Что такое InterSystems IRIS(Cache`)?

(простым языком)

Платформа данных InterSystems IRIS ™ - это платформа для быстрой разработки и развертывания важных приложений. Все необходимые инструменты и возможности предоставляются в надежной, унифицированной платформе, охватывающей управление данными, совместимость, обработку транзакций и аналитику.

Функции, доступные в InterSystems IRIS, включают в себя:

- Масштабируемая многомодельная база данных (Доступ к данным: SQL,Object,Bin)
- Встроенная аналитическая платформа для реагирования на ваши данные в режиме реального времени(OLAP-кубы)
- Инструменты обработки естественного языка для анализа неструктурированных данных (iKnow)
- Инструменты интеграции и взаимодействия
- Автоматизированные механизмы для легкого развертывания

Платформа IRIS (Cache`) в качестве языка написания кода использует MUMPS(M-система) Определение MUMPS как языка программирования условно, так как он зародился во времена кристаллизации языков программирования, и сфера его применения простирается от работы с медицинским оборудованием до операционных СУБД и экспертных систем (например, VA VistA).

MUMPS зародился в специфичной медицинской среде кардиологов (Octo Barnett). Изначально разработан с целью облегчить написание приложений баз данных, одновременно с максимальным использованием компьютерных ресурсов. Часто использовался в лечебных учреждениях и для финансовых информационных систем и баз данных (особенно для разработанных в 1970-х и начале 1980-х годов) и продолжает использоваться у ряда клиентов. Сейчас MUMPS является базовым языком в системах GT.M, MiniM, YottaDB, DSM, MSM, M3-LITE. Наиболее крупной и известной реализацией MUMPS языка является объектно-ориентированная СУБД Caché (с объектно-ориентированным расширением M-языка - ObjectScript) и IRIS Data Platform.

Одним из главной спецификой языка является система хранения глобальных многомерных массивов на жестком диске(глобалы). Структура глобальных массивов — уникальная особенность языка MUMPS. Эти массивы автоматически записываются на диск, без необходимости резервирования и адресации пространства для записи, относятся к иерархическим и разрежённым структурам. В массивах может содержаться произвольное количество уровней иерархии, причём данные могут содержаться на любом из уровней массива. Индексом массива может быть любое корректное MUMPS выражение, результатом оценки которого может быть число, или строка символов. Глобальные массивы в MUMPS системах используются для построения и управления базами данных. Все таблицы сохраняются на жесткий диск в виде массива и ссылок на другие массивы. В IRIS реализованы функции получения данных из глобальных массивов с использованием языка SQL. Текст SQL запроса конвертируется в MUMPS программу, которой закладывается механизм перемещения курсора по массиву на жестком диске и сбора результата по условию, описанному в запросе. Генерация алгоритма сбора данных происходит один раз, в момент написания запроса, дальнейшее использование (пользователем) не несет за собой потерю ресурсов системы, так как программа не создает временные таблицы, в которые перегружает данные (результат), и не генерирует никакие сопутствующие элементы. За счет этого ускоряется процесс получения данных.

В IRIS реализован доступ к данным через объектную модель, которая имеет механизм сохранения своего экземпляра в глобалы. При таком подходе программист, при разработке оперирует не реляционными данными, а применяет объектно-ориентированный подход (комбинированный SQL+Объекты).

MUMPS является интерпретирующим языком. Команды могут запускаться на исполнение прямо с клавиатуры или записываться в виде программ на диск. MUMPS интерпретирует типы данных в контексте совершаемых над ними операций. В непосредственном режиме пользователь легко может проверить любую из языковых конструкций, с помощью отладчика провести отладку программ и их процедур, просмотреть и изменить в любой момент содержание локальных и глобальных переменных.

Глобаль — массив, автоматически адресуемый для записи на диск. Основная форма записи данных MUMPS. Глобали являются разрежёнными массивами и занимаемое ими пространство на диске определяется только размером данных, записываемых при индексах. Глобали могут быть распределены одновременно между многими пользователями. Глобальные переменные доступны всем процессам работающим в одном каталоге(области имен). Индексация — организация взаимного расположения записей внутри массива называется индексацией и достигается с помощью индексов глобальных массивов. Индексами в этих массивах могут быть не только числа, но и любые символьные строки, имеющие значение (семантику), которая затем используется в программах. Например, это может быть программный код обработки узла. Индексы в массиве выстраиваются по возрастанию в момент присваивания(инициализации) значения узла.

В большинстве MUMPS реализаций глобальные структуры основываются на концепции так называемых сбалансированных деревьев (именуемых также и B-деревьями). B-деревья представляют собой мощный инструмент для организации разрежённых структур с использованием ключей (индексов). При этом обеспечивается эффективный механизм записи и чтения данных с минимальным количеством обращений к диску.

Этот язык является строчно-ориентированным языком, каждая строка может содержать много операторов.

В публикациях сторонников MUMPS утверждается, что эта технология является малоизвестной и существенно менее распространённой за пределами больших корпоративных систем. Новые приложения баз данных чаще всего создаются с использованием поддерживаемого в современных реализациях MUMPS SQL и популярных языков программирования.

Критики M-систем прямо называют эту технологию устаревшей и указывают на такие недостатки как:

- отсутствие типизации (все данные хранятся как строки);
- низкий уровень абстракции;
- нечитабельность синтаксиса, особенно при использовании сокращений.

Язык MUMPS критики называют провоцирующим ошибки, поскольку:

- Отсутствует обязательное объявление (декларирование) переменных;
- Не поддерживаются привычные приоритеты арифметических операций (например, выражение 2+3×10 даёт в значение 50);
- Лишний пробел или разрыв строки может совершенно изменить смысл синтаксической конструкции;
- Ключевые слова языка не зарезервированы и могут широко использоваться в качестве идентификаторов.

В защиту этого языка можно сказать: Обязательное декларирование переменных отсутствует не только в MUMPS, но и в других популярных интерпретируемых языках как Python,JavaScript , и .т.д. Что не ставит все эти языки в разряд устаревших. Многие другие претензиям по языку MUMPS тоже встречаются в современных интерпретируемых языках программирования, но это не мешает им быть современными и развивающимися.

В поставку платформы IRIS входит:

Встроенный Apache,REST;

Шлюз позволяющий интегрироваться с серверами IIS,Apache;

Среда разработки CacheStudio (WIN);

ODBC шлюз;

Библиотеки прямого доступа для :Java,Cpp,C, dotnet,ldap, node.js, perl, python;

Встроенный механизм взаимодействия MUMPS с подключаемыми библиотеками(*.dll,*.o);

Так же при необходимости есть возможность реализовать абсолютно любой механизм взаимодействия на языке программирования MUMPS, в плоть до собственного вэб сервера. Пример такой реализации https://github.com/MyasnikovIA/CacheApache

Компания Intersystems бесплатно предоставляет однопользовательскую версию **Caché** для обучения (https://download.intersystems.com/download/download.csp). Перед скачиванием необходимо зарегистрироваться.

Разработка приложений на Caché(IRIS)

Основное средство разработки в этой системе — интегрированная с СУБД GUI-среда Caché Studio. В ней можно создавать различные классы объектов путем описания их определений, включая свойства и методы. Эти определения задаются с помощью форм и меню (встроенных мастеров) или напрямую на языке описания объектов. В дополнение к этому Caché Studio можно использовать для создания программ на встроенных языках Caché Object Script и Caché Basic, а также CSP-страниц (Caché Server Pages — технология создания Web-интерфейса для приложений Caché).

Объектная модель построена на основе стандарта ODMG (Object Data Management Group). При создании, сохранении и манипулировании экземплярами объектов Caché поддерживает основные концепции современной объектной технологии с инкапсуляцией, полиморфизмом и наследованием. В объектной модели Caché реализовано также множественное наследование, при котором класс порождается сразу несколькими родительскими классами.

Наибольший интерес для нас представляют хранимые классы, которые наследуются от системного класса %Persistent. Объекты этого класса и размещаются в базе данных. В Caché поддерживается несколько способов хранения объектов:

- автоматическое хранение в многомерной базе данных Caché;
- хранение в любых структурах, определенных пользователем;
- хранение во внешних таблицах реляционных баз данных, доступных через шлюз Caché Gateway.

Для доступа к объектам можно использовать уникальные идентификаторы объектов, а также SQL-запросы, причем значением атрибута могут быть как простые типы данных, так и экземпляры встраиваемых классов:

```
SELECT Company->Name FROM Sample.Employee
ORDER BY Company->Name
```

Встраиваемые классы наследуются от класса %SerialObject. Их экземпляры существуют в памяти как независимые объекты, однако сохраняются в базе данных, лишь будучи встроенными в другие объекты. Кроме того, свойствами объектов могут быть ссылки на объекты, потоки данных, различные виды коллекций, многомерные переменные и двунаправленные связи между хранимыми объектами.

Внешние интерфейсы Caché

Команда программистов будет работать более продуктивно, имея дело со знакомой средой разработки. Предоставляя объектный доступ и SQL, Caché поддерживает интерфейсы для большинства распространенных средств разработки. Это позволяет выбрать ту модель данных, которая наилучшим образом соответствует требованиям задачи и условиям, в которых ведется разработка.

В Caché существует широкий спектр интерфейсов, отвечающих за быстрый доступ к объектам Caché из разнообразных сред, включая Java, C++, COM, .NET, EJB, Perl, Python и Delphi. Компоненты Caché отвечают за весь сетевой обмен между клиентом и сервером приложений. Caché обеспечивает двустороннее взаимодействие по протоколу SOAP и поддерживает XML-обмен. Рассмотрим эти способы подробнее.

Доступ через ODBC. Этот внешний реляционный доступ обеспечивается сервером Caché. Вместе с Caché поставляется ODBC-драйвер (InterSystems ODBC), который позволяет внешним приложениям выполнять SQL-запросы к данным Caché как к реляционным таблицам. Этот вид доступа подходит, например, для получения данных для генераторов отчетов, аналитических пакетов и других реляционных приложений. Использование реляционного доступа к Caché по протоколу ODBC мало чем отличается от использования ODBC для других СУБД — за исключением, пожалуй, возможности добиться выигрыша по скорости.

Объектное взаимодействие с Java. Язык программирования Java весьма популярен, но связь написанных на нем приложений с большинством баз данных — непростая задача. Подключение к реляционной базе данных требует отображения объектного представления в плоские таблицы. Написание большого объема кода на SQL занимает очень много времени и лишает преимуществ объектную технологию Java. Возможность работать с объектами Caché напрямую избавляет разработчика от забот, связанных с предоставлением описания данных для хранения, а использовать объекты для работы с базой данных гораздо проще и предпочтительней. Java-приложение может взаимодействовать с Caché следующими способами:

- для любого класса Caché можно создать его проекцию как класс Java, так что свойства и методы этого класса будут доступны, как если бы это был объект Java;
- классы Caché можно проецировать как Enterprise Java Beans;
- высокопроизводительный доступ для SQL полностью обеспечивается "родным" драйвером JDBC на Java, по аналогии с другими СУБД;
- технология InterSystems Jalapeno создает классы Caché из описаний классов POJO.

Объектный доступ через проекции классов. Для каждого класса Caché можно создать соответствующую проекцию — класс Java (или EJB) с методами, соответствующими каждому свойству и методу класса Caché. Для программ на Java эти классы ничем не отличаются от других классов Java-приложения. Сгенерированные классы Java используют библиотеку Java из поставки InterSystems для связи клиента с сервером.

Сервер приложений Caché хранит состояние каждого объекта Caché, в то время как свойства класса также кэшируются клиентом для обеспечения оптимальной производительности. Вызовы методов Java вызывают соответствующие методы на сервере приложений Caché, включая методы сохранения и открытия объекта. Для клиента эти процессы полностью прозрачны — для него не играет роли, с каким сервером данных Caché происходит обмен данными, даже если сервер приложений Caché работает с реляционной базой данных.

Методы Caché, написанные на Java. Методы классов Caché могут быть написаны на Java в интегрированной среде разработки Caché Studio. Но, в отличие от Caché ObjectScript и Basic, методы Java не выполняются на сервере Caché — вместо этого они включаются в генерируемый класс Java и выполняются на любой виртуальной машине Java.

Предоставление механизма хранения для приложений J2EE. Разработчики приложений J2EE, использующие EJB (Enterprise Java Beans), работают с объектами в основном до момента доступа к базе данных; после этого они должны использовать SQL. Caché может предоставить для таких приложений быстрый SQL-доступ через интерфейс JDBC. В то же время доступ с использованием SQL далеко не всегда оказывается предпочтительным.

Объектные базы данных представляют более естественную технику доступа для программистов EJB. Caché проецирует классы Caché как EJB, автоматически генерируя высокопроизводительные методы хранения для BMP (Bean-Managed Persistence). Это позволяет обойтись без лишнего кодирования при использовании SQL и объектно-реляционного отображения.

Описание классов Java в Jalapeno. Вместо описания классов в Caché и создания их проекции как компонента Java технология InterSystems Jalapeno предлагает другой подход, который позволяет разработчикам на Java описывать классы в привычной среде разработки Java и автоматически создавать описания для хранения этих классов в Caché. При этом классы, разработанные в Java, остаются без

изменений, а Caché предоставляет библиотеку с API, обеспечивая механизм хранения и запросов по этим классам.

Caché и .NET

На сегодняшний день широко распространена разработка приложений на платформе .NET. Благодаря открытому и гибкому механизму доступа к данным Caché предоставляет различные интерфейсы к .NET — SQL, объектный, SOAP — и обеспечивает целостное взаимодействие с платформой .NET Framework. Разработчик может выбирать наиболее подходящую для себя технологию — с любой из них будет достигнута высокая производительность и масштабируемость Caché.

Основной механизм, предоставляющий доступ к Caché из .NET-приложений, — это Caché Managed Provider для .NET. Особенность этого механизма заключается в том, что он, используя единый интерфейс, поддерживает реляционный и объектный доступ, не требуя создания проекции объектов на таблицы. Caché Managed Provider обеспечивает реляционный доступ к данным, используя API ADO.NET, а также высокопроизводительный и удобный объектный доступ к данным через автоматически генерируемые промежуточные классы. Эти промежуточные классы соответствуют классам Caché и обеспечивают хранение объектов, извлечение и кэширование данных, управление жизненным циклом объектов. За счет того, что Caché Managed Provider реализован как управляемый .NET-код, достигается простота развертывания в среде .NET. И наконец, Caché Managed Provider для .NET можно использовать в многопоточных (multi-threaded) .NET-приложениях, так как он безопасен для многопоточных процессов.

Для работы Caché Managed Provider используется системная библиотека InterSystems.Data.CacheClient.dll, которая содержит два пространства имен: Intersystems.Data.CacheClient и Intersystems.Data.CacheTypes.

Первое из них, Intersystems.Data.CacheClient, представляет собой стандартизованный интерфейс ADO.NET, который содержит классы, полностью соответствующие аналогичным классам других провайдеров, таких, как Microsoft System.Data.OleDb или System.Data.SqlClient. В рамках Caché Managed Provider реализованы CacheClient-версии стандартных ADO-классов (например, CacheCommand, CacheConnection, CacheDataAdapter и CacheDataReader). Они применяются точно так же, как аналогичные классы для работы с другими базами данных, поэтому разработчик, используя привычные для него ADO-команды, может получать доступ к данным Caché и обрабатывать их, не обладая никакими специальными знаниями о синтаксисе и командах Caché.

Второе пространство имен, Intersystems.Data.CacheTypes, включает в себя специальные средства для работы с объектами Caché. Классы CacheTypes позволяют работать с объектами .NET, которые представляют собой точные аналоги объектов Caché, и могут использоваться так же, как они используются во встроенных языках Caché — ObjectScript или Caché Basic. В отличие от стандартного ADO-интерфейса прокси-объекты Caché обеспечивают прямое управление объектами и структурами данных Caché.

Таким образом, Caché Managed Provider — это компонент, который обеспечивает одновременно и реляционный, и объектный доступ к данным, используя общий программный интерфейс.

Реляционный доступ к данным

Caché Managed Provider предоставляет стандартный набор классов, таких, как Connection, Command, DataAdapter и т. п. для взаимодействия с Caché (рис. 1). Ниже приведены краткие описания некоторых классов и примеры их использования.

Рис. 1. Схема доступа к данным Caché из ADO.NET.

CacheConnection — этот класс предназначен для соединения с источником данных (т. е. Caché) и содержит свойства, необходимые для подключения. Пример установления соединения:

```
String cacheConnectString = "Server =localhost; Log File=c:\\cprovider.log;Port=1972;amespace=Samples; Password =
SYS; User ID = _SYSTEM;";
cnCache = new CacheConnection(cacheConnectString);
cnCache.Open();
…
```

CacheDataAdapter — этот класс включает в себя набор команд, а также соединения с источником данных, которые используются для получения данных из источника и помещения их в объект .NET типа System.Data.DataSet. Может также использоваться для обновления данных источника. Пример заполнения DataSet:

```
dsCache= new DataSet();
daCache= new CacheDataAdapter();
```

```
daCache.SelectCommand= new CacheCommand("Select ID, Name, DOB from Sample.Person",
cnCache, txCache);
daCache.Fill(dsCache, "Sample_Person");
…
```

CacheCommand — класс, представляющий собой команды SQL или хранимые процедуры, которые необходимо выполнить. Пример обновления данных в БД через запрос с параметром (CacheParameter):

```
spCache = new CacheCommand("Update Sample.Person Set Name = ?  where ID  = ?", cnCache, txCache);
 CacheParameter pName = new CacheParameter();
 pName.ParameterName= "Name";
 pName.CacheDbType= CacheDbType.NVarChar;
 pName.Direction = ParameterDirection.Input;
 pName.Value = txtName.Text;
 …
spCache.Parameters.Add(pName);
spCache.Parameters.Add(pID);
spCache.ExecuteNonQuery();
```

Кроме описанных классов имеются и другие: как стандартные (например, CacheDataReader), так и специальные, предназначенные для удобства работы с Caché. При этом в общем случае, как видно из примеров, работа с данными на основе реляционного доступа аналогична работе средствами ADO с другими источниками данных.

Объектный доступ к данным

Одна из важнейших особенностей Caché — возможность объектного доступа к данным за счет использования Caché Managed Provider, а не представление данных в виде строк реляционных таблиц. В результате в .NET-приложении используются абсолютно те же классы, что и были описаны в Caché. Простейший пример работы с объектами Caché в .NET-приложении выглядит так:

```
CacheConnection CacheConnect = new CacheConnection();
CacheConnect.ConnectionString = "Server = localhost; "
             + "Port = 1972; " + "Namespace = SAMPLES; "
             + "Password = sys; " + "User ID = _system;";
CacheConnect.Open();
Sample.Person person = Sample.Person.OpenId(CacheConnect, "1");
Console.WriteLine("TinyProxy output: \r\n "+ person.Id() + ": "+ person.Name);
person.Close();
CacheConnect.Close();
```

Очевидно, что можно реализовать полностью аналогичный код, используя реляционный способ работы с данными, но также очевидно, что работа в объектной модели имеет ряд существенных преимуществ. Главные из них — простота и скорость реализации: объектный код достаточно понятен, что упрощает его модификацию и сопровождение.

Механизм Caché Managed Provider работает следующим образом (рис. 2). Сначала нужно описать необходимые классы в Caché Studio. Это могут быть классы, хранимые в базе данных Caché, или классы для описания бизнес-логики в Caché. Программа Caché Object Binding Wizard создает прокси-классы .NET на основе классов, определенных в библиотеке классов Caché. Эти классы реализованы при помощи managed code и содержат вызовы методов и ссылки на свойства объектов Caché.

Во время выполнения .NET-приложение подключается к серверу Caché; при этом можно создавать экземпляры .NET-объектов, которые связаны с объектами на сервере Caché. Эти объекты используют точно так же, как другие объекты .NET .

Caché автоматически управляет всеми взаимодействиями и при этом поддерживает кэширование данных на стороне клиента. Таким образом, д??я работы приложения необходимы только сервер Caché и клиентские .NET-приложения, которые взаимодействуют с сервером через стандартный TCP/IP протокол. Для работы механизма Caché Managed Provider для .NET на сервере должна быть установлена СУБД Caché начиная с версии 5.1, на клиенте — Microsoft .NET Framework v2.0. Для разработки .NET-приложений требуется Microsoft Visual Studio 2005.

Предоставление информации через Web-сервисы

Для связи с .NET по протоколу SOAP СУБД Caché используется как сервер Web-сервисов, а на стороне .NET создается SOAP-клиент для работы с сервисом Caché. Любой метод класса Caché, хранимую процедуру или запрос можно автоматически представить как Web-сервис. Caché генерирует описание сервиса в файл WSDL и при вызове сервиса отправляет соответствующий документ XML.

Организовать работу Caché как сервера Web-сервисов несложно. Для этого используются специальные классы Caché, которые наследуются от системного класса %SOAP.WebService. Методы, которые должны будут предоставляться Web-сервисом, описываются так же, как обычные методы Caché, но для того чтобы эти методы были доступны внешним приложениям, для них следует указать ключевое слово WebMethod:

```
ClassMethod Test() As %String [ WebMethod ]
{
     Quit "Test"
}
```

После компиляции класса Web-сервис готов к использованию, и его работу можно проверить в Caché, просматривая соответствующую Web-страницу в браузере.

Чтобы использовать методы Web-сервиса Caché в .NET, нужно в проект Visual Studio.NET добавить Web-ссылку (Web reference), указав адрес WSDL-документа Web-сервиса. После этого в приложении .NET будет создан класс, соответствующий Web-сервису Caché. Создав новый экземпляр этого класса, разработчик получает доступ к его методам, которые представляют собой отображение соответствующих методов Web-сервиса Caché. Методы Web-сервиса Caché могут возвращать значения следующих типов:

- %String — строка (при использовании строки с разделителями следует обратить внимание на ограничения XML, касающиеся передачи некоторых специальных символов);
- %ListOfDataTypes — список значений;
- %XML.DataSet — нетипизированный DataSet, который формируется как результат выполнения запроса.

Использование типа DataSet может выглядеть так:

```
ClassMethod GetByName() As %XML.DataSet [ WebMethod ]
{
           Set ds=##class(%XML.DataSet).%New("User.Person:All")
           Do ds.SetArgs()
           Quit dst
}
```

Пример описания объекта пользовательского класса:

```
ClassMethod TestObj(pID as %Integer)
As Sample.Person [ WebMethod ]
{
           Quit ##class(Sample.Person).%OpenId(pID)
}
```

Соответствие между перечисленными выше типами значений методов Web-сервиса Caché и методов в .NET описано в таблице.

Типы значений методов Web-сервиса Caché и соответствующие им значения в .NET

Caché	.NET
%String	String — строка
%ListOfDataTypes	String() — массив строк
%XML.DataSet	DataSet
Объект пользовательского класса	Object

При использовании %XML.DataSet результат запроса из Caché попадает в .NET в виде DataSet, содержащего единственную таблицу с данными.

При передаче объекта пользовательского класса в .NET разработчик получает значения всех свойств данного объекта Caché, в том числе и сложных, которые представляют собой ссылки на объекты других классов (например, person.adress.street). Таким образом, в общем случае в клиентском приложении имеется полная информация об объекте в виде некой иерархической структуры.

Такой способ передачи данных достаточно прост, но у него есть существенный недостаток: поскольку при использовании Web-сервисов не устанавливается постоянная связь с базой данных, невозможно напрямую вызвать методы, отвечающие за сохранение и удаление данных со стороны клиента. Впрочем, есть простой способ решения этой проблемы — всегда можно создать новый серверный метод Caché, который будет использовать соответствующие методы на стороне сервера Caché.

Заключение

СУБД Caché предоставляет как объектный, так и реляционный доступ к данным для разных средств разработки приложений. Для каждого приложения команды программистов могут выбрать ту модель данных и те интерфейсы, которые наиболее полно отвечают требованиям задачи и условиям разработки. Поддержка средств разработки Microsoft в Caché имеет очень давнюю историю — от использования COM-объектов и поддержки C++-binding до Caché Managed объектов для Framework 2.0.

Для работы с Caché можно использовать сразу три способа доступа к одним и тем же данным в рамках одного приложения (в том числе и «прямой» доступ к многомерным массивам данных). При этом объектный и реляционный доступ будут равноправными.

Материалы для разработчиков:

Учебный курс по Cache` Intersystems

http://www.smwrap.ru/doc/UchebniyKurs.pdf

http://www.smwrap.ru/doc/UchebniyKurs2.pdf

Описание языка MUMPS

http://www.smwrap.ru/doc/MUMPS.pdf

Разработка приложений для СУБД Caché(статья)

Описание среды компонентной разработки приложений (аналог Delphi)

http://grigory.ru/archiv/pro/2001_06_SMWrap/www/index.html

новая версия www.smwrap.ru

http://www.smwrap.ru/SubdCache.html

3 вида доступа данным к СУБД InterSystems Caché

https://www.youtube.com/watch?v=Aeb_mxVEQ_A

Вебинар "Управление данными интеграционных решений"

https://www.youtube.com/watch?v=iVY5D2l2Rq0

Вебинар "Оптимизация производительности приложений"

https://www.youtube.com/watch?v=7hc1NgYRvSo

Вебинар "Шардинг и основы масштабирования в InterSystems IRIS"

(объедиенение серверов в единых пул)

https://www.youtube.com/watch?v=2LuJHt9z6NM&t=75s

Вебинар "Continuous Delivery с использованием контейнеров"

https://www.youtube.com/watch?v=hoD4IpwXHrM