УПРАВЛЕНИЕ ДАННЫМИ 2025

ЛЕКЦИЯ 6

Поддержка разработки десктопных приложений баз данных

Операция	Приоритет
RENAME	4
WHERE	3
PROJECT	3
TIMES	2
JOIN	2
INTERSECT	2
DIVIDE BY	2
UNION	1
MINUS	1

API ДОСТУПА К ДАННЫМ

- **ODBC** (Open Database Connectivity) - это программный интерфейс доступа к базам данных, разработанный компанией Microsoft на основе спецификаций Call Level Interface (CLI).
- Стандарт CLI призван унифицировать программное взаимодействие с СУБД, сделать его независимым от поставщика СУБД и программно-аппаратной платформы.
- С помощью ODBC прикладные программисты могут разрабатывать приложения для использования одного интерфейса доступа к данным, не беспокоясь о тонкостях взаимодействия с несколькими источниками. Это достигается благодаря тому, что поставщики различных баз данных создают драйверы, реализующие конкретное наполнение стандартных функций из ODBC API с учётом особенностей их продукта.

УСТАНОВКА ODBC

- В ОС MS Windows данный интерфейс устанавливается, как драйвер через автоматический установщик. В дальнейшем, настраивается в оболочке программирования для взаимодействия с разными источниками данных (Администрирование -> Источники данных).

- В ОС Linux и MacOS интерфейс устанавливается через оболочку bash по команде, распаковываясь из пакета. Настраивается через конфигурационные файлы, входящие в комплект поставки (~etc/odbc.ini) и конфигурационные файлы самой ОС (когда это необходимо).

УСТАНОВКА ODBC (BASH)

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/inst
all/master/install.sh)"

brew tap microsoft/mssql-release
https://github.com/Microsoft/homebrew-mssql-
release

brew update
HOMEBREW_ACCEPT_EULA=Y brew install msodbcsql18
mssql-tools18
```

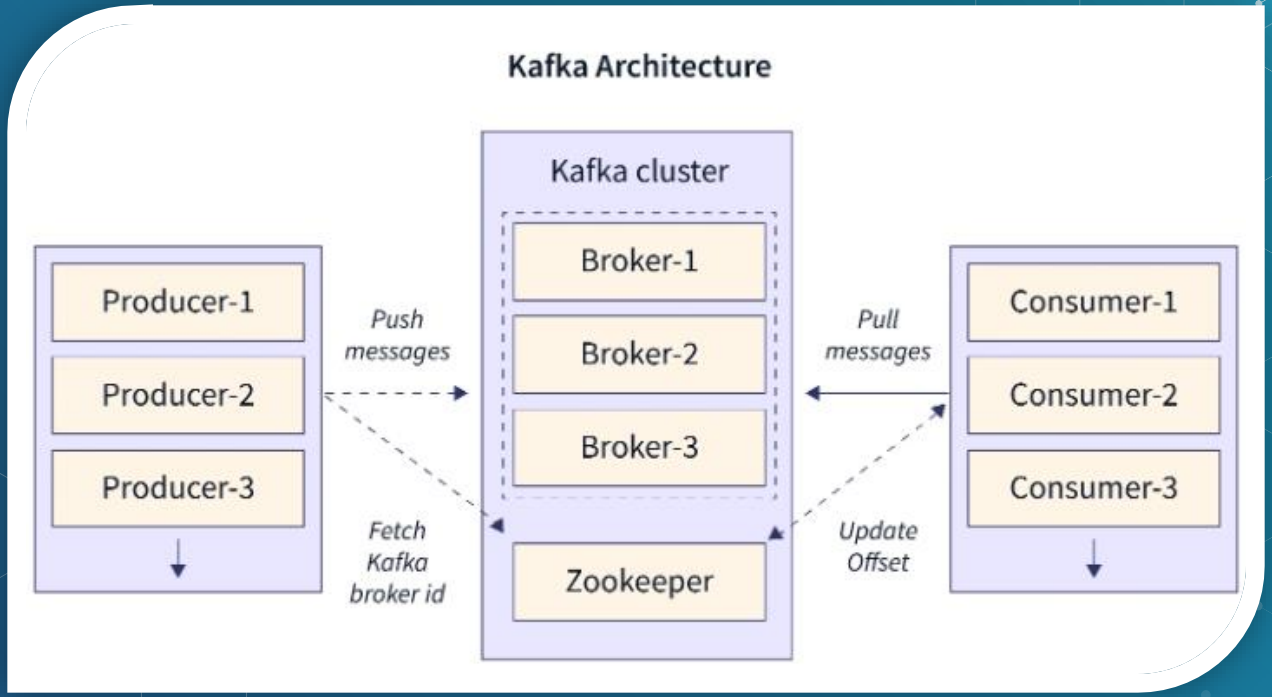
НАСТРОЙКА ODBC НА СТОРОНЕ ПРИЛОЖЕНИЯ (НА ПРИМЕРЕ QT)

```
QString connectString = "Driver={SQL Server};"; // Driver is now {SQL Server}
connectString.append("Server=10.1.1.15,5171;"); // IP,Port
connectString.append("Database=SQLDBSCHEMA;");  // Schema
connectString.append("Uid=SQLUSER;");           // User
connectString.append("Pwd=SQLPASS;");           // Pass
db.setDatabaseName(connectString);
if(db.open())
{
    ui->statusBar->showMessage("Connected");
}else{
    ui->statusBar->showMessage("Not Connected");
}
```

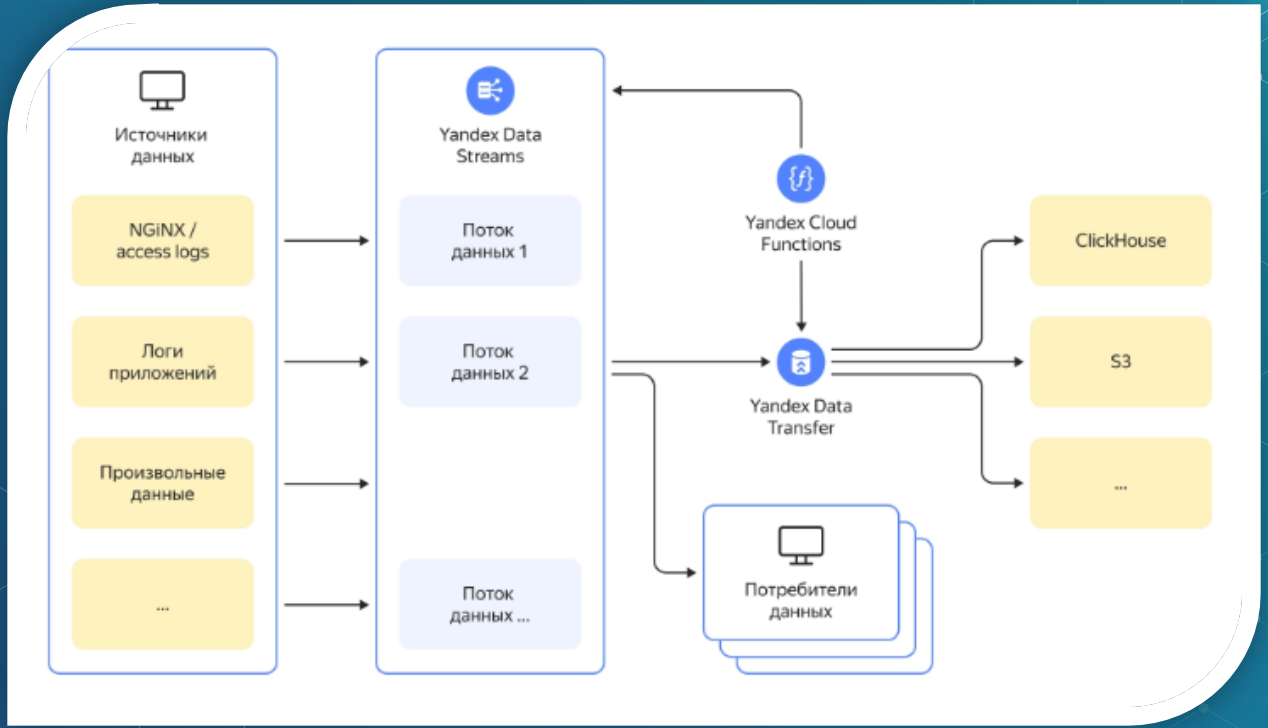
АЛЬТЕРНАТИВНЫЕ API ДОСТУПА К ДАННЫМ

- **OLE DB** (*Object Linking and Embedding, Database*) - набор интерфейсов, которые позволяют приложениям унифицировано работать с данными разных источников и хранилищ информации.
- **JDBC** (Java DataBase Connectivity) - платформенно независимый промышленный стандарт взаимодействия Java-приложений с различными СУБД, реализованный в виде пакета java.sql, входящего в состав Java SE.
- **ADO.NET, Node.JS (tedious), PHP, Python (pyodbc), Spark (Spark Connector)** и так далее.

АРХИТЕКТУРА ПОТОКОВ ДАННЫХ PB KAFKA



АРХИТЕКТУРА ПОТОКОВ ДАННЫХ РВ YANDEX Data Streams



ЯЗЫК SQL В КОДЕ ПРИЛОЖЕНИЙ

- Команды SQL, при определенных условиях, могут напрямую исполняться из программы, написанной на одном из языков программирования высокого уровня. При этом:
  - запросы могут взаимодействовать с внутренними переменными программы;
  - должны содержать команды на подключение и поддержание связи с базой данных.
- Основные подходы к интеграции:
  - embedded (внедренный) SQL, статический;
  - embedded (внедренный) SQL, динамический.

ВОПРОС.

Можно ли в скрипте программы, написанной на одном из языков программирования использовать традиционный SQL код?

ОТВЕТ – НЕТ, НЕЛЬЗЯ

Почему при трансляции языка SQL внутри скрипта компьютерной программы возникают проблемы?

- Переменные приложения баз данных не могут быть использованы в обычной инструкции SQL.
- Запросы на выборку возвращающие несколько строк не всегда эффективно могут быть обработаны на стороне приложения.

РЕШЕНИЕ ПРОБЛЕМЫ ТРАНСЛИРОВАНИЯ ИНСТРУКЦИЙ SQL

- Помещение инструкций языка SQL в программу, написанную на одном из традиционных языков программирования. Изменить правила работы с переменными.

- Запросы на выборку возвращающие несколько строк обрабатывать с помощью специального инструмента (курсор).

РЕШЕНИЕ ПРОБЛЕМЫ ТРАНСЛИРОВАНИЯ ИНСТРУКЦИЙ SQL

- Внедренные инструкции SQL обрабатываются специальным предкомпилятором SQL. Все инструкции SQL начинаются с символа начала и заканчиваются признаком конца, который помечает инструкцию SQL для предварительной компиляции.
- Переменные из программы приложения могут использоваться в внедренных инструкциях SQL везде, где разрешены константы.
- Запросы, возвращающие одну строку данных, обрабатываются с помощью одноэлементной инструкции SELECT. Эта инструкция задает как запрос, так и переменные программы, в которых возвращаются данные.
- Запросы, возвращающие несколько строк данных, обрабатываются с помощью курсоров.

СТАТИЧЕСКИЙ ВНЕДРЕННЫЙ SQL

- Внедренный SQL называется статическим в том случае, когда инструкции SQL, находящиеся в коде не изменяются каждый раз при запуске программы.

- Статический SQL хорошо работает во многих ситуациях и может использоваться в любом приложении, для которого доступ к данным может быть определен во время разработки программы.

- Так как такие инструкции могут быть жестко запрограммированы в программе, они анализируются, проверяются и оптимизируются только один раз во время компиляции. Это приводит к относительно быстрому коду.

ДИНАМИЧЕСКИЙ ВНЕДРЕННЫЙ SQL

- Предположим, что электронная таблица позволяет пользователю ввести запрос, который затем электронная таблица отправляет СУБД для получения данных. Очевидно, что содержимое этого запроса неизвестно программисту при написании программы электронной таблицы.

- Чтобы решить эту проблему, электронная таблица использует форму внедренного SQL, который называется динамический SQL. В отличие от статических инструкций SQL, которые жестко запрограммированы в программе, динамические инструкции SQL могут быть построены во время выполнения и помещены в переменную кода приложения. Затем они отправляются в СУБД для обработки.

ФРАГМЕНТ КОДА СО СТАТИЧЕСКИМ ВНЕДРЕННЫМ SQL

```
int main() {
   EXEC SQL INCLUDE SQLCA;
   EXEC SQL BEGIN DECLARE SECTION;
      int OrderID;          /* Employee ID (from user)         */
      int CustID;             /* Retrieved customer ID         */
      char SalesPerson[10]   /* Retrieved salesperson name      */
      char Status[6]         /* Retrieved order status        */
   EXEC SQL END DECLARE SECTION;

   /* Set up error processing */
   EXEC SQL WHENEVER SQLERROR GOTO query_error;
   EXEC SQL WHENEVER NOT FOUND GOTO bad_number;

   /* Prompt the user for order number */
   printf ("Enter order number: ");
   scanf_s("%d", &OrderID);

   /* Execute the SQL query */
   EXEC SQL SELECT CustID, SalesPerson, Status
      FROM Orders
      WHERE OrderID = :OrderID
      INTO :CustID, :SalesPerson, :Status;
```

```
   /* Display the results */
   printf ("Customer number:  %d\n", CustID);
   printf ("Salesperson: %s\n", SalesPerson);
   printf ("Status: %s\n", Status);
   exit();

query_error:
   printf ("SQL error: %ld\n", sqlca->sqlcode);
   exit();

bad_number:
   printf ("Invalid order number.\n");
   exit();
}
```

КОММЕНТАРИЙ К ФРАГМЕНТУ КОДА

- Переменные узла объявляются в разделе, заключенном в ключевые слова **BeginDeclareSection** и **EndDeclareSection**. Каждое имя переменной узла имеет префикс с двоеточием (:). При появлении в внедренной инструкции SQL. Двоеточие позволяет предкомпилятору различать переменные программы и объекты базы данных, такие как таблицы и столбцы, имеющие одинаковое имя.

- Типы данных, поддерживаемые СУБД и основным языком, могут различаться. Если тип языка программы, соответствующий типу данных СУБД, отсутствует, СУБД автоматически преобразует данные.

- Инструкция, используемая для возврата данных, является одноэлементной инструкцией SELECT. В результате возвращается только одна строка данных. Поэтому в примере кода не объявляются и не используются курсоры.

ФРАГМЕНТ КОДА С ИСПОЛЬЗОВАНИЕМ БИБЛИОТЕКИ, КЛАССА И ПУБЛИЧНЫХ ФУНКЦИЙ

```
{
    QSqlQuery *query = new QSqlQuery();

    query->prepare("INSERT INTO product VALUES"
                   ":name,:category)");
    query->bindValue(":name", ui->lineEdit->text());
    query->bindValue(":category", ui->lineEdit_2->text());

    query->exec();

    close(); //закрытие диалогового окна
}
```

КУРСОРЫ SQL

Операции в реляционной базе данных выполняются над множеством строк. Например, набор строк, возвращаемый инструкцией SELECT, содержит все строки, которые удовлетворяют условиям, указанным в предложении WHERE инструкции. Такой полный набор строк, возвращаемых инструкцией, называется результирующим набором.

Приложения, особенно интерактивные, не всегда эффективно работают с результирующим набором как с единым целым. Им нужен инструмент, позволяющий обрабатывать одну строку или небольшое их число за один раз. Курсоры являются расширением результирующих наборов, которые предоставляют такой механизм.

КУРСОРЫ SQL. СТАТИЧЕСКИЙ КУРСОР.

- Статический курсор всегда отображает результирующий набор точно в том виде, в котором он был при открытии курсора.
- Курсор не отражает изменения в базе данных, влияющие на вхождение в результирующий набор или изменяющие значения в столбцах строк, составляющих набор строк. Статический курсор не отображает новые строки, вставленные в базу данных после открытия курсора, даже если они соответствуют критериям поиска инструкции SELECT курсора.
- Статический курсор продолжает отображать строки, удаленные из базы данных после открытия курсора.
- Статический курсор всегда доступен только для чтения.

КУРСОРЫ SQL. ДИНАМИЧЕСКИЙ КУРСОР.

- Динамические курсоры отражают все изменения строк в результирующем наборе при прокрутке курсора. Значения типа данных, порядок и членство строк в результирующем наборе могут меняться для каждой выборки. Все инструкции UPDATE, INSERT, DELETE, выполняемые пользователями, видимы посредством курсора.

- Обновления, сделанные вне курсора, не видны до момента фиксации, если только уровень изоляции транзакций с курсорами не имеет значение READ UNCOMMITTED.

РАБОТА СО СТАТИЧЕСКИМ КУРСОРОМ (ПРИМЕР)

```
DECLARE vend_cursor CURSOR
    FOR SELECT * FROM Purchasing.Vendor
OPEN vend_cursor
FETCH NEXT FROM vend_cursor;
```

САМОСТОЯТЕЛЬНОЕ ЗАДАНИЕ (КНИГИ)

- Системы баз данных (полный курс), стр. 349-374.

СПАСИБО!

ВАШИ ВОПРОСЫ, ПОЖАЛУЙСТА?