**Механизмы доступа к БД**

1. **Понятие технологии доступа к данным.**

**Технология доступа к данным** — это методы и инструменты, которые позволяют пользователям извлекать, изменять, перемещать и обмениваться данными, хранящимися на автономном устройстве хранения, в хранилище данных или в облаке.

1. **Схема движения информации между приложением и базой данных.**

**Схема движения информации между приложением и базой данных** включает следующие основные этапы:

1. Пользователь в процессе диалога с приложением формулирует запрос на некоторые данные из базы данных (БД);
2. Приложение на программном уровне средствами языка манипулирования данными формулирует запрос, с которым обращается к системе управления базой данных (СУБД);
3. Используя свои системные управляющие блоки и таблицы, СУБД с помощью словаря данных определяет местоположение требуемых данных и обращается за ними к операционной системе (ОС);
4. Программы методов доступа файловой системы ОС считывают из внешней памяти искомые данные и помещают их в системные буферы СУБД;
5. Преобразуя полученные данные к требуемому формату, СУБД пересылает их в соответствующую область программы и сигнализирует о завершении операции каким-либо образом (например, кодом возврата);
6. Результаты выбора данных из базы приложение отображает на терминале пользователя.

Также для визуализации движения данных между системами и базами данных можно использовать диаграммы потоков данных (DFD).

1. **Субъекты, участвующие в движении информации между базой данных и приложением.**

Можно выделить несколько субъектов, участвующих в движении информации между базой данных и приложением (например, пользовательским интерфейсом):

1. интерфейсная часть приложения или его программная часть, манипулирующая информацией, хранимой в базе данных;
2. компоненты, обеспечивающие связь приложения с механизмом доступа к базе данных;
3. механизм доступа к базе данных;
4. база данных.
5. **Категории механизмов доступа к данным (перечислить).**

Некоторые категории механизмов доступа к данным:

* Шифрование данных. Обеспечивает безопасность информации при её передаче и хранении. Например, провайдеры шифруют данные при передаче их в облачные приложения или хранилища;
* Контроль доступа. Может быть физическим и логическим (техническим). Физический контроль ограничивает доступ к помещениям, логический — к сетям, файлам, данным, хранящимся на информационных ресурсах;
* Универсальные механизмы доступа к данным. К ним относятся, например, Open Database Connectivity (ODBC), OLE DB, ActiveX Data Objects (ADO), Borland Database Engine (BDE);
* Схемы управления доступом. Дискреционная (избирательная) схема, в рамках которой поддерживаются списки, содержащие информацию о том, кто и к чему имеет доступ. Мандатная схема, при которой к данным и тем, кто имеет к ним доступ, присваиваются классификационные уровни.

1. **Понятие API, функции API.**

API (Application programming interface) — это программный интерфейс приложений, набор инструкций, который позволяет разным приложениям общаться между собой.

Функции API включают получение, отправку, изменение или удаление информации. Для этого одно приложение отправляет запрос в другое, а то, в свою очередь, формирует ответ.

Некоторые конкретные функции API:

* отправка курса валют;
* авторизация в CRM-системе с возможностью добавлять, удалять и изменять информацию;
* проведение платежа через банковскую систему;
* распознавание голоса и перевод его в текст;
* перевод текста;
* получение информации об актуальных остатках товара.

API позволяет не разрабатывать свои функции, а пользоваться уже разработанными.

1. **Достоинства и недостатки API.**

Преимущества протоколов API

1. Стандартизация

Протоколы API определяют общие структуры и форматы данных, что способствует стандартизации и обеспечивает согласованность взаимодействия между разными приложениями и сервисами.

1. Модульность

Протоколы API позволяют разбить сложные системы на более мелкие, независимые компоненты, что упрощает разработку и поддержку приложений.

1. Улучшенное взаимодействие

Протоколы API способствуют сотрудничеству между разными командами разработчиков и организациями, позволяя им создавать приложения и сервисы, которые взаимодействуют друг с другом.

1. Безопасность

Многие протоколы API предоставляют механизмы аутентификации и авторизации, что позволяет контролировать доступ к данным и функциональности.

1. Расширяемость

API-протоколы могут быть расширены и изменены без необходимости изменения всей системы. Это упрощает добавление новых функций и возможностей.

Недостатки протоколов API

1. Сложность

Некоторые API-протоколы могут быть сложными и требовать глубокого понимания для их использования и реализации.

1. Совместимость

Изменения в протоколах API могут привести к проблемам совместимости, особенно если старые версии клиентов или серверов не поддерживают новые изменения.

1. Производительность

Некоторые протоколы могут иметь накладные расходы в виде лишнего объема данных или дополнительных запросов, что может сказаться на производительности.

1. **Универсальные механизмы: ODBC, ОLE DB, ADO.**

Спецификация открытого интерфейса баз данных (ODBC - Open Database Connectivity), предназначена для унификации доступа к данным, размещенным на удаленных серверах. ODBC опирается на спецификации CLI. За реализацию особенностей доступа к каждой отдельной СУБД отвечает соответствующий специальный ODBC-драйвер. Пользовательское приложение этих особенностей не видит, т.к. взаимодействует с универсальным программным слоем более высокого уровня. Таким образом, приложение становится в значительной степени независимым от СУБД. Вместо создания в каждом отдельном случае СУБД-приложения с обращениями через «родной», но быстро устаревающий интерфейс, можно использовать один общий стандартизированный программный интерфейс.

В архитектуре ODBC используется один ODBC Driver Manager и несколько ODBC-драйверов, обеспечивающих доступ к конкретным СУБД. Driver Manager связывает приложение и интерфейсные объекты, которые выполняют обработку SQL-запросов к конкретной СУБД.

Такой подход является достаточно универсальным, стандартизируемым, что и позволяет использовать ODBC-механизмы для работы практически с любой системой.

Однако этот способ также не лишен недостатков:

* увеличивается время обработки запросов (как следствие введения дополнительного программного слоя);
* необходимы предварительная инсталляция и настройка ODBC-драйвера (указание драйвера СУБД, сетевого пути к серверу, базы данных и т.д.) на каждом рабочем месте. Параметры этой настройки являются статическими, т.е. приложение их изменить самостоятельно не может.
* OLE DB (Object Linking and Embedding Data Base), как и ODBC – это прикладные интерфейсы доступа к данным с использованием SQL. OLE DB специфицирует взаимодействие, обеспечивая единый интерфейс доступа к данным через провайдеров – поставщиков данных не только из реляционных БД. В отличие от ODBC, OLE DB предоставляет общее решение обеспечения COM-приложениям доступа к информации независимо от типа источника данных. OLE DB включает два базовых компонента: провайдер данных и потребитель данных. Потребитель (клиент) – это приложение или COM-компонент, обращающийся посредством API-вызовов к OLE DB. Провайдер (сервер) — это приложение отвечающее на вызовы OLE DB и возвращающее запрашиваемый объект – обычно это данные в табличном виде.
* ADO (Active Data Object) – это универсальный интерфейс высокого уровня к OLE DB. Модель объекта ADO не содержит таблиц, среды или машины БД. Здесь основными объектами являются следующие: объект Соединение, создающий связь с провайдером данных; объект Набор данных и объект Команда – выполнение процедуры, SQL-строки. В общем случае ADO можно рассматривать как язык программирования с БД, позволяющий выбирать, модифицировать и удалять записи. И, поскольку он опирается на универсальный OLE DB, то может использоваться практически в любых приложения Microsoft.
* Рассмотренные технологии построения приложения ориентированы на извлечение данных непосредственно из статического источника (хранилища данных) и не могут обращаться за данными к другому прикладному модулю.

1. **Схема выполнения программы с использованием протокола ODBC.**

Схема выполнения программы с использованием протокола ODBC для доступа к данным приводится на рисунке 1.

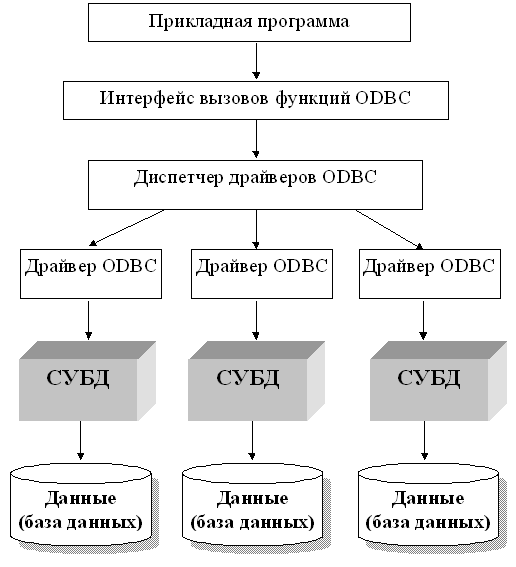


Рисунок 1 – Схема выполнения программы со встроенными инструкциями динамического SQLс применением двухэтапной схемы

1. **Логическая схема доступа к данным с помощью OLE DB .**

Логическая схема доступа к данным с помощью OLE DB представлена на рисунке 2.

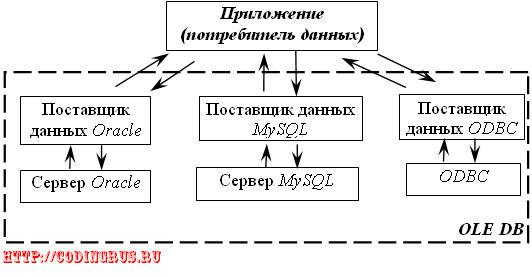


Рисунок 2 - Движение информации между приложением и БД при использовании OLE DB

1. **Схема доступа к информации с помощью ADO.**

Схема доступа к информации с помощью ADO представлена на рисунке 3.

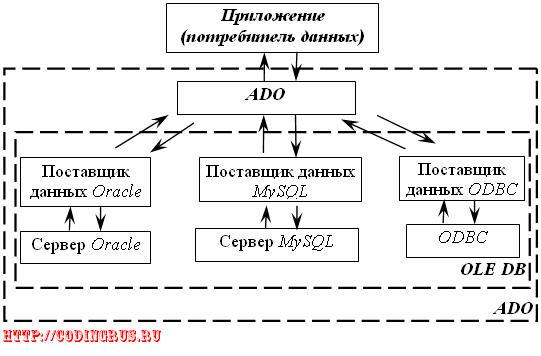


Рисунок 3 - Движение информации между приложением и БД при использовании ADO

1. **Схема доступа к информации с помощью BDE.**

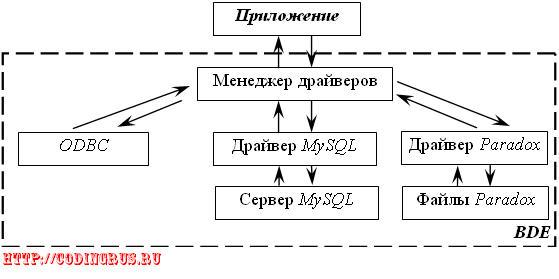


Рисунок 4 - Движение информации между приложением и БД при использовании BDE

1. **Технологии dbExpress, InterBase, JDBC.**

dbExpress – более новая и совершенная технология доступа к удаленным БД от фирмы Borland. Схема работы этого механизма аналогична схеме работы BDE, но при использовании dbExpress не происходит обращения к ODBC. При использовании этого механизма нет необходимости развертывания dbExpress вместе с приложением: достаточно установить только dll-библиотеку либо прилинковать драйвер к исполняемому файлу. Технология dbExpress по сравнению с BDE обеспечивает более быстрый и удобный доступ к данным.

InterBase – специализированная технология доступа к серверу БД Borland InterBase. При работе с этой СУБД обеспечивает наивысшую производительность по сравнению с другими технологиями.

Принципы работы с БД с использованием любой из вышеперечисленных технологий практически одинаковы.

Протокол JDBC

JDBC (Java Database Connectivity) представляет собой API для выполнения SQL-запросов к базам данных из программ, написанных на языке Java.

С развитием глобальных сетей, в частности Интернета, и всех сопутствующих технологий стали появляться новые языки, специально предназначенные для работы в новых условиях. Одним из таких языков является язык программирования Java. В настоящее время Интернет-приложения занимают существенное место на рынке, работая в рамках 2-, 3- и многозвенной архитектуры. При этом значение языка Java как средства создания приложений, работающих с базами данных, существенно возрастает. Именно это и явилось одной из основных причин разработки нового программного интерфейса – JDBC. Первоначально интерфейс JDBC был разработан компанией Sun Microsystems, в настоящий момент этот API поддерживается всеми ведущими коммерческими СУБД.