованность данных.
- Эта цель достигается применением двухфазного протокола фиксации транзакций (two-phase commit protocol), ставшего фактическим стандартом обработки распределенных транзакций. Его применение гарантирует согласованное изменение данных на нескольких узлах в рамках распределенной (или, как ее еще называют, глобальной) транзакции.

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

9. *Независимость от оборудования*

- Это свойство означает, что в качестве узлов распределенной системы могут выступать компьютеры любых моделей и производителей - от мэйнфреймов до "персоналок".

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

10. *Независимость от операционных систем*

- Это качество вытекает из предыдущего и означает многообразие операционных систем, управляющих узлами распределенной системы.

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

11. Прозрачность сети

- Доступ к любым базам данных может осуществляться по сети. Спектр поддерживаемых конкретной СУБД сетевых протоколов не должен быть ограничением системы с распределенными базами данных.
- Данное качество формулируется максимально широко - в распределенной системе возможны любые сетевые протоколы.

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

12. Независимость от баз данных

- Это качество означает, что в распределенной системе могут мирно сосуществовать СУБД различных производителей, и возможны операции поиска и обновления в базах данных различных моделей и форматов.

ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ DDB

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Возникающие проблемы:

1. **Проблемы техники представлений (Views)**
2. **Проблема целостности данных**
3. **Проблема обработка распределенных запросов**
4. **Проблема межоперабельности**

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Определение представления

- Представлением называется сохраняемый в БД авторизованный глобальный запрос на выборку.
- Авторизованность означает возможность запуска его конкретным поименованным пользователем.
- Глобальность означает возможность выборки данных из всех БД, находящихся на разных узлах системы.

- В результате таких глобальных авторизованных запросов для конкретного пользователя создается некая виртуальная БД, со своим перечнем таблиц, связей, со своей схемой данных.
- Техника представлений реализуется через введение в язык SQL специализированных конструкций.
- Практическая реализация техники представлений встречает ряд серьезных проблем.

Проблемы практической реализации представлений

- I. **Размещение системного каталога БД**: ядро СУБД должно узнавать, где в самом деле находятся данные. Требование отсутствия центральной установки (независимость узлов) приводит к тому, что системный каталог должен быть на каждом узле.
- II. **При копировании системного каталога**, с целью его обновления на всех узлах возникает ряд серьезных проблем, в том числе, связанных с обеспечением целостности данных.

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Проблема целостности данных

- В DDB поддержка целостности и согласованности данных, ввиду свойств 1-2, представляет собой сложную проблему. Ее решение - синхронное и согласованное изменение данных в нескольких базах данных, составляющих DDB - достигается применением транзакций.
- Если DDB однородна - то есть на всех узлах данные хранятся в формате одной базы и на всех узлах функционирует одна и та же СУБД, то используется механизм двухфазной фиксации транзакций данной СУБД.
- В случае же неоднородности DDB для обеспечения согласованных изменений в нескольких базах данных используют менеджеры распределенных транзакций.

- Если в DDB предусмотрено тиражирование данных, то это сразу предъявляет дополнительные жесткие требования к поддержки целостности данных на узлах, куда направлены потоки тиражируемых данных.
- Проблема в том, что изменения в данных инициируются как локально - на данном узле - так и извне, посредством тиражирования.
- Неизбежно возникают конфликты по изменениям, которые необходимо отслеживать и разрешать.

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Обработка распределенных запросов

- Обработка распределенных запросов (Distributed Query -DQ) - задача, более сложная, нежели обработка локальных запросов и она требует интеллектуального решения с помощью особого компонента - оптимизатора DQ.
- Оптимизатор DQ запросов должен учитывать такие параметры, как, в первую очередь, размер таблиц, статистику распределения данных по узлам, объем данных, передаваемых между узлами, скорость коммуникационных линий, структуру хранения данных, соотношение производительности процессоров на разных узлах и т.д. От интеллекта оптимизатора DQ впрямую зависит скорость выполнения распределенных запросов.

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Межоперабельность

- Во-первых, - это качество, позволяющее обмениваться данными между БД различных поставщиков. Как, например, тиражировать данные из базы данных Informix в Oracle и наоборот? Ответом стало появление продуктов, выполняющих тиражирование между разнородными БД.
- Во-вторых, это возможность некоторого унифицированного доступа к данным в DDB из приложения. Возможны как универсальные решения (стандарт ODBC), так и специализированные подходы.
- Очевидный недостаток ODBC - недоступность для приложения многих полезных механизмов каждой конкретной СУБД, поскольку они могут быть использованы в большинстве случаев только через расширения SQL в диалекте языка данной СУБД, но в стандарте ODBC эти расширения не поддерживаются.

ОТСТУПЛЕНИЕ ОТ ПРИНЦИПОВ ИДЕАЛЬНОЙ DDB КРИСА ДЕЙТА

- Если в жертву приносится принцип 2 (независимость узлов), то получаем DDB, реализованную по технологии клиент -сервер.
- Если в жертву приносится принцип 3 (непрерывные операции), то получаем DDB, реализованную по технологии реплицирования (тиражирования).
- Если в жертву приносится принцип 1 (локальная автономия), то получаем DDB, реализованную по технологии объектного связывания.

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

ТЕХНОЛОГИИ "КЛИЕНТ-СЕРВЕР"

Основной принцип данной технологии заключается в разделении функций стандартного клиентского приложения на четыре группы:

- Первая группа — это функции ввода и отображения данных – интерфейс пользователя.
- Вторая группа объединяет чисто прикладные функции, характерные для данной предметной области (набор запросов, правил, процедур, функций).
- К третьей группе относятся фундаментальные физические функции хранения и управления информационными ресурсами (базами данных, файловыми системами и т.д.).
- Функции четвертой группы — служебные, играющие роль связок между функциями первых трех групп.

Логические компоненты СУБД

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

- В соответствии с этим в любой СУБД выделяются следующие **логические компоненты**:
- компонент представления, реализующий функции первой группы (ввода и отображения данных);
- прикладной компонент, поддерживающий функции второй группы (предметная область);
- компонент физического доступа к информационным ресурсам, поддерживающий функции третьей группы.
- протокол взаимодействия, поддерживающий функции четвертой группы, в котором вводятся и уточняются соглашения о способах взаимодействия первых трех компонент

Модели технологии «клиент-сервер»

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

1. Модель файлового сервера (File Server — FS);
2. Модель доступа к удаленным данным (Remote Data Access — RDA);
3. Модель сервера базы данных (DataBase Server — DBS);
4. Модель сервера приложений (Application Server — AS).

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

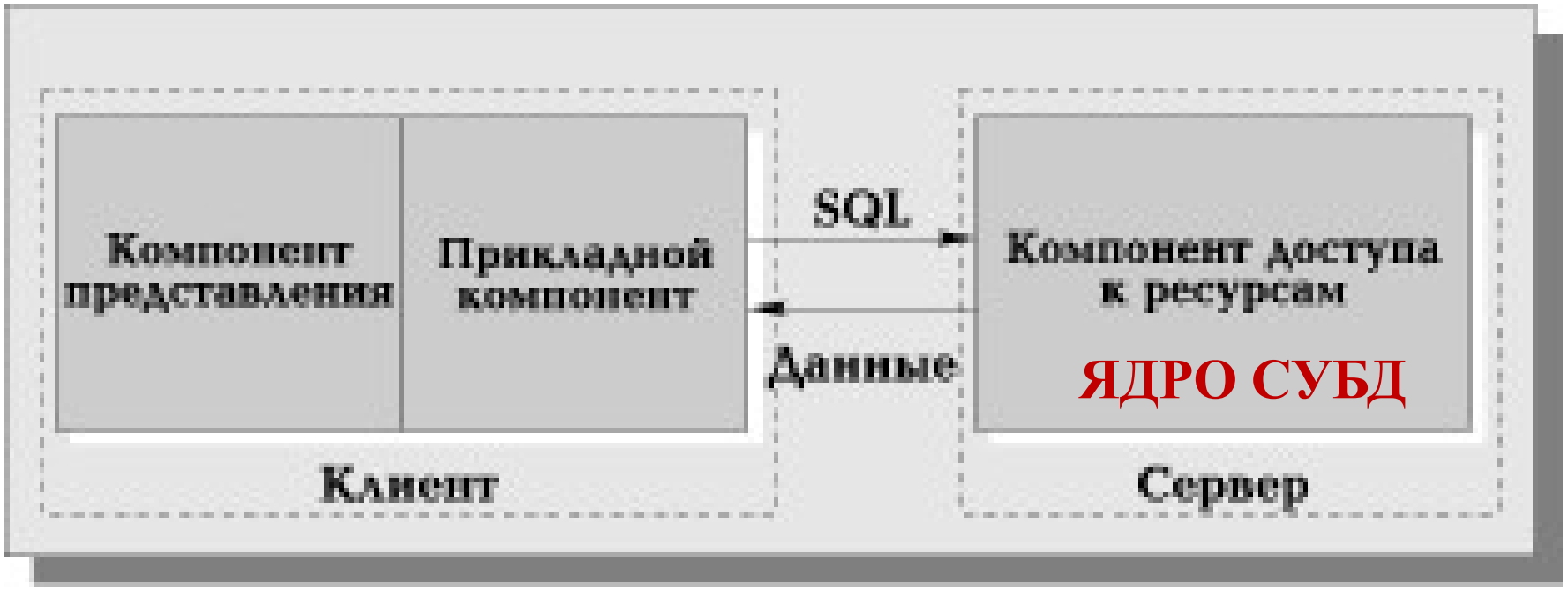
Модель файлового сервера (FS)

Недостатки модели FS

- К технологическим недостаткам модели относят высокий сетевой трафик (передача множества файлов, необходимых приложению), узкий спектр операций манипулирования данными ("данные — это файлы"), отсутствие адекватных средств безопасности доступа к данным (защита только на уровне файловой системы).
- Недоразумения возникают в том случае, когда FS-модель используют не по назначению — например, пытаются интерпретировать как модель сервера базы данных. Место FS-модели в иерархии моделей "клиент-сервер" — это место модели файлового сервера и ничего более.

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Модель доступа к удаленным данным (RDA)



МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Отличие RDA – модели от FS

- Более технологичная RDA-модель существенно отличается от FS-модели характером компонента доступа к информационным ресурсам. Это, как правило, SQL-сервер.
- В RDA-модели коды компонента представления и прикладного компонента совмещены и выполняются на компьютере-клиенте. Клиент поддерживает как функции ввода и отображения данных, так и чисто прикладные функции.
- Доступ к информационным ресурсам на сервере обеспечивается либо операторами языка SQL, либо вызовами функций специальной библиотеки (если имеется интерфейс прикладного программирования — API)

Достоинство RDA-модели

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

- Основное достоинство RDA-модели заключается в унификации интерфейса "клиент-сервер" в виде языка SQL. Действительно, взаимодействие прикладного компонента с ядром СУБД невозможно без стандартизованного средства общения. Поэтому язык SQL используется не только в качестве средства доступа к данным, но и как стандарт общения клиента и сервера.
- С другой стороны, резко уменьшается загрузка сети, так как по ней передаются от клиента к серверу не запросы на ввод-вывод файлов (как в системах с файловым сервером), а запросы на языке SQL, а их объем существенно меньше.

Пассивная роль ядра СУБД в RDA

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

- Клиент направляет запросы к информационным ресурсам (например, к базам данных) по сети удаленному компьютеру.
- На нем функционирует ядро СУБД, которое обрабатывает запросы, выполняя предписанные в них действия и возвращает клиенту результат, оформленный как блок данных.
- При этом инициатором манипуляций с данными выступают программы, выполняющиеся на компьютерах-клиентах, в то время как ядру СУБД отводится пассивная роль — обслуживание запросов и обработка данных.

Недостатки RDA-модели

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

- К сожалению, RDA-модель не лишена ряда недостатков. Во-первых, взаимодействие клиента и сервера посредством SQL-запросов существенно загружает сеть.
- Во-вторых, удовлетворительное администрирование приложений в RDA-модели практически невозможно из-за совмещения в одной программе различных по своей природе функций (функции представления и прикладные функции)

Модель сервера базы данных (DBS)

Достоинства DBS-модели

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

- В DBS-модели компонент представления выполняется на компьютере-клиенте, в то время как прикладной компонент оформлен как набор хранимых процедур и функционирует на компьютере-сервере БД. Там же выполняется компонент доступа к данным, то есть ядро СУБД.
- Достоинства DBS-модели очевидны: это и возможность централизованного администрирования прикладных функций, и снижение трафика (вместо SQL-запросов по сети направляются вызовы хранимых процедур), и возможность разделения процедуры между несколькими приложениями, и экономия ресурсов компьютера за счет использования единожды созданного плана выполнения процедуры.

Модель сервера базы данных

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

реализована в реляционных СУБД. Ее основу составляет механизм хранимых процедур — средство программирования SQL-сервера.

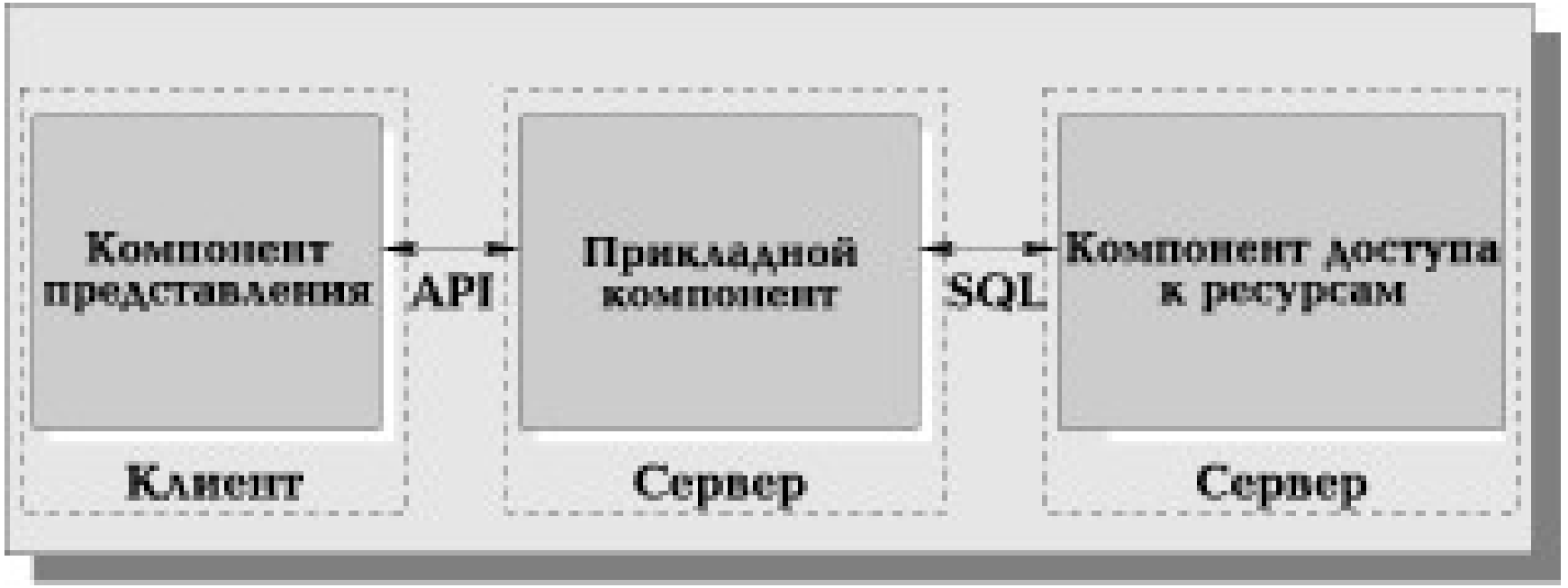
- Процедуры хранятся в словаре базы данных, разделяются между несколькими клиентами и выполняются на том же компьютере, где функционирует SQL-сервер.
- Язык, на котором разрабатываются хранимые процедуры, представляет собой процедурное расширение языка запросов SQL и уникален для каждой конкретной СУБД.

Недостатки DBS-модели

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

- ограниченность средств, используемых для написания хранимых процедур (ХП), которые представляют собой разнообразные процедурные расширения SQL
- Сфера использования ХП ограничена конкретной СУБД, в большинстве СУБД отсутствуют возможности отладки и тестирования разработанных хранимых процедур.

Модель сервера приложений (AS)



Application Programming Interface (API) - стандарт прикладного программного интерфейса

Реализация AS-модели

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

- В AS-модели процесс, выполняющийся на компьютере-клиенте, отвечает за интерфейс с пользователем (то есть реализует функции первой группы). Обращаясь за выполнением услуг к прикладному компоненту, этот процесс играет роль клиента приложения (Application Client — AC).
- Прикладной компонент реализован как группа процессов, выполняющих прикладные функции, и называется сервером приложения (Application Server — AS).
- Все низкоуровневые операции над информационными ресурсами выполняются компонентом доступа на отдельном сервере, по отношению к которому AS играет роль клиента.

Двухзвенная схема разделения функций

- RDA- и DBS-модели опираются на *двухзвенную схему разделения* функций.
- В RDA-модели прикладные функции приданы программе-клиенту («толстый» клиент), в DBS-модели («тонкий» клиент) ответственность за выполнение прикладных функций берет на себя ядро СУБД.
- В RDA-модели прикладной компонент сливается с компонентом представления, в DBS-модели он интегрируется в компонент доступа к информационным ресурсам.
- Двухзвенные модели не могут рассматриваться в качестве базовой модели распределенной системы.

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Трехзвенная схема разделения функций

- В AS-модели реализована *трехзвенная схема разделения* функций, где прикладной компонент выделен как важнейший *изолированный* элемент приложения, для его определения используются универсальные механизмы многозадачной операционной системы, и стандартизованы интерфейсы с двумя другими компонентами.
- AS-модель является фундаментом для мониторов обработки транзакций (Transaction Processing Monitors — TPM), которые выделяются как особый вид программного обеспечения, ориентированного на оперативную обработку распределенных транзакций.

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Программное обеспечение промежуточного слоя (Middleware)

Трехзвенной AS – модель можно считать и потому, что в ней явно выделены:

- Компонент интерфейса с пользователем
- Прикладной компонент управления данными (и базами данных, в том числе)
- *Между ними* расположено программное обеспечение промежуточного слоя (Middleware), выполняющее функции управления транзакциями и коммуникациями, транспортировки запросов, управления именами, доступом и множество др.
- ПО промежуточного слоя (Middleware) - это главный компонент распределенных информационных систем.

- **Главная ошибка**, которая может быть совершена при построении современных распределенных систем - это полное игнорирование ПО промежуточного слоя класса Middleware и использование вместо него двухзвенных моделей "клиент-сервер".
- Существует фундаментальное различие между

двухзвенными моделями (технология «SQL-клиент - SQL-сервер» и

трехзвенными моделями (технология ПО класса Middleware, например, менеджера распределенных транзакций Tuxedo System).

- В случае двухзвенной модели клиент явным образом запрашивает данные, зная структуру базы данных (имеет место так называемый data shipping, то есть "поставка данных" клиенту). Клиент передает СУБД SQL-запрос, в ответ получает данные. Имеет место жесткая связь типа "точка- точка«, для реализации которой все СУБД используют закрытый SQL-канал (например, Oracle SQL*Net).
- Канал закрыт в том смысле, что невозможно, например, использовать программу шифрования SQL- запросов по специальному алгоритму (стандартные алгоритмы шифрования, используемые, например, в Oracle SQL*Net, не сертифицированы и вряд ли будут сертифицированы в будущем.)

- В случае трехзвенной модели клиент явно запрашивает один из сервисов (предоставляемых прикладным компонентом), передавая ему некоторое сообщение (например) и получает ответ также в виде сообщения.
- Клиент направляет запрос в информационную шину (которую строит менеджер Tuxedo System), ничего не зная о месте расположения сервиса.
- Имеет место так называемый function shipping (то есть "поставка функций" клиенту). Важно, что для клиента база данных (в том числе и DDB) закрыта слоем сервисов. Более того, он вообще ничего не знает о ее существовании, так как все операции над базой данных выполняются внутри сервисов.

Вывод по моделям «Клиент-сервер»

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

- Таким образом, речь идет о двух принципиально разных подходах к построению распределенных информационных систем по технологии "клиент-сервер". Первый из них (*двухзвенный*: RDA, DBS) устарел и явно уходит в прошлое.
- Дело в том, что SQL (ставший фактическим стандартом общения с реляционными СУБД) был задуман и реализован как декларативный язык запросов, но отнюдь не как средство взаимодействия "клиент-сервер" (об этой технологии тогда речи не было). Только потом он был "притянут за уши" разработчиками СУБД в качестве такого средства.

Технология тиражирования

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

- В отличие от распределенных баз DDB, тиражирование данных (Data Replication). предполагает *отказ* от их физического распределения и опирается на идею дублирования данных в различных узлах сети компьютеров.
- Суть технологии тиражирования состоит в том, что любая БД (как для СУБД, так и для работающих с ней пользователей) всегда является локальной; данные всегда размещаются локально на том узле сети, где они обрабатываются; все транзакции в системе завершаются локально.

Тиражирование данных

- Тиражирование данных — это асинхронный перенос изменений объектов исходной базы данных в БД, принадлежащие различным узлам распределенной системы.
- Функции тиражирования данных выполняет специальный модуль СУБД — сервер тиражирования данных, называемый репликатором.
- Его задача — поддержка идентичности данных в принимающих базах данных данным в исходной БД. Сигналом для запуска репликатора служит срабатывание правила, перехватывающего любые изменения тиражируемого объекта БД.

Технологии - антиподы

- Технология распределенных БД и технология тиражирования данных — в определенном смысле антиподы.
- Краеугольный камень первой (DDB) — синхронное завершение транзакций одновременно на нескольких узлах распределенной системы, то есть синхронная фиксация изменений в распределенной БД.
- "Ахиллесова пята" этой технологии— жесткие требования к производительности и надежности каналов связи.

- Поскольку БД распределена по нескольким территориально удаленным узлам, объединенным медленными и ненадежными каналами связи, а число одновременно работающих пользователей составляет десятки и выше, то вероятность того, что распределенная транзакция будет зафиксирована в обозримом временном интервале, становится чрезвычайно малой.
- Поэтому практически реализуемым вариантом является технология тиражирования.
- Процесс тиражирования полностью скрыт от прикладной программы; ее функционирование никак не зависит от работы репликатора, который целиком находится в ведении администратора БД.

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Преимущества технологии тиражирования

- Технология тиражирования данных не требует синхронной фиксации изменений (и в этом ее сильная сторона).
- В действительности далеко не во всех задачах требуется обеспечение идентичности БД на различных узлах в любое время. Достаточно поддерживать тождественность данных лишь в определенные критичные моменты времени.
- Следовательно, можно накапливать изменения данных в виде транзакций в одном узле и периодически копировать эти изменения на другие узлы. Это и есть асинхронные фиксации изменений.

Жизненность технологии тиражирования

подтверждается опытом ее использования в области, предъявляющей повышенные требования к надежности — в сфере банковских информационных систем.

- Во-первых, данные всегда расположены там, где они обрабатываются — следовательно, скорость доступа к ним существенно увеличивается.
- Во-вторых, передача только операций, изменяющих данные (а не всех операций доступа к удаленным данным, как в технологии DDB), и к тому же в асинхронном режиме позволяет значительно уменьшить трафик.

Недостатки технологии тиражирования

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

- невозможно полностью исключить конфликты между двумя версиями одной и той же записи.
- Они могут возникнуть, когда вследствие все той же асинхронности два пользователя на разных узлах исправят одну и ту же запись в тот момент, пока изменения в данных из первой базы данных еще не были перенесены во вторую.

- Следовательно, при проектировании распределенной информационной системы с использованием технологии тиражирования данных необходимо предусмотреть конфликтные ситуации (тупиковые ситуации, тупики) и запрограммировать репликатор на какой-либо вариант их разрешения.
- Алгоритмы распознавания и разрешения тупиков основаны на технике приоритетов.

Технология объектного связывания

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

- Современные настольные СУБД обеспечивают возможность **прямого доступа к объектам** (таблицам, запросам, формам) **внешних баз данных «своих» форматов.** В текущем сеансе работы с одной БД пользователь имеет возможность вставить специальные ссылки-объекты и оперировать с данными из другой (внешней, т. е. не открываемой специально в данном сеансе) БД.
- *Объекты из внешней базы данных, вставленные в текущую базу данных,* называются **связанными,** и, как правило, имеют специальные обозначения для отличия от внутренних объектов.
- При этом следует подчеркнуть, что *сами данные физически* в файл текущей базы данных не помещаются, а *остаются в файлах «своих» баз данных.*

Работа со связанными объектами

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

- В системный каталог текущей БД помещаются все необходимые сведения о связанных объектах — внутреннее имя и внешнее, т. е. имя объекта во внешней БД, путь к файлу внешней БД.
- *Связанные объекты* для пользователя ничем не отличаются от *внутренних объектов.* Он может открывать связанные во внешних базах таблицы, выполнять поиск и редактирование данных, строить запросы по таким таблицам и т. д.
- Связанные объекты можно интегрировать в схему внутренней БД, т. е. устанавливать *связи между внутренними и связанными таблицами.*

- *Ядро СУБД* при обращении к данным связанного объекта по системному каталогу текущей БД находит сведения о параметрах файла внешней БД и прозрачно, т. е. невидимо для *пользователя открывает* этот *файл*, а далее обычным порядком организует в оперативной памяти *буферизацию страниц внешнего файла данных для доступа и манипулирования данными.*
- С файлом внешней базы данных, если он находится на другой вычислительной установке, может *в тот же момент времени работать и другой пользователь, что и обеспечивает коллективную обработку общих распределенных данных*.

Недостатки технологии объектного связывания

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

1. Данная технология построения распределенных систем при больших объемах данных в связанных таблицах приведет к **существенному увеличению трафика сети,** так как по сети постоянно передаются, даже не наборы данных, а страницы файлов баз данных, что может приводить к пиковым перегрузкам сети.

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

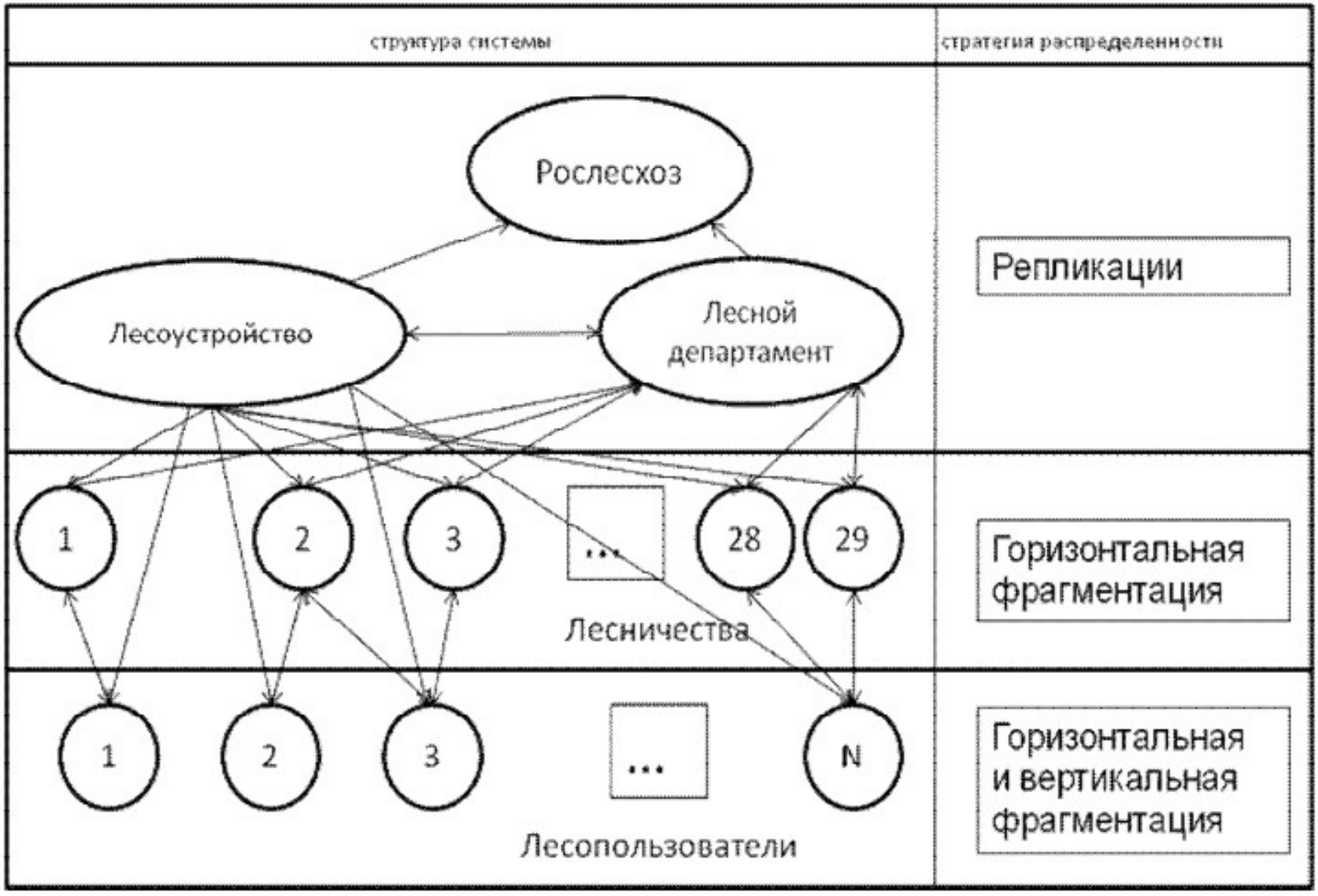
2. Не менее существенной проблемой является **отсутствие надежных механизмов безопасности данных** и обеспечения **ограничений целостности.** Так же как и в модели файлового сервера, совместная работа нескольких пользователей с одними и теми же данными обеспечивается только функциями операционной системы по одновременному доступу к файлу нескольких приложений.

3. Существенной проблемой технологий объектного связывания является появление «брешей» в системах защиты данных и разграничения доступа. Вызовы драйверов ODBC для осуществления процедур доступа к данным помимо пути, имени файлов и требуемых объектов (таблиц), если соответствующие базы защищены, содержат в открытом виде пароли доступа, в результате чего может быть проанализирована и раскрыта система разграничения доступа и защиты данных.

4. Доступ к данным может обеспечиваться как *непосредственно ядром СУБД,* так и специальными дополнительными драйверами ISAM (Indexed Sequential Access Method), входящими, как правило, в состав комплекта СУБД. Такой подход реализует **интероперабельность** построенных распределенных гетерогенных систем, т.е. «разномастность» типов СУБД, поддерживающих локальные базы данных.

При этом, однако, объектное связывание ограничивается только непосредственно таблицами данных, исключая другие объекты базы данных (запросы, формы, отчеты), реализация и поддержка которых зависят от специфики конкретной СУБД.

Распределенная БД лесных насаждений



Агентные технологии

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Понятие программного агента

- ***Программный агент*** – это автономный процесс, способный реагировать на среду исполнения и вызывать изменения в среде исполнения, возможно, в кооперации с пользователями или другими агентами

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Агенты могут применяться при решении следующих задач:

- ***мобильные вычисления*:** миграция агентов может поддерживаться не только между постоянно подсоединенными к сети узлами, но и между мобильными платформами, подключаемыми к постоянной сети на некоторые промежутки времени и возможно по низкоскоростным каналам.

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Агенты могут применяться при решении следующих задач:

- ***поиск информации:*** один человек может быть не в состоянии за короткий срок найти и проанализировать всю необходимую ему информацию. Использование агента позволяет автоматизировать данный процесс. Агент может странствовать по сети и собирать информацию, лучше всего удовлетворяющую поставленной задаче.

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Агенты могут применяться при решении следующих задач:

- ***отбор (обработка) информации.*** Из всех данных, приходящих к клиенту, агент может выбирать только те данные, которые могут быть интересны клиенту.

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

Агенты могут применяться при решении следующих задач:

- ***мониторинг данных***. Агент может осуществлять извещение пользователя об изменениях в различных источниках данных в реальном времени

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Агенты могут применяться при решении следующих задач:

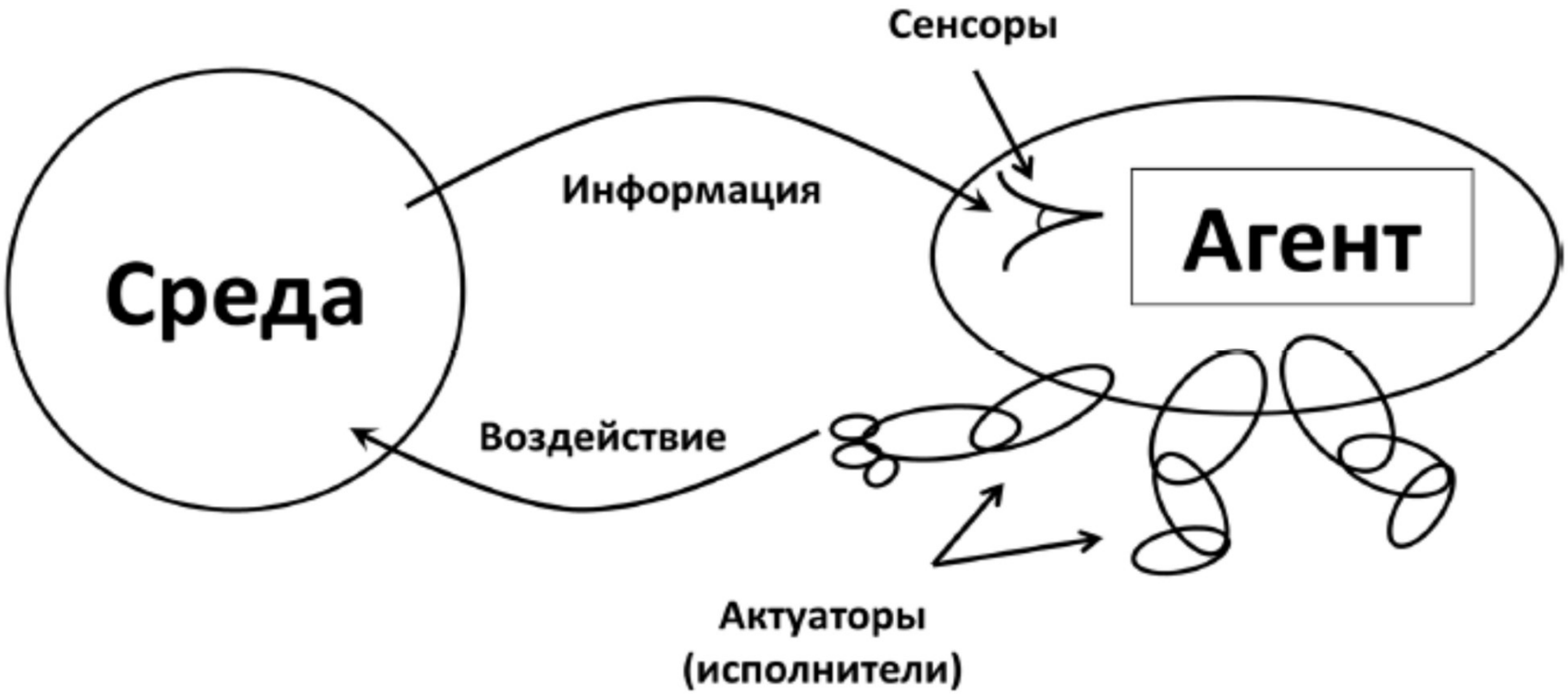
- ***универсальный доступ к данным.*** Агенты могут быть посредниками для работы с различными источниками данных, имеющими механизмы для взаимодействия друг с другом

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Автономный агент

- **это система,** находящаяся внутри окружения и являющаяся его частью, воспринимающая это окружение (его сигналы) и воздействующая на окружение для выполнения собственной программы действий.

Автономный агент



МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

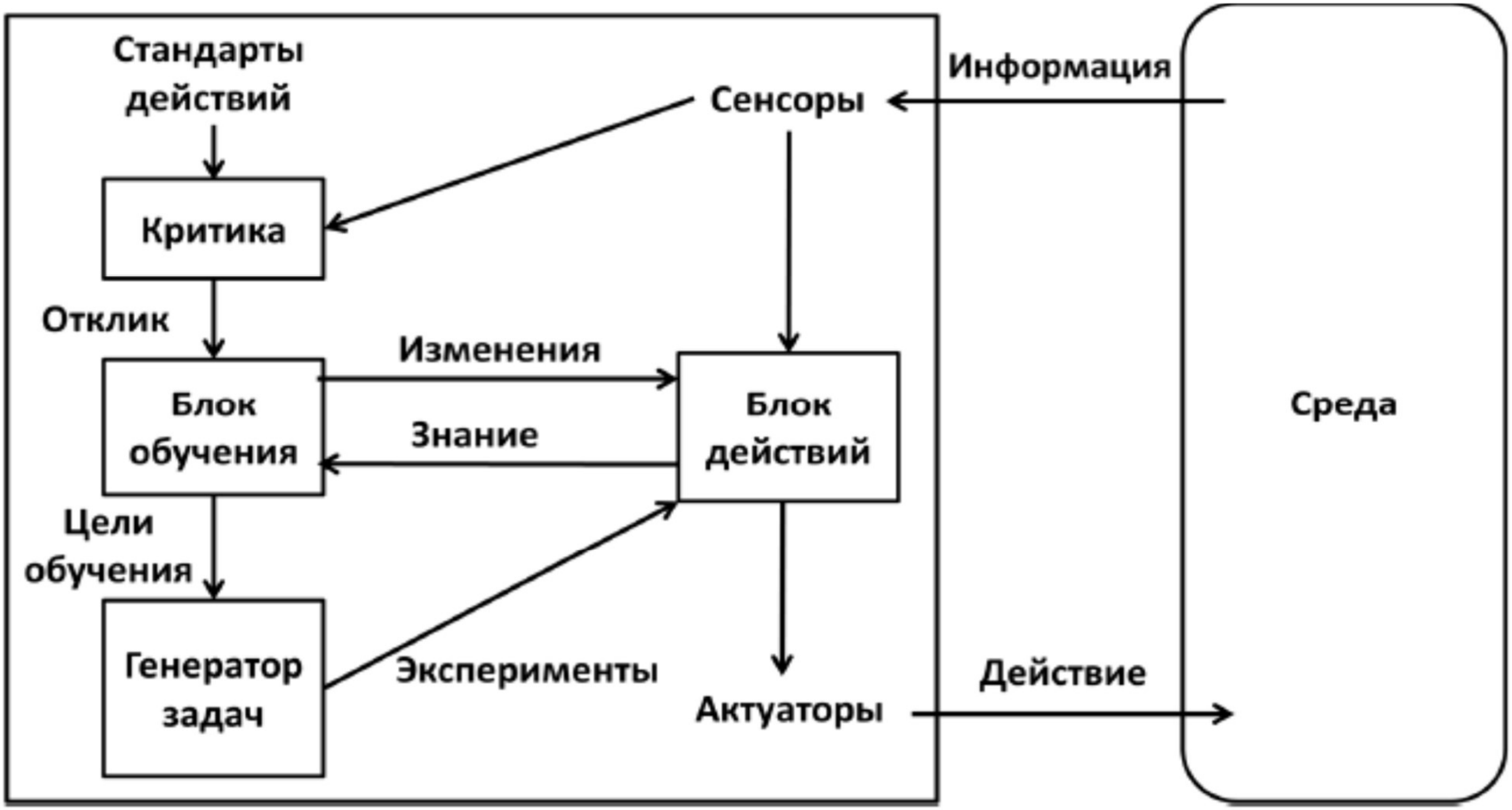
Основные составляющие автономного агента

- ***Сенсоры***: блоки агента, обеспечивающие получение информации об окружающей среде и других агентах;
- ***Актуаторы***: блоки агента, обеспечивающие воздействие на окружающую среду.

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Структура автономного агента

Структура интеллектуального агента



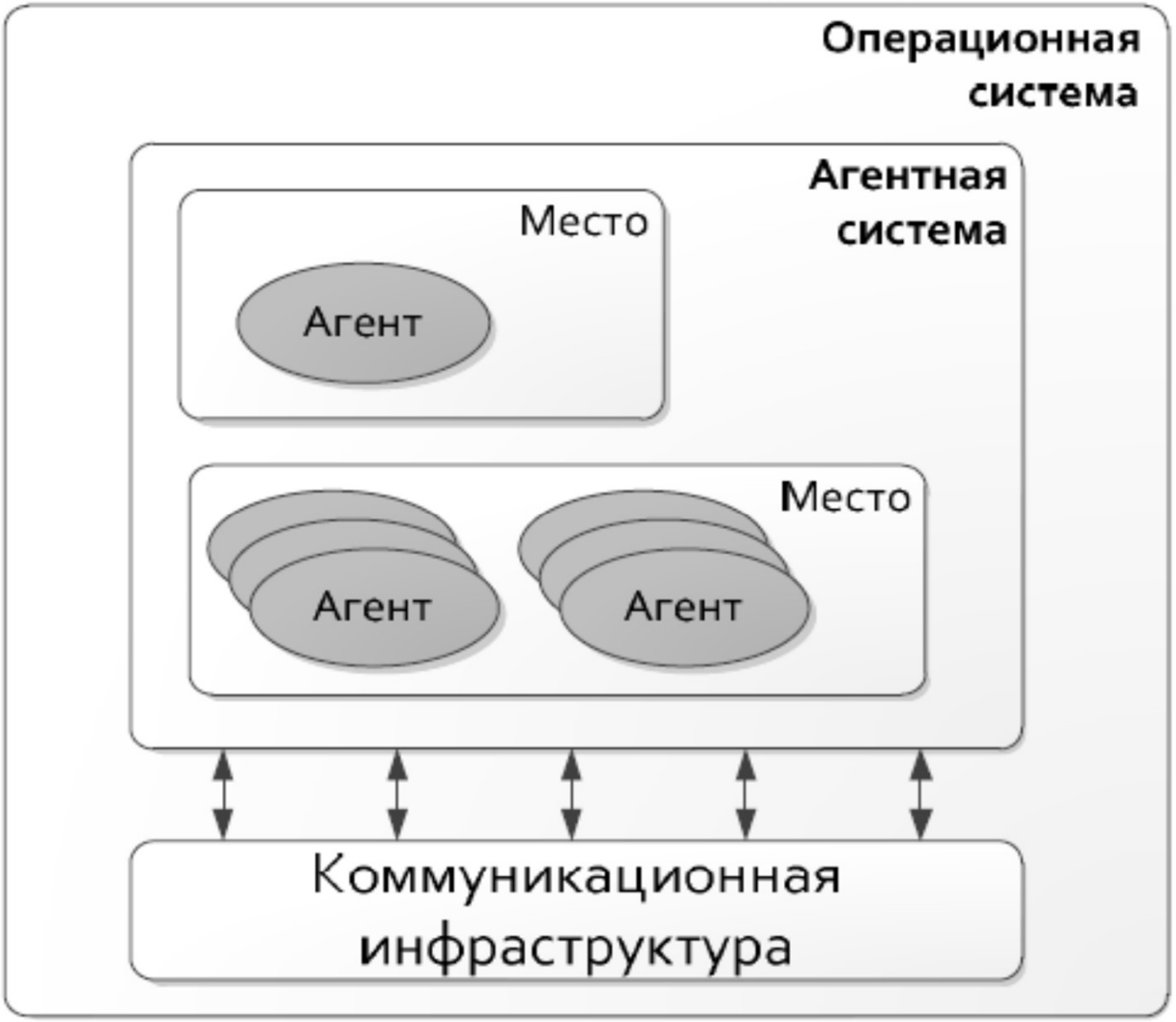
МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Мультиагентные системы

- ***Мультиагентная система***

(МАС, англ. Multi-agent system) — **это система**, образованная несколькими взаимодействующими агентами.

Агентные платформы



МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Агентные платформы

- ***Агентная платформа*** – это промежуточное программное обеспечение, поддерживающее создание, интерпретацию, запуск, перемещение и уничтожение агентов.

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Основные составляющие автономного агента

- ***Сенсоры***: блоки агента, обеспечивающие получение информации об окружающей среде и других агентах;
- ***Актуаторы***: блоки агента, обеспечивающие воздействие на окружающую среду.