Artículo
Dmitrii Kuznetsov · 19 sep, 2022 Lectura de 17 min

Cuatro APIs de base de datos

Una sesión simultánea en IRIS: SQL, Objects, REST y GraphQL


Kazimir Malevich, "Deportistas" (1932)

> *"¡Pues claro que no lo entiende! ¿Cómo puede una persona que siempre ha viajado en un carruaje tirado por caballos entender los sentimientos e impresiones del viajero expres o del piloto de aviones?"*
>
> Kazimir Malevich (1916)

Introducción

Ya hemos abordado el tema de por qué la representación de objetos/clases es superior a la de SQL

para implementar modelos de áreas temáticas. Y esas conclusiones y hechos son tan ciertos ahora como lo han sido siempre. Entonces, ¿por qué deberíamos dar un paso atrás y hablar sobre las tecnologías que arrastran las abstracciones de vuelta al nivel global, donde habían estado en la era pre-objetos y pre-clases? ¿Por qué debemos fomentar el uso de "código espagueti", que provoca errores que son difíciles de rastrear y se basa en las habilidades de desarrolladores virtuosos?

Hay varios argumentos a favor de la transmisión de datos por medio de APIs basadas en SQL/REST/GraphQL en lugar de representarlos como clases/objetos:

- Estas tecnologías están muy bien estudiadas y son bastante fáciles de implementar
- Disfrutan de una increíble popularidad y han sido ampliamente implementadas en software accesible y de código abierto
- Con frecuencia no hay otra alternativa que utilizar estas tecnologías, especialmente en la web y en las bases de datos
- Lo más importante es que las APIs siguen utilizando objetos, ya que ofrecen la forma más apropiada de implementar las APIs en el código

Antes de hablar de la implementación de las APIs, vamos a echar un vistazo a las capas de abstracción por debajo. El siguiente diagrama muestra cómo se mueven los datos entre el lugar donde se almacenan permanentemente y el lugar donde se procesan y se presentan al usuario de nuestras aplicaciones.

Actualmente, los datos se almacenan en discos duros giratorios (HDD) o, para utilizar una tecnología más moderna, en chips de memoria flash en un SSD. Los datos se escriben y se leen utilizando un flujo que consiste en bloques de almacenamiento separados en el HDD/SSD.

La división en bloques no es aleatoria. Más bien, está determinado por la física, la mecánica y la electrónica del medio de almacenamiento de datos. En un disco duro, estas son las pistas/sectores de un disco magnético giratorio. En un SSD, son los segmentos de memoria en un chip de silicio regrabable.

La esencia es la misma: son bloques de información que debemos encontrar y reunir para recuperar los datos que necesitamos. Los datos deben ensamblarse en estructuras que emparejen nuestro modelo/clase de datos con los valores que corresponden al momento de la consulta. El DBMS (Sistema de Administración de Bases de Datos) integrado en el subsistema de archivos del sistema operativo es el responsable del proceso de ensamblaje y recuperación de datos.

Podemos evitar el DBMS dirigiéndonos directamente al sistema de archivos o incluso al HDD/SSD. Pero entonces perdemos dos puentes súper importantes para los datos: entre los bloques de almacenamiento y los flujos de archivos; y entre los archivos y la estructura ordenada en el modelo de base de datos. En otras palabras, asumimos la responsabilidad del desarrollo de todo el código para procesar bloques, archivos y modelos, incluyendo todas las optimizaciones, la depuración minuciosa y las pruebas de fiabilidad a largo plazo.

El DBMS nos ofrece una excelente oportunidad para tratar los datos en un lenguaje de alto nivel, utilizando modelos y representaciones comprensibles. Esta es una de las grandes ventajas de estos sistemas. Los DBMS y las plataformas de datos, como InterSystems IRIS, ofrecen aún más: la capacidad de acceder simultáneamente a los datos ordenados de muy diversas maneras. Y cada uno decide cuál utilizar en su proyecto.

Vamos a aprovechar la variedad de herramientas que nos ofrece IRIS. Vamos a hacer el código más atractivo y limpio. Utilizaremos directamente ObjectScript, el lenguaje orientado a objetos, para utilizar y desarrollar APIs. En otras palabras, por ejemplo, llamaremos al código SQL directamente desde el software ObjectScript. Para otras APIs, utilizaremos librerías listas y herramientas integradas en ObjectScript.

Tomaremos nuestros ejemplos del proyecto de Internet SQLZoo, que ofrece recursos de aprendizaje para SQL. Utilizaremos los mismos datos en nuestros otros ejemplos de API.

> Si quieres obtener una visión general de la variedad de enfoques para el diseño de API y aprovechar las soluciones ya hechas, esta es una interesante y útil colección de APIs públicas, que han sido reunidas en un único proyecto en GitHub.

SQL

No hay una forma más natural de empezar que con SQL. ¿Quién no lo conoce?

Hay un enorme cantidad de tutoriales y libros sobre SQL. Nos basaremos en SQLZoo. Es un buen curso de SQL para principiantes, con ejemplos, tutoriales y una referencia del lenguaje SQL.

Vamos a trasladar algunas tareas de SQLZoo a la plataforma de IRIS y a resolverlas utilizando diversos métodos.

> ¿Con qué rapidez puedes acceder a InterSystems IRIS en tu ordenador? Una de las opciones más rápidas es implementar un contenedor en Docker a partir de una imagen ya preparada de la Community Edition de InterSystems IRIS. La Community Edition es una versión gratuita de InterSystems IRIS Data Platform para desarrolladores.

Otras maneras de acceder a la Community Edition de InterSystems IRIS en el Portal de Formación.

Para pasar los datos de SQLZoo a nuestro propia instancia de almacenamiento de IRIS.

Para hacer esto:

- Abre el Portal de Administración (el mío, por ejemplo, está en http://localhost:52773/csp/sys/UtilHome.csp),
- Cambia al área USER: en el Namespace %SYS haz clic en el enlace “ Switch ” y selecciona USER
- Ve a System > SQL - abre el Explorador del Sistema, luego SQL y haz clic en el botón “ Go ”.
- En la parte derecha se abrirá la pestaña "Execute query" con el botón "Execute", que es lo que necesitamos.

Para obtener más información sobre cómo trabajar con SQL a través del Portal de Administración, consulta esta documentación.

Echa un vistazo a los scripts listos para implementar la base de datos y el conjunto de datos de prueba de SQLZoo en la descripción de la sección Datos.

Aquí hay un par de enlaces directos para la tabla world:

- Un script que se utiliza para crear la base de datos world
- Los datos que entran en esa tabla

El script para crear la base de datos se puede ejecutar en el formulario Query Executor que está en el Portal de administración de IRIS.

```
CREATE TABLE world(
   name VARCHAR(50) NOT NULL
   ,continent VARCHAR(60)
   ,area DECIMAL(10)
   ,population DECIMAL(11)
   ,gdp DECIMAL(14)
   ,capital VARCHAR(60)
   ,tld VARCHAR(5)
   ,flag VARCHAR(255)
   ,PRIMARY KEY (name)
)
```

Para cargar el conjunto de pruebas para el formulario Query Executor, ve al menú Wizards > Data Import. Ten en cuenta que el directorio con el archivo de datos de prueba debe añadirse previamente, cuando creas el contenedor, o cargarse desde el ordenador por medio del navegador. Esta opción está disponible en el panel de control del asistente de importación de datos.

Revisa si la tabla con los datos está presente ejecutando este script en el formulario del Query Executor:

```
SELECT * FROM world
```

Ahora podemos acceder a los ejemplos y las tareas desde la página web de SQLZoo. Los siguientes ejemplos requieren que implementes una consulta SQL en la primera asignación:

```
SELECT population
  FROM world
WHERE name = 'France'
```

De esta manera, podrás seguir trabajando sin problemas con la API, transfiriendo tareas de SQLZoo a la plataforma IRIS.

Nota: como he descubierto, los datos en la interfaz de SQLZoo son diferentes de los datos exportados. Al menos en el primer ejemplo, los valores para la población de Francia y Alemania difieren. No te preocupes por eso. Utiliza los datos de Eurostat como referencia.

> Otra forma práctica de obtener acceso SQL a la base de datos en IRIS es el editor de Visual Studio Code con el *plugin* SQL Tools y el driver SQLTools para InterSystems IRIS. Esta solución es muy popular entre los desarrolladores - dale una vuelta.

Con el fin de proceder sin problemas al siguiente paso y obtener acceso a través de objetos a nuestra base de datos, vamos a tomar un pequeño desvío desde las consultas SQL "puras" a las consultas SQL embebidas en el código de la aplicación en ObjectScript, que es un lenguaje orientado a objetos integrado en IRIS.

> Cómo configurar el acceso a IRIS y desarrollar en ObjectScript en VSCode.

```
Class User.worldquery
{
ClassMethod WhereName(name As %String)
{
    &sql(
        SELECT population INTO :population
          FROM world
         WHERE name = :name
    )

    IF SQLCODE<0 {WRITE "SQLCODE error ",SQLCODE," ",%msg  QUIT}
    ELSEIF SQLCODE=100 {WRITE "Query returns no results"  QUIT}
    WRITE name, " ", population
}
}
```

Vamos a comprobar el resultado en el terminal:

```
do ##class(User.worldquery).WhereName("France")
```

Deberías recibir el nombre del país y el número de habitantes como respuesta.

Objetos/Tipos

Ahora, vamos a la historia de REST/GraphQL. Estamos implementando una API para protocolos web. La mayoría de las veces, tendríamos el código fuente oculto en el lado del servidor en un lenguaje que tiene un buen soporte para clases, o incluso un paradigma totalmente orientado a objetos. Estos son algunos de los lenguajes de los que hablamos: Spring en Java/Kotlin, Django en Python, Ruby en Rails, ASP.NET en C#, o Angular en TypeScript. Y, por supuesto, objetos en ObjectScript, que es nativo de la plataforma IRIS.

¿Por qué esto es importante? Las clases y objetos en tu código se simplificarán a estructuras de datos cuando se envían. Hay que tener en cuenta cómo se simplifican los modelos en el programa, lo que es similar a tener en cuenta las pérdidas en los modelos relacionales. También hay que asegurarse de que, al otro lado de la API, los modelos se restauran adecuadamente y se pueden utilizar sin distorsiones. Esto supone una carga adicional: una responsabilidad adicional para ti como programador. Fuera del código y más allá de la ayuda de traductores, compiladores y otras herramientas automáticas, hay que asegurarse continuamente de que los modelos se transfieren correctamente.

Si nos fijamos en el problema anterior desde una perspectiva diferente, todavía no vemos ninguna tecnología ni herramienta que se pueda utilizar para transferir fácilmente clases/objetos de un programa en un lenguaje a un programa en otro. ¿Qué es lo que queda?

Hay implementaciones simplificadas de SQL/REST/GraphQL, y una infinidad de documentación que describen la API en un lenguaje amigable. La documentación más informal (desde la perspectiva del ordenador) para desarrolladores describe exactamente lo que debería ser traducido en código formal utilizando todos los medios disponibles, para que el ordenador pueda procesarlos.

Los programadores desarrollan constantemente diferentes enfoques para resolver los problemas anteriores. Uno de los enfoques exitosos es el paradigma del lenguaje cruzado en el objeto DBMS de la plataforma IRIS.

La siguiente tabla te ayudará a entender la relación entre los modelos OPP y SQL en IRIS:

Programación orientada a objetos (OOP)	Lenguaje de consulta estructurado (SQL)
Paquete	Esquema
Clase	Tabla
Propiedad	Columna
Método	Procedimiento almacenado
Relación entre dos clases	Clave foránea, *join*
Objeto(en memoria o en disco)	Fila (en el disco)

Puedes obtener más información sobre la visualización de objetos y modelos relacionales en la documentación de IRIS.

Al ejecutar nuestra consulta SQL para crear la tabla world a partir del ejemplo anterior, IRIS generará automáticamente las descripciones del objeto correspondiente en la clase denominada User.world.

```
Class User.world Extends %Persistent [ ClassType = persistent, DdlAllowed, Final, Own
er = {_SYSTEM}, ProcedureBlock, SqlRowIdPrivate, SqlTableName = world ]

{

Property name As %Library.String(MAXLEN = 50) [ Required, SqlColumnNumber = 2 ];

Property continent As %Library.String(MAXLEN = 60) [ SqlColumnNumber = 3 ];

Property area As %Library.Numeric(MAXVAL = 9999999999, MINVAL = -9999999999, SCALE =
0) [ SqlColumnNumber = 4 ];

Property population As %Library.Numeric(MAXVAL = 99999999999, MINVAL = -99999999999,
SCALE = 0) [ SqlColumnNumber = 5 ];

Property gdp As %Library.Numeric(MAXVAL = 99999999999999, MINVAL = -99999999999999, S
CALE = 0) [ SqlColumnNumber = 6 ];

Property capital As %Library.String(MAXLEN = 60) [ SqlColumnNumber = 7 ];

Property tld As %Library.String(MAXLEN = 5) [ SqlColumnNumber = 8 ];
```

```
Property flag As %Library.String(MAXLEN = 255) [ SqlColumnNumber = 9 ];

Parameter USEEXTENTSET = 1;

/// Bitmap Extent Index auto-
generated by DDL CREATE TABLE statement.  Do not edit the SqlName of this index.

Index DDLBEIndex [ Extent, SqlName = "%%DDLBEIndex", Type = bitmap ];

/// DDL Primary Key Specification

Index WORLDPKey2 On name [ PrimaryKey, Type = index, Unique ];
}
```

Esta es una plantilla que puedes utilizar para desarrollar tu aplicación en un estilo orientado a objetos. Todo lo que necesitas hacer es añadir métodos a la clase en ObjectScript, que tiene paquetes listos para la base de datos. De hecho, los métodos para esta clase son "procedimientos almacenados", para tomar prestada la terminología SQL.

Vamos a intentar implementar el mismo ejemplo que completamos antes de utilizar SQL. Añade el método WhereName a la clase User.world, que desempeñará el papel del diseñador de objetos "Información del país" para el nombre del país introducido:

```
ClassMethod WhereName(name As %String) As User.world
{
    Set id = 1
    While ( ..%ExistsId(id) ) {
        Set countryInfo = ..%OpenId(id)
        if ( countryInfo.name = name ) { Return countryInfo }
        Set id = id + 1
    }
    Return countryInfo = ""
}
```

Verifica las siguientes líneas de comando en el terminal:

```
set countryInfo = ##class(User.world).WhereName("France")


write countryInfo.name


write countryInfo.population
```

De este ejemplo podemos entender que, para encontrar el objeto deseado por el nombre del país, a diferencia de una consulta SQL, necesitamos ordenar manualmente los registros de la base de datos uno por uno. En el peor de los casos, si nuestro objeto está al final de la lista (o no está allí en absoluto), tendremos que ordenar todos los registros. Hay un debate aparte sobre cómo se puede acelerar el proceso de búsqueda mediante la indexación de campos de objeto y la autogeneración de métodos de clase en IRIS. Puedes leer más en la documentación y en este artículo de la Comunidad de Desarrolladores.

Por ejemplo, para nuestra clase, conociendo el nombre del índice generado por IRIS a partir del nombre del país WORLDPKey2, se puede iniciar/diseñar un objeto desde la base de datos utilizando una sola consulta rápida:

```
set countryInfo = ##class(User.world).WORLDPKey2Open("France")
```

Revisa también:

```
write countryInfo.name write countryInfo.population
```

En esta documentación puedes encontrar algunas pautas para decidir si utilizar acceso a través de objetos o SQL.

Por supuesto, siempre debes tener en cuenta que puede que sólo necesites uno de ellos para realizar tus tareas.

Además, gracias a la disponibilidad de paquetes binarios listos en IRIS que son compatibles con los lenguajes OOP más comunes, como Java, Python, C, C# (.Net), JavaScript, e incluso Julia (consulta GitHub y OpenExchange), que está ganando popularidad rápidamente, siempre podrás elegir las herramientas de desarrollo de lenguajes que más te convengan.

Ahora vamos a profundizar en el debate sobre los datos en la API web.

REST, o la API web RESTful

Vamos a salir de los límites del servidor y del terminal familiar y a utilizar algunos *interfaces* más convencionales: el navegador y aplicaciones similares. Estas aplicaciones dependen de los protocolos de hipertexto de la familia HTTP para administrar las interacciones entre sistemas. IRIS incorpora varias herramientas adecuadas para este fin, incluyendo un servidor de base de datos real y el servidor HTTP Apache.

> La Transferencia de Estado Representacional (REST) es un estilo de arquitectura para diseñar aplicaciones distribuidas y, en particular, aplicaciones web. Aunque es popular, REST solo es un conjunto de principios arquitectónicos, mientras que SOAP es un protocolo estándar preservado por el Consorcio World Wide Web (W3C), y por tanto las tecnologías basadas en SOAP están respaldadas por un estándar.

El ID global en REST es una URL, y define cada unidad de información sucesiva cuando se intercambia con una base de datos o una aplicación *back-end*. Consulta esta documentación para desarrollar servicios REST en IRIS.

En nuestro ejemplo, el identificador base será algo parecido a la base desde la dirección del servidor IRIS, http://localhost:52773, y la ruta /world/ a nuestros datos que es un subdirectorio de ella. En particular, se trata de nuestro directorio de países /world/France.

En un contenedor Docker tendrá un aspecto similar a este:

http://localhost:52773/world/France

Si estás desarrollando una aplicación completa, asegúrate de revisar las recomendaciones de la documentación de IRIS. Una de ellos se basa en la descripción de la API REST, según la especificación OpenAPI 2.0.

Hagamos esto de la manera fácil, implementando la API manualmente. En nuestro ejemplo, crearemos la solución REST más simple, que solo requiere dos pasos en IRIS:

- Crear una clase monitor (API) que se ejecutará al llamar a la URL. Esta clase heredará de la clase del sistema %CSP.REST
- Configurar una aplicación web para que invoque a nuestra clase API cuando se llame a la URL.

Paso 1: Clase API

Debería estar claro cómo se puede implementar una clase. Sigue las instrucciones en la documentación para crear una REST "manual".

```
/// Description

Class User.worldrest Extends %CSP.REST

{

Parameter UseSession As Integer = 1;

Parameter CHARSET = "utf-8";

XData UrlMap [ XMLNamespace = "http://www.intersystems.com/urlmap" ]

{

<Routes>

    <Route Url="/:name" Method="GET" Call="countryInfo" />

</Routes>

}

}
```

Asegúrate de incluir un método que gestione la llamada. Debería realizar exactamente la misma función que las llamadas en el terminal del ejemplo anterior:

```
ClassMethod countryInfo(name As %String) As %Status

{
```

```
    set countryInfo = ##class(User.world).WhereName(name)

    write "Country: ", countryInfo.name

    write "<br>"

    write "Population: ", countryInfo.population

    return $$$OK

}
```

Como se puede ver, el parámetro que empieza con dos puntos, :name,está indicado para que se pase como variable al método que gestiona la llamada.

Paso 2: Configurando la aplicación web IRIS

En System Administration > Security > Applications > Web Applications, añade una nueva aplicación web con una dirección de entrada URL en /world y este handler o manejador: la clase API que has implementado en el paso anterior.

Después de que se configura, la aplicación web debería responder inmediatamente cuando vayas a http://localhost:52773/world/France. Ten en cuenta que la base de datos distingue entre mayúsculas y minúsculas, por lo que se debe utilizar la letra correcta al transmitir los datos de la solicitud al parámetro del método.

> Consejos útiles:
>
> - Utiliza las herramientas de depuración si es necesario. Puedes encontrar una buena descripción en este artículo de dos partes (echa un vistazo a los comentarios también)
> - Si aparece el error "401 Unauthorized", y estás seguro de que la clase monitor está en el servidor y de que no hay errores en el enlace, intenta añadir la función %All en la pestaña Application Roles que se encuentra en la configuración de la aplicación web. Este no es un método totalmente seguro, y es necesario comprender las posibles implicaciones de permitir el acceso para todas las funciones, pero es aceptable para una instalación local.

GraphQL

Este es un terreno nuevo desde el punto de vista de que no encontrarás nada en la documentación actual de IRIS sobre APIs que utilizan GraphQL. Sin embargo, esto no debería impedirnos utilizar esta maravillosa herramienta.

> Hace sólo cinco años que GraphQL salió a la luz. Desarrollado por la Fundación Linux, GraphQL es un lenguaje de consulta para APIs. Y probablemente se puede decir que se trata de la mejor tecnología derivada de la mejora de la arquitectura REST y de las

> distintas APIs web. Aquí hay un breve artículo introductorio para principiantes. Y, gracias a los esfuerzos de los seguidores de InterSystems y a sus ingenieros, IRIS ofrece aplicaciones de ejemplo para GraphQL desde 2018.

Aquí está el artículo relacionado "Cómo implementar GraphQL en las plataformas de InterSystems" .

La aplicación GraphQL se compone de dos módulos: el *backend* de la aplicación en el lado de IRIS y el *frontend* que se ejecuta en el navegador. En otras palabras, es necesario configurarlo de acuerdo con las instrucciones para la aplicación web GraphQL y GraphiQL.

Por ejemplo, así se ve la configuración de la aplicación para mí en IRIS dentro de un contenedor Docker. Estas son las configuraciones para una aplicación web GraphQL que actúa como API REST y un gestor de esquemas de base de datos:

Y la segunda aplicación GraphQL es una interfaz de usuario para el navegador, escrita en HTML y JavaScript:

Se puede ejecutar visitando http://localhost:52773/graphiql/index.html.

Sin ninguna configuración restrictiva adicional, la aplicación inmediatamente buscará todos los esquemas de la base de datos que pueda encontrar en el área de instalación. Esto significa que nuestros ejemplos comenzarán a funcionar de inmediato. Además, el *frontend* proporciona un maravilloso conjunto organizado de pistas de los objetos disponibles.

Este es un ejemplo de una consulta GraphQL en nuestra base de datos:

```
{
  User_world ( name: France ) {
    name
    population
  }
}
```

Y aquí está la respuesta correspondiente:

```
{
  "data": {
    "User_world": [
      {
        "name": "France",
        "population": 65906000
      }
    ]
  }
}
```

Así es como se ve en el navegador:

Resumen

Tecnología	Antiguedad de la tecnología	Ejemplo de consulta
SQL	50 años Edgar F. Codd	`SELECT population  FROM world  WHERE name = 'France'`
OOP	40 años	`set countryInfo = ##class(User.world).WhereName("Fran`

	Alan Kay and Dan Ingalls	
REST	20 años Roy Thomas Fielding	http://localhost:52773/world/France
GraphQL	5 años Lee Byron	`{ User_world ( name: France ) {  name  population`

- Se está invirtiendo mucha energía en tecnologías como SQL, REST y probablemente también en GraphQL. También hay mucha historia detrás de ellas. Todas funcionan bien entre sí, dentro de la plataforma IRIS, para crear programas que procesan datos
- Aunque no se menciona en este artículo, IRIS también es compatible con otras APIs, basadas en XML (SOAP) y JSON, que están bien implementadas
- A menos que te ocupes específicamente de, por ejemplo, organizar tus objetos, recuerda que los datos intercambiados con la ayuda de APIs siguen representando una versión incompleta y reducida de una transferencia de objetos. Como desarrollador (no el código), eres responsable de garantizar la correcta transferencia de información sobre el tipo de datos de un objeto

Una pregunta para vosotros, estimados lectores

El propósito de este artículo no era solo comparar las API modernas, y ni siquiera revisar las funciones básicas de IRIS. Era ayudar a que se viera lo fácil que es cambiar de una API a otra cuando se accede a una base de datos, aprender a dar los primeros pasos en IRIS y obtener rápidamente resultados de las tareas.

Por esta razón estoy muy interesado en saber cuál es vuestra opinión:

- ¿Esta clase de enfoque ayuda a empezar a trabajar con el software?
- ¿Qué pasos del proceso dificultan el dominio de las herramientas para trabajar con la API en IRIS?
- ¿Podríais nombrar un obstáculo que no habéis podido prever?

Si conocéis a alguien que está aprendiendo a utilizar IRIS, podéis pedirle que deje un comentario en este artículo. Las preguntas y respuestas siempre son útiles para todos.

#API #Modelo de datos #Modelo de datos de objetos #Principiante #InterSystems IRIS