
文章

[Louis Lu](#) · 九月 22, 2021 阅读大约需 13 分钟

IRIS 2021 **技术文档** First Look 5-- **技术概要**：InterSystems SQL

技术概要（First Look）有助于您了解在InterSystems IRIS

®数据平台中如何使用SQL：标准的SQL功能、特有功能以及如何快速上手使用。

要体验技术概要（First Look）的所有内容，您可以在[InterSystems IRIS的免费评估实例](#)上执行相关操作，请参阅[InterSystems First Looks](#)（《InterSystems 技术概要》）。

1. InterSystems SQL: **特性和性能**

InterSystems IRIS 提供高性能、功能完善的 SQL。在 InterSystems IRIS

中使用SQL，包括在单个CPU内核上运行查询，到使用数十个内核的CPU上并行执行查询，已至在InterSystems IRIS服务器集群上运行分布式查询。

在InterSystems IRIS中，可以使用 SQL 的范围包括：

- 联接（Joins）
- 灵活、高性能索引
- 聚合函数和分组
- 以SQL或InterSystems ObjectScript（以下简称“ObjectScript”）编写的存储过程
- JDBC和ODBC连接
- 自动并行查询
- 透明分布式查询

InterSystems SQL提供了强大的工具来实现最佳的SQL查询性能。其中一个工具是使用压缩位图索引：使用紧凑、高度有效的结构和向量化的CPU指令，InterSystems SQL可以只用单个内核即可执行每秒数十亿行的聚合操作以及检查逻辑判断条件。本指南的后续内容将提供位图索引的示例。

想要快速尝试InterSystems IRIS的SQL功能吗？查看[SQL QuickStart](#)！

2. **演示**：SQL Shell

可以通过各种应用程序接口、交互式客户端和标准协议在InterSystems IRIS中执行SQL，包括：

- 用于交互式SQL语句执行的InterSystems IRIS SQL Shell
- ODBC和JDBC客户端、交互式应用（例如，Squirrel SQL或WinSQL）或通过将InterSystems IRIS驱动程序嵌入使用的应用程序
- InterSystems IRIS管理门户中的System Explorer，为执行SQL提供了交互式的Web界面
- 在ObjectScript类中执行嵌入式SQL或动态SQL

如果在阅读本指南后，想了解这些主题的更多有关信息，请参阅下面的“了解有关InterSystems SQL的更多信息”。

此示例向您展示如何使用SQL Shell交互式执行SQL语句，或执行文件中的SQL语句。

2.1 用前须知

要使用该程序，需要一个正在运行的InterSystems IRIS实例。可以选择多种类型的已授权的免费评估实例；该实例不必存储在正在运行的系统（尽管它们可以互相进行网络访问）。如何部署每个类型的实例（如果您还没有要使用的实例）的有关信息，请参阅 InterSystems IRIS Basics: Connecting an IDE（《InterSystems IRIS 基础：连接一个IDE》）中的[Deploying InterSystems IRIS](#)（部署 InterSystems IRIS）。

还需要从GitHub存储库<https://github.com/intersystems/FirstLook-SQLBasics>获取本指南的实用程序文件。应该克隆存储库来下载以下文件：

- stocktabledemoone.sql，其中包含用于创建和加载小型（20行）存量数据的表的SQL语句
- stocktabledemo2.csv，其中包含一百万行存量表数据
- Loader.xml，包含实用程序方法的类文件，用于将数据从stocktabledemo2.csv加载到InterSystems IRIS表中

??? stock_table_demo_two.csv ??? ???????????? [Git Large File Storage](#)?

必须通过该实例访问FirstLook-SQLBasics源。下载文件的程序取决于所使用的[实例类型](#)，如下所示：

- 如果使用的是ICM部署的实例：

1.使用具有-machine 和-interactive 选项的 [icm ssh](#) 命令在储存该实例的节点上打开默认shell，例如：

```
icm ssh -machine MYIRIS-AM-TEST-0004 -interactive
```

2. 在Linux命令行上，使用以下命令之一将存储库克隆到该实例的[数据存储卷\(data storage volume\)](#)。
例如，对于部署在Azure上的配置，数据卷的[默认装入点](#)为 /dev/sdd，因此应使用如下命令：

```
$ git clone https://github.com/intersystems/FirstLook-SQLBasics /dev/sdd/FirstLook-SQLBasics
```

```
????????
```

```
$ wget -qO- https://github.com/intersystems/FirstLook-SQLBasics/archive/master.tar.gz | tar xzv -C /dev/sdd
```

这些文件现在可以通过容器文件系统的/irissys/data/FirstLook-SQLBasics 目录被InterSystems IRIS获取。

- 如果您使用的是通过其他方式部署的容器化实例（授权的版本或社区版本）：

1. 在主机上打开Linux命令行。（如果您在云节点上使用社区版本，请[使用SSH连接到节点](#)，如部署和浏览InterSystems IRIS中所述。）

2. 在Linux命令行上，使用git clone或wget命令，如上所述，将存储库克隆到加载为容器中的数据卷的存储位置。

对于社区版实例，可以克隆到该实例的[%SYS](#)

目录（社区版实例指定的配置数据存储的位置）。在Linux文件系统上，此目录为/opt/ISC/dur。该文件可以在容器文件系统的/ISC/dur/FirstLook-SQLBasics目录下被InterSystems IRIS获取。

对于授权的容器化实例，选择加载为容器中的数据卷的存储位置（包括durable%SYS目录，如果使用的话）。例如，如果您的docker run命令包括选项 -v /home/user1:/external，然后将存储库克隆到/home/user1，这些文件在容器文件系统上的/external/FirstLook-SQLBasics目录中可被Inter- Systems IRIS获取。

- 如果您使用的是InterSystems Learning Labs实例：

1. 在集成的IDE中打开命令行终端。
2. 将目录更改为/home/project/shared，并使用 git clone命令克隆存储库：

```
$ git clone https://github.com/intersystems/FirstLook-SQLBasics
```

该文件夹添加到“Shared”下左侧的Explorer面板，且该目录可在InterSystems IRIS 选择路径的/home/project/shared下被获取。

- 如果您使用的是已安装的实例：

如果实例的主机是安装有GitHub Desktop和GitHub Large File Storage 的Windows系统：

- a. 在主机的web浏览器中转到 <https://github.com/intersystems/FirstLook-SQLBasics>。
- b. 选择Clone or download，然后选择 Open in Desktop。

这些文件可在GitHub目录中被InterSystems IRIS 获取，例如 C:/Users/User1/Documents/GitHub/FirstLook-SQLBasics。

如果主机是Linux系统，只需使用Linux命令行上的git clone命令或wget命令将存储库克隆到您选择的位置即可。

1.

2.2 使用SQL脚本文件创建和填充表

为达到演示目的，我们使用一个SQL脚本文件，stocktabledemoone.sql，来创建并加载包含几行示例数据的表。

创建和加载表格：

1. [打开InterSystems 终端terminal](#)。将看到如下交互式提示：

```
USER>
```

此提示表示您目前在 USER [命名空间\(namespace\)](#) 中，默认情况下该空间为空，可供客户自由使用。从该提示开始，可以执行ObjectScript。

2. 通过输入如下命令立刻打开SQL Shell

```
DO $SYSTEM.SQL.Shell()
```

这将输出以下内容：

```
SQL Command Line Shell
The command prefix is currently set to: <<nothing>>. Enter q to quit, ? for help.
[SQL]USER>>
```

3. 将当前SQL数据库方言设置为IRIS:

```
SET DIALECT=IRIS
```

4. 要运行stocktabledemoone.sql中的语句，请输入命令

RUN <Path>stock_table_demo_one.sql

其中Path是放置文件的位置（请参阅前面的用前须知）。系统会提示您为日志文件指定名称，日志文件包含文件中的语句及其输出结果、如何处理错误以及语句分隔符。接受所有默认值。

语句创建一个表格并插入20行。文件的前几行是：

```
CREATE TABLE FirstLook.StockTableDemoOne (ClientID INTEGER, BrokerID INTEGER, Symbol
VARCHAR(10), TransactionType VARCHAR(4), TransactionDate TIMESTAMP,Quantity INTEGER,
Price DECIMAL(15,2), CommmissionRate DECIMAL(15,2))
GO
INSERT INTO FirstLook.StockTableDemoOne (ClientID, BrokerID, Symbol, TransactionType,
TransactionDate, Quantity, Price, CommmissionRate) VALUES (29834783, 3103, 'RTYU', '
SELL', '2016-01-03', 342, 5.05, 3.25)
GO
```

当脚本运行时，在处理每个SQL语句后将看到输出结果：

```
1. INSERT INTO FirstLook.StockTableDemoOne (ClientID, BrokerID, Symbol,
2.      TransactionType, TransactionDate, Quantity,
3.      Price, CommmissionRate)
4.      VALUES (92609349, 3103, 'HWVT', 'BUY', '2017-10-25', 1500, 451.09, 3.25)
1 Row Affectedu
```

处理完所有语句后，SQL Shell会列出编译的语句数量以及报告的错误和警告，并报告已用时间：

```
Statements
.....compiled: 21
.....with errors reported: 0
...with warnings reported: 0
Elapsed time: .125181 seconds
```

2.3 在SQL Shell中直接运行查询

有了已填充的表后，可以对它运行查询。可以使用单行或多行模式来执行此操作，但后者可能更方便。

1. 若要进入多行模式，请根据提示按Enter键。将看到处于多行模式的确认信息。
2. 输入以下SQL语法，逐行输入。关键字GO 指示shell执行查询并退出多行模式：

```
SELECT BrokerID, TO_CHAR((Quantity * Price),'9,999,999.99') as SubTotal, TransactionD
ate FROM FirstLook.StockTableDemoOne
WHERE TransactionType='SELL' ORDER BY SubTotal DESC
GO
```

您输入的语句将与SQL Shell呼应，后附查询结果。

BrokerID	SubTotal	TransactionDate
5001	302780.00	2017-11-06 09:51:24.735

5002	92350.00	2018-01-15 22:21:17.638
3103	57645.00	2017-09-24 19:36:43.079
3103	45015.00	2016-10-31 19:21:08.913
5001	23180.50	2017-07-31 23:05:49.83
5001	13113.60	2015-11-13 22:13:49.457
5001	12636.00	2015-10-13 05:50:23.209
3103	1727.10	2016-01-03 13:59:01.098
1009	1693.50	2016-01-15 18:18:15.346

在查询结果后，您将看到准备和执行语句所需时间的信息：

```
9 Rows(s) Affected
statement prepare time(s)/globals/cmds/disk: 0.0625s/47683/263292/0ms execute time(s)
/globals/cmds/disk: 0.0006s/64/2903/0ms
cached query class: %sqlcq.USER.cls47
```

准备步骤包括从一条SQL语句的语法到生成可执行代码。此代码被缓存，以供重复使用，因此一条语句通常仅完全准备一次。后续准备只需要使用语句文本的hash来定位已经缓存的代码。

执行步骤包括执行针对查询生成的代码并返回其结果。在每个步骤的列表中都包含以下指标：

- 每个步骤所花费的时间。
- Globals计数，即为准备或执行SQL语句而引用InterSystems IRIS存储的数量。如需进一步了解全局有关信息，请参阅Orientation Guide for Server-Side Programming (《服务器端编程入门指南》) 中的 [“Introduction to Globals \(Globals简介\)”](#) 一章。
- 为准备或执行SQL语句而执行的ObjectScript命令的数目。

显示结果的结尾是缓存的查询类 (cached query class)，这是首次准备语句时生成的缓存代码的ObjectScript类。

3. 也可以使用聚合函数和 GROUP BY。请注意，可以对聚合函数的别名排序：

```
SELECT BrokerID, TO_CHAR(SUM(Quantity * Price), '9,999,999.99') as SubTotal FROM FirstLook.StockTableDemoOne
GROUP BY BrokerID ORDER BY SubTotal DESC
GO
```

BrokerID	SubTotal
3103	868,993.60
1009	808,453.50
5001	593,242.82
5002	187,560.00

```
4 Rows(s) Affected
statement prepare time(s)/globals/cmds/disk: 0.1665s/45832/237712/77ms
execute time(s)/globals/cmds/disk: 0.0025s/122/2434/2ms
cached query class: %sqlcq.USER.cls9
```

3. 演示：使用位图索引提高查询性能

如果您正在使用大型数据集，寻找改善查询性能的方法，位图索引是您可以使用的几种方法之一。

如果一个表有一个或多个字段，字段的可能值集合较小，则创建位图索引特别适合。

有关位图索引运行的详细信息，请参阅InterSystems SQL Optimization Guide (《InterSystems SQL优化指南》) 的 [Bitmap Indices \(位图索引\)](#) 章节。

在本演示中，您将看到在股票交易数据的百万行数据表格中创建目标位图索引引发的变化。您将使用几个简单的ObjectScript 命令；很容易地从SQL Shell内无缝访问 ObjectScript 类库。

要运行演示：

1. 按照 [使用SQL脚本文件创建和填充表格](#) ([Creating and Populating a Table With a SQL Script File](#)) 所述，在终端中启动SQL Shell。
2. 创建表：

```
CREATE TABLE FirstLook.StockTableDemoTwo (ClientID INTEGER, BrokerID INTEGER,  
      Symbol VARCHAR(10), TransactionType VARCHAR(4),  
      TransactionDate TIMESTAMP, Quantity INTEGER,  
      Price DECIMAL(15,2), CommissionRate DECIMAL(15,2))
```

```
0 Rows Affected  
statement prepare time(s)/globals/cmds/disk: 0.0063s/1811/22260/0ms  
      execute time(s)/globals/cmds/disk: 0.2138s/76495/655985/76ms  
      cached query class: %sqlcq.USER.cls1
```

3. 导入Loader类 (Loader.xml文件)。OBJ前缀表示 SQL Shell 按照 ObjectScript 处理以下命令；“c”标志指示InterSystems IRIS编译代码，“k”标志确保源代码存储在活动命名空间 (active namespace) 中。

```
OBJ DO $ system.OBJ.Load ("< Path > Loader.xml", "ck")
```

其中Path是放置文件的位置 (请参阅[用前须知](#))。将输出如下结果：

```
Load started on 04/19/2018 15:17:53  
Loading file C:\Users\user\repos\FirstLook-  
SQLBasics\Loader.xml as xml Imported class: FirstLook.Loader  
Compiling class FirstLook.Loader Compiling routine FirstLook.Loader.1 Load finished s  
uccessfully.
```

4. 要将stocktabledemoTwo.csv中的数据加载到表中，请在终端中运行以下命令：

```
OBJ WRITE ##class(FirstLook.Loader).LoadStockTableCSV("<Path>stock_table_demo_two.csv  
")
```

其中Path是放置文件的位置。此命令的输出结果1000000仅表示已加载100万行。

5. 运行以下查询:

```
SELECT DISTINCT BrokerID FROM FirstLook.StockTableDemoTwo
```

输出结果显示，可能的经纪人ID的数量非常小，因此该字段是位图索引的好选择。

BrokerID
115
107
101
114
119
104
109
108

```
20 Rows(s) Affected
statement prepare time(s)/globals/cmds/disk: 0.0645s/43430/197693/9ms
execute time(s)/globals/cmds/disk: 1.2569s/2000039/9001314/0ms
cached query class: %sqlcq.USER.cls10
```

6. 要在添加位图索引之前查看涉及BrokerID字段的COUNT查询的性能，请运行以下查询：

```
SELECT BrokerID, COUNT(*) As Transactions FROM FirstLook.StockTableDemoTwo GROUP BY BrokerId ORDER BY Transactions DESC
```

BrokerId	Transactions
103	50386
118	50304
107	50247
112	50207
101	50174
109	50088
115	50088
104	50048
111	50031
105	50008

113	49996
119	49942
114	49919
116	49894
110	49888
108	49882
102	49843
120	49768
106	49742
117	49545

20 Rows(s) Affected

观察查询返回结果后显示的查询性能统计信息：所用总时间（包括准备和执行时间）约为0.65秒。

```
statement prepare time(s)/globals/cmds/disk: 0.0695s/45048/225490/13ms
      execute time(s)/globals/cmds/disk: 0.5878s/1000250/11002218/0ms
      cached query class: %sqlcq.USER.cls7
```

7. 在BrokerID上添加位图索引:

```
CREATE BITMAP INDEX BrokerIDIdx ON TABLE FirstLook.StockTableDemoTwo (BrokerID)
```

0 Rows Affected

```
statement prepare time(s)/globals/cmds/disk: 0.0056s/1723/15958/0ms
      execute time(s)/globals/cmds/disk: 0.9805s/2071557/18505697/1ms
      cached query class: %sqlcq.USER.cls11
```

8.

运行与上述相同的SELECT查询。请注意性能提升：在下面的示例中，查询总共花费了大约0.35秒，减少了近50%。

```
SELECT BrokerID, COUNT(*) As Transactions FROM FirstLook.StockTableDemoTwo GROUP BY B
rokerId ORDER BY Transactions DESC
```

...

```
statement prepare time(s)/globals/cmds/disk: 0.0573s/45585/231374/0m
      execute time(s)/globals/cmds/disk: 0.2926s/622/15004397/0ms
      cached query class: %sqlcq.USER.cls1
```

4. 了解有关InterSystems SQL的更多信息

要了解有关SQL和InterSystems IRIS的更多信息，请参阅：

4.1 介绍材料

[Using InterSystems SQL \(使用InterSystems SQL\)](#)

[InterSystems SQL Reference \(InterSystems SQL参考书目\)](#)

[InterSystems SQL Overview \(InterSystems SQL概述\)](#)

4.2 SQL开发

[SQL –Things You Should Know \(SQL-您应该知道的事情\)](#)

[Developing with InterSystems Objects and SQL \(使用InterSystems Objects和SQL开发\)](#)

4.3 查询优化

[First Look: Optimizing SQL Performance with InterSystems IRIS \(技术概要：使用InterSystems IRIS优化SQL性能\)](#)

[InterSystems SQL Optimization Guide \(InterSystems SQL优化指南\)](#)

[Academy – Optimizing SQL Performance \(学院派–优化SQL性能\)](#)

[Optimizing SQL Queries \(优化SQL查询\)](#)

4.4 分片和可扩展性

[First Look: Scaling for Data Volume with Sharding \(技术概要：带分片的数据卷扩展\)](#)

[Scalability Guide \(可扩展性指南\)](#)

4.5 SQL Search

[SQL Search First Look: SQL Search with InterSystems IRIS \(技术概要：使用InterSystems IRIS进行\)](#)

[Using InterSystems SQL Search \(使用InterSystems SQL Search\)](#)

[Creating iFind Indices for Searching Text Fields \(创建用于搜索文本字段的iFind索引\)](#)

4.6 JDBC

[First Look: JDBC and InterSystems IRIS \(技术概要：JDBC和InterSystems IRIS\)](#)

[Using Java JDBC with InterSystems IRIS \(documentation\) \(使用InterSystems IRIS进行Java JDBC <文档>\)](#)

[Java Overview \(Java概述\)](#)

[Using JDBC with InterSystems IRIS \(online learning\) \(在InterSystems IRIS中使用JDBC <在线学习>\)](#)

[#InterSystems IRIS #InterSystems IRIS for Health](#)

URL:

<https://cn.community.intersystems.com/post/iris-2021-%E6%8A%80%E6%9C%AF%E6%96%87%E6%A1%A3-first-look-5-%E6%8A%80%E6%9C%AF%E6%A6%82%E8%A6%81%E6%BC%9Aintersystems-sql>