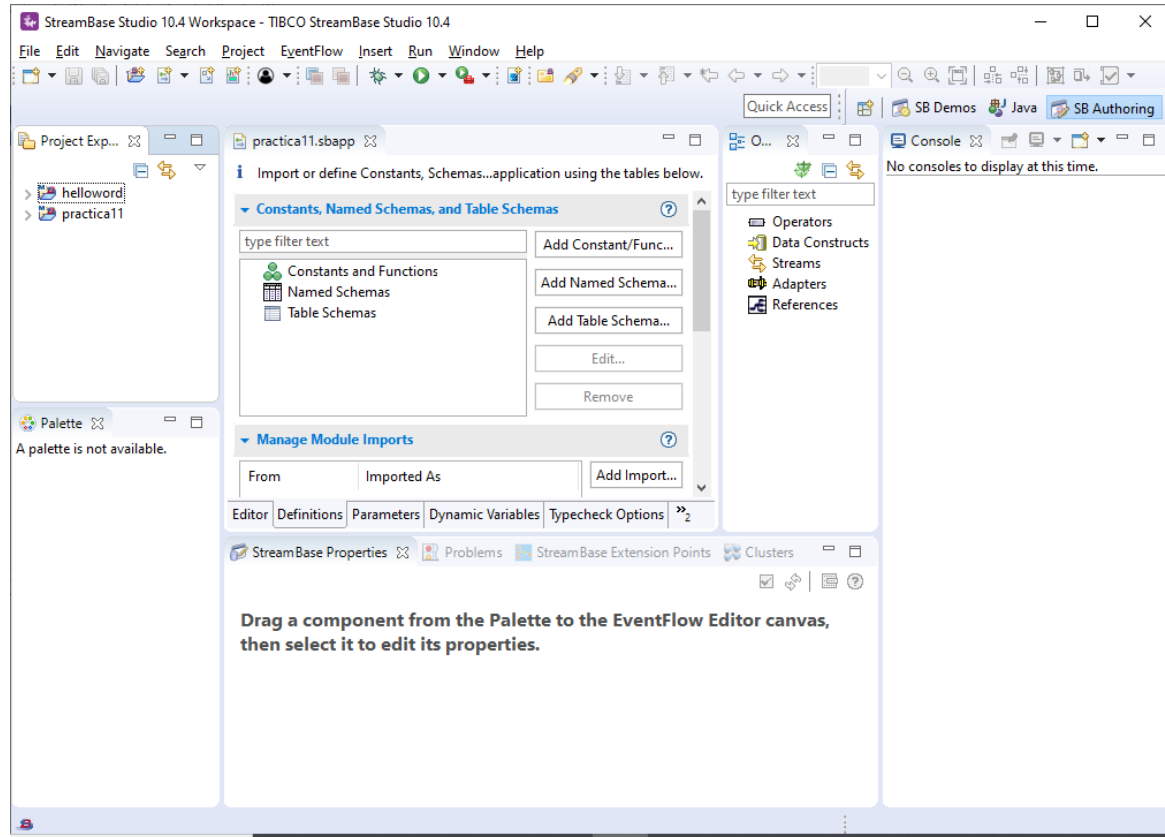


Práctica Lección 11. Gestión de Liquidez y Órdenes para FX

Creacion de proyecto.

Se crea un proyecto con e nombre paractica 11 siuguiendo la ruta indicada(File-> New->Streambase Project), luego de algunas modificaciones queda la siguiente pantalla:



Definición de esquemas.

Se añade un nuevo esquema, se sigue la siguiente instrucción : Named Schemas -> Add Named Schema y se añade el esquema correspondiente al tick o quote (el precio del proveedor) y de una orden. Los esquemas se pueden ver a continuación:

Edit Named Schema

Edit Named Schema

Enter a name for the Schema and use the table to specify its fields.

Name: Quote

Fields: No parent schemas

Field Name	Type	Description
Symbol	string	The code of the execution
BidVol	int	The bid volume
BidPrice	double	The bid price
AskVol	int	The ask volume
AskPrice	double	The ask price

Schema Description

?

OK

Cancel

Edit Named Schema

Edit Named Schema

Enter a name for the Schema and use the table to specify its fields.

Name: Order

Fields: No parent schemas

Field Name	Type	Description
Symbol	string	The code of the execution
Vol	int	The volume to be
Price	double	The price of the order
Side	string	Determines if buys

Schema Description

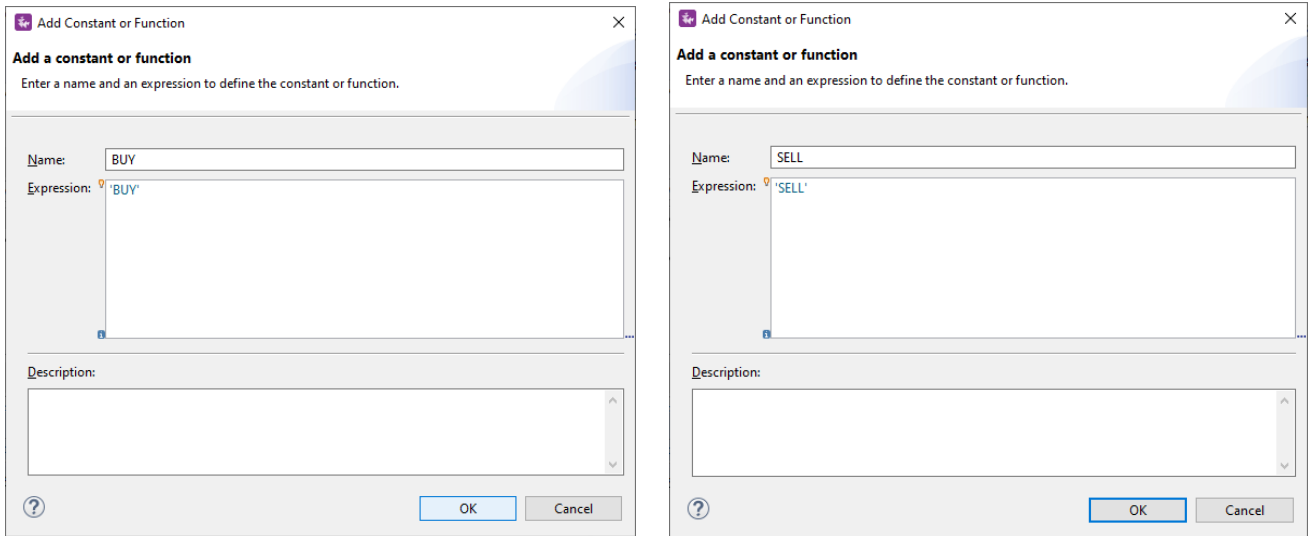
?

OK

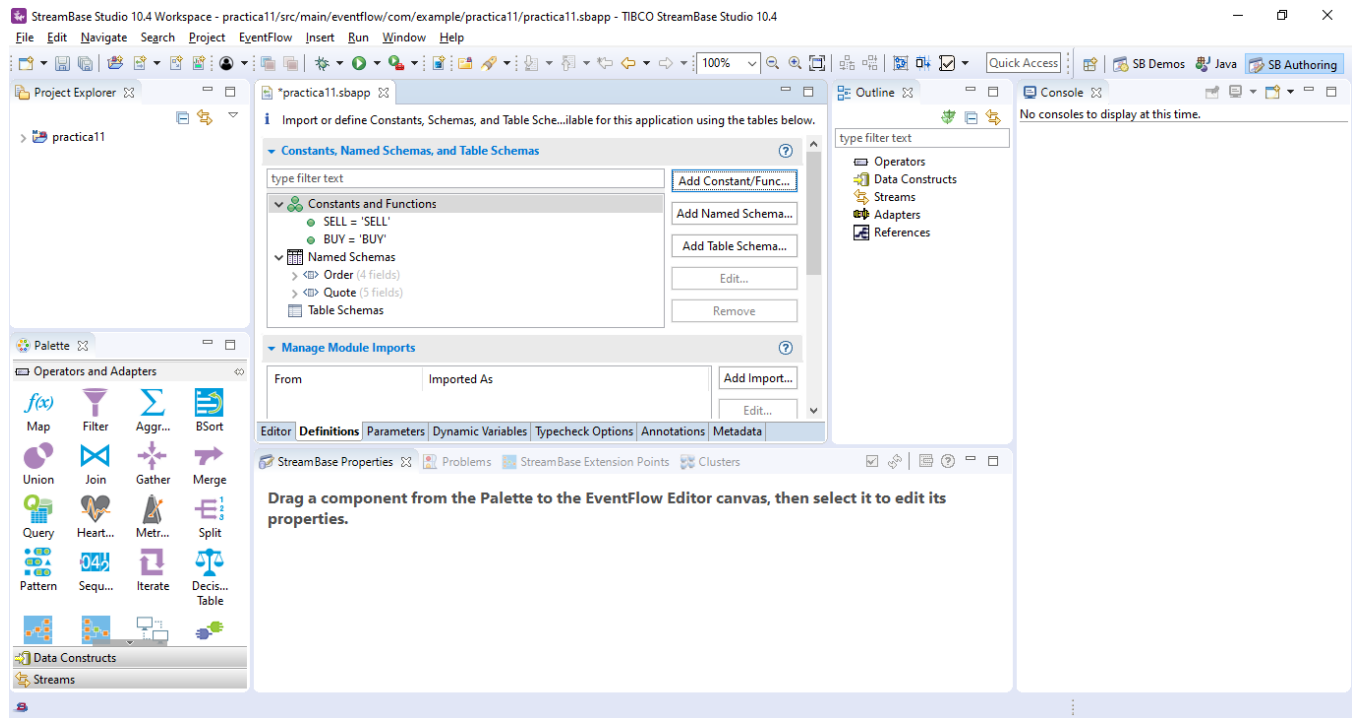
Cancel

Definición de constantes

A continuación se definen dos constantes 'BUY' Y 'SELL' que van a indicar si la orden es de compra o venta para ello se sigue la siguiente ruta: Definitions -> Constants and Functions -> Add Constant /Func. Las pantallas donde crean son las siguiente:

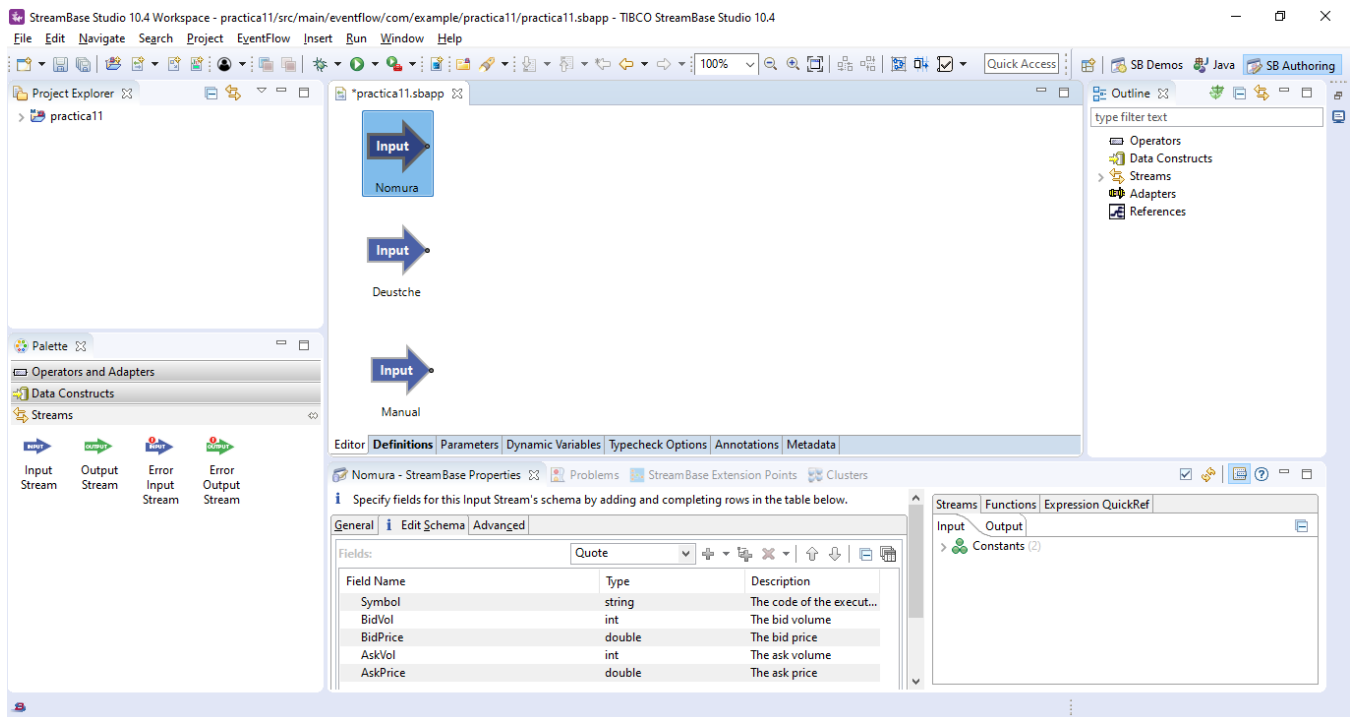


Quedando finalmente la pestaña "Definitions" del siguiente modo:



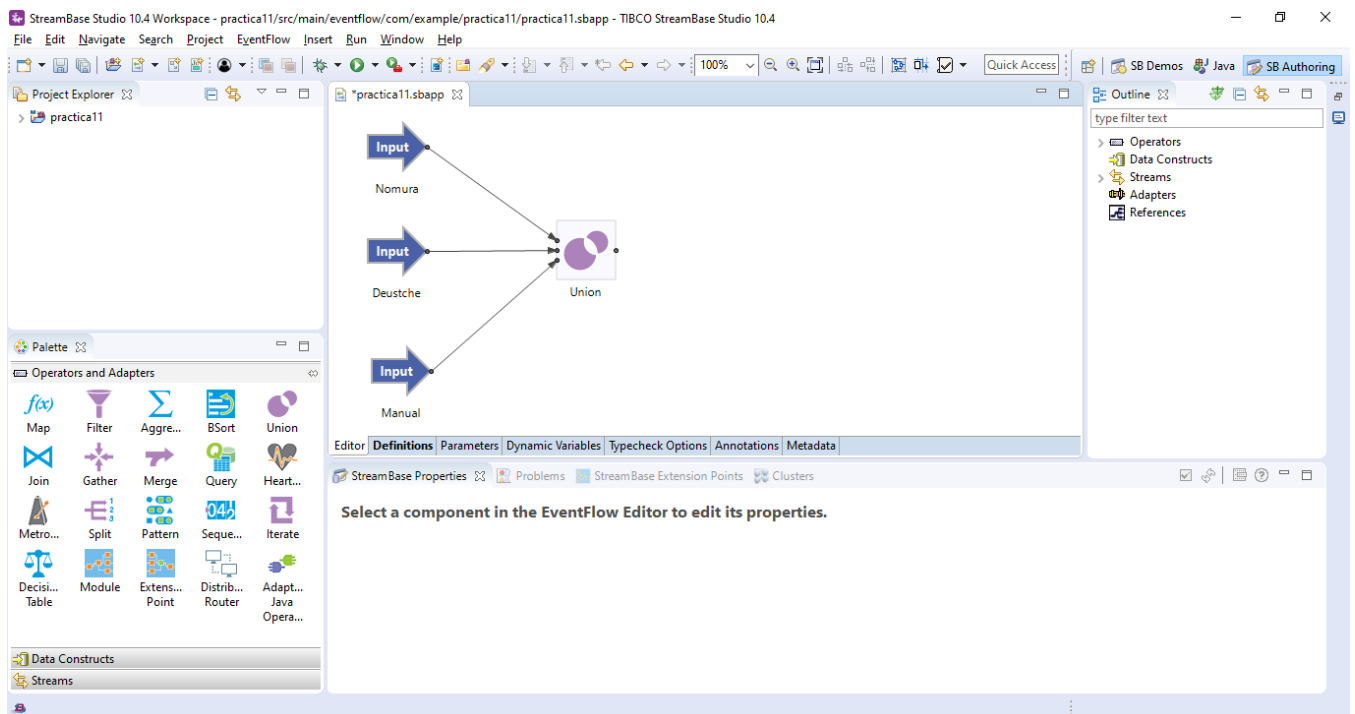
Creación de flujos de entrada de liquidez

Ahora se va a añadir los flujos de entrada de la liquidez proveniente de tres fuentes, una llamada Nomura, otra llamada Deutsche y otra que será llamada MANUAL y la utilizaremos para inyectar quotes de modo controlado para probar el sistema, el resultado es el siguiente:

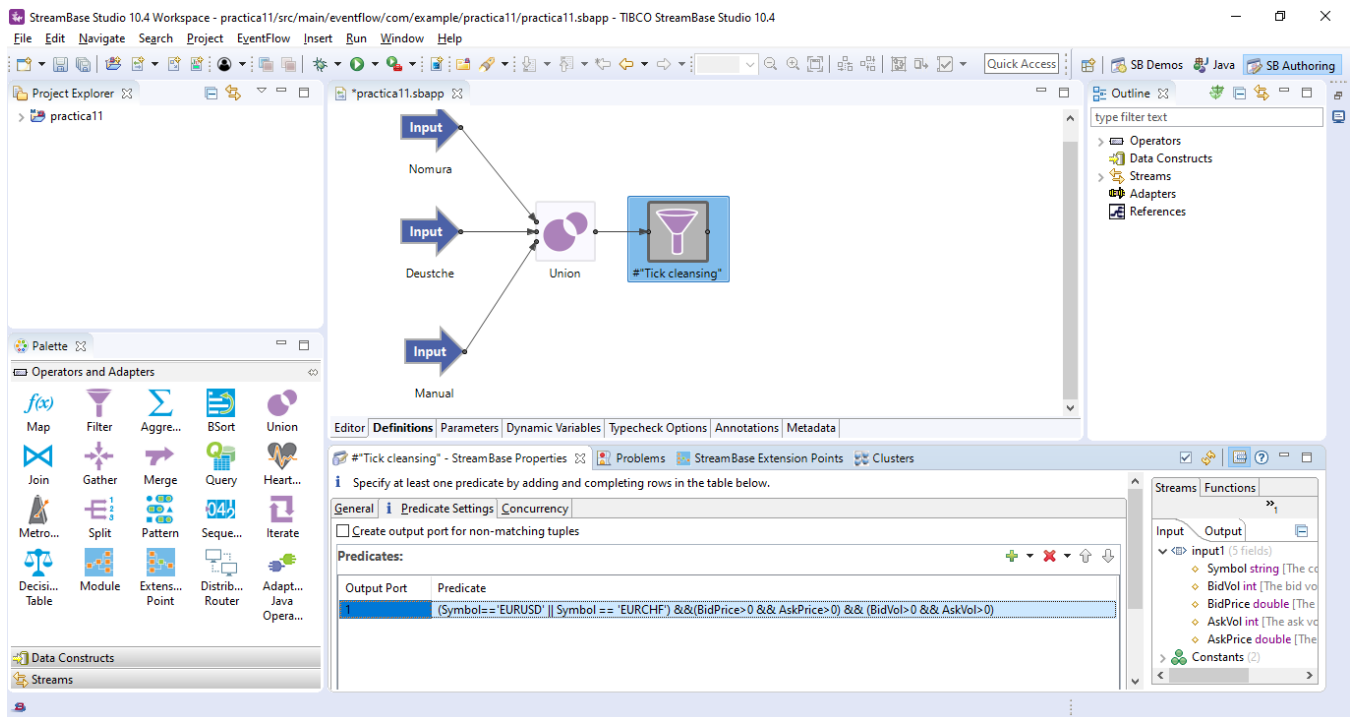


Cleansing de quotes

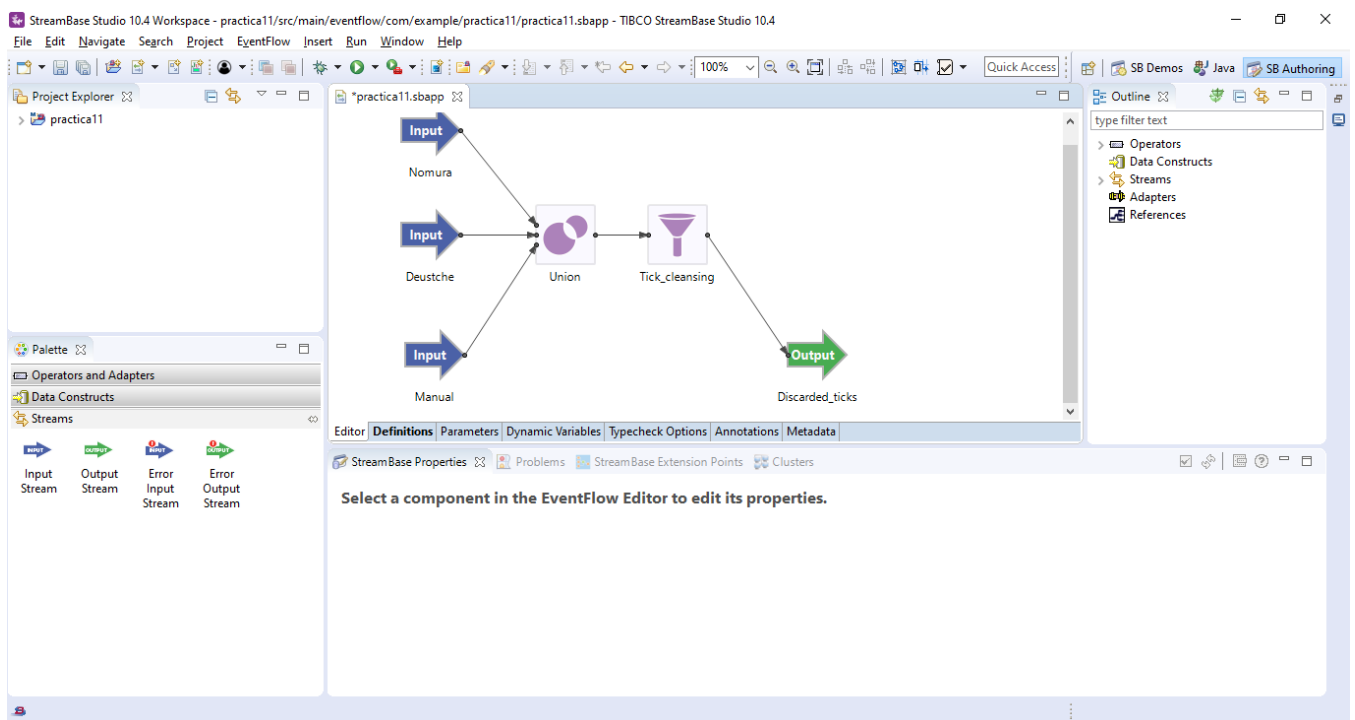
Ahora vamos a pasar por el proceso que filtra los quotes, descartando aquellos que contengan algún dato que a nuestro sistema no le interese, pero para ello, es necesario que los tres flujos de entrada se unan en uno solo, añadiendo un operador “Union” al que le diremos que tiene 3 puertos de entrada (uno por cada flujo):



Y ya estamos listos para añadir nuestro filtro de quotes y realizar nuestro “Cleansing”, añadiendo el operador “Filter”: Establecemos el criterio de filtro:

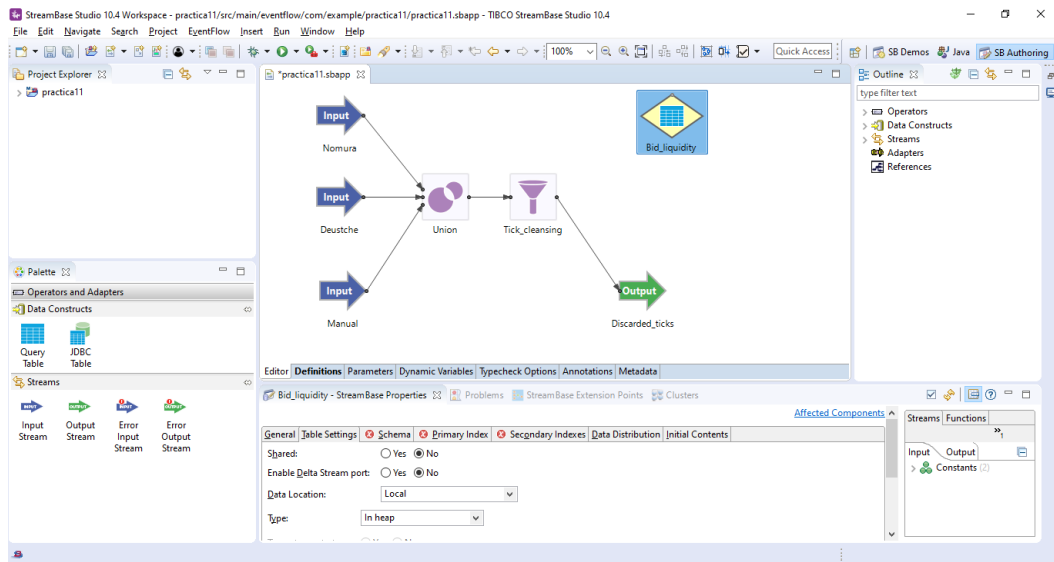


Y añadimos un flujo para poder ver los ticks descartados:

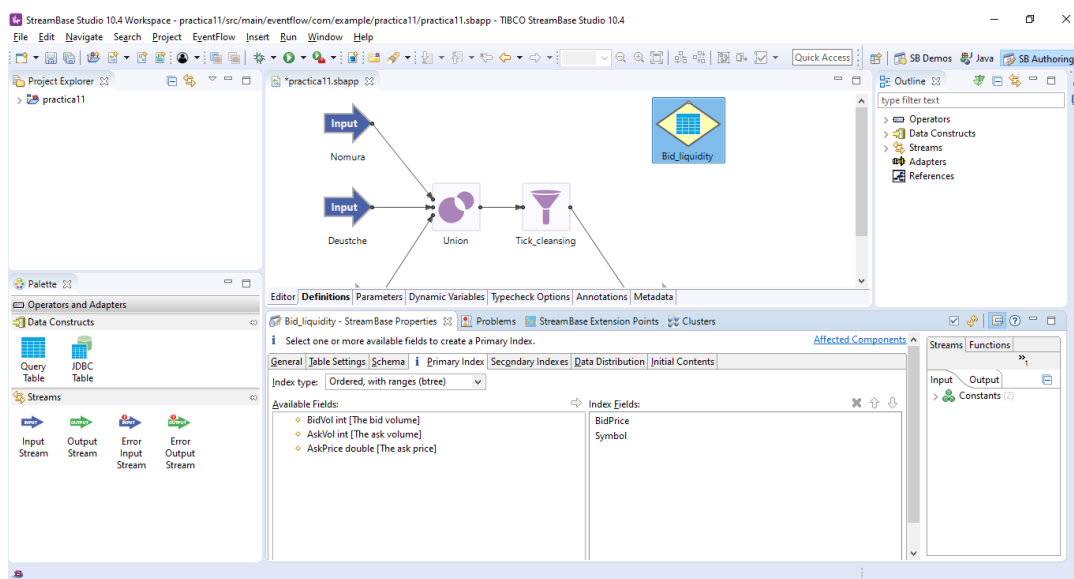
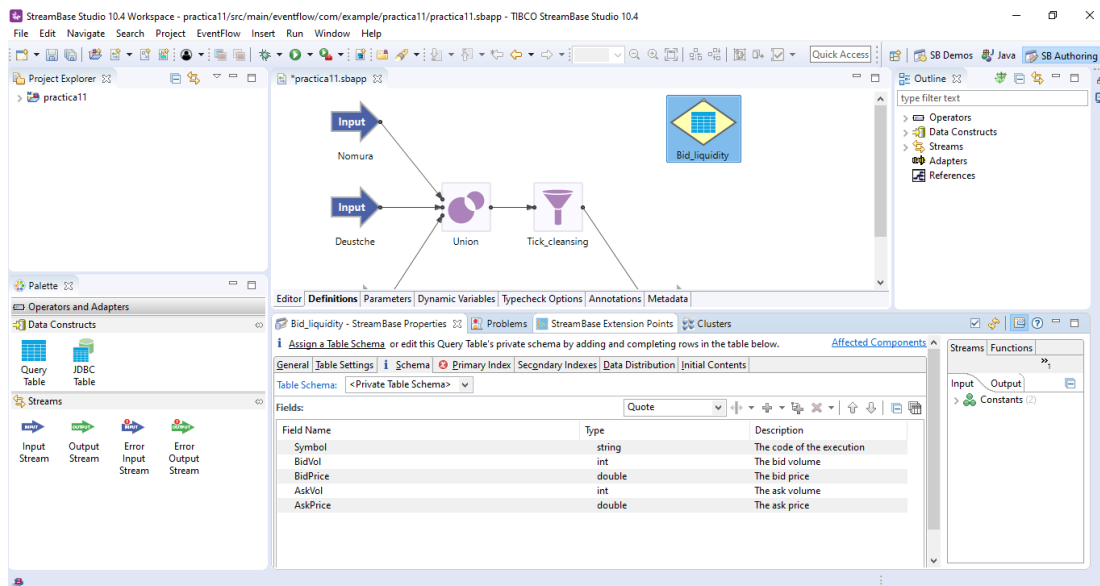


Creación del Liquidity Book

Añadimos las tablas en memoria que van a soportar la liquidez. Por sencillez vamos a separar la liquidez en una tabla para Bid y otra para Ask.

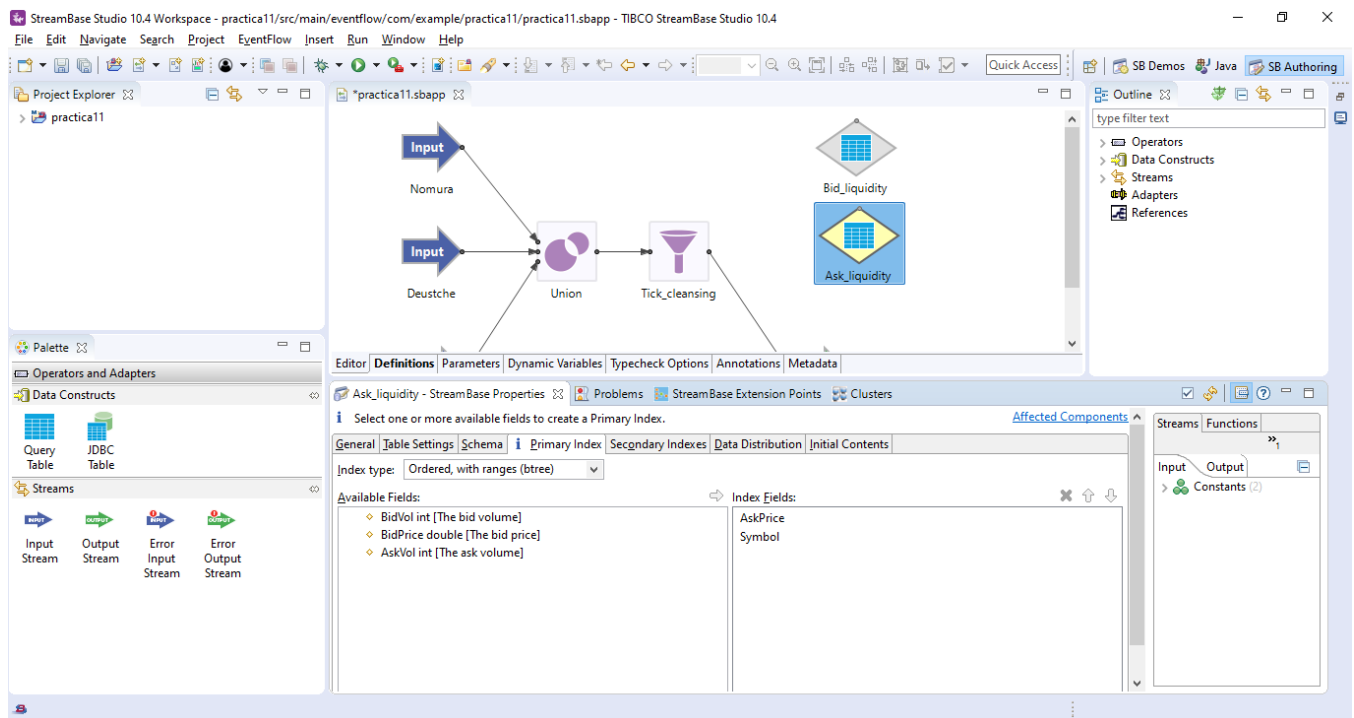


Establecemos el esquema y el índice primario de la tabla:

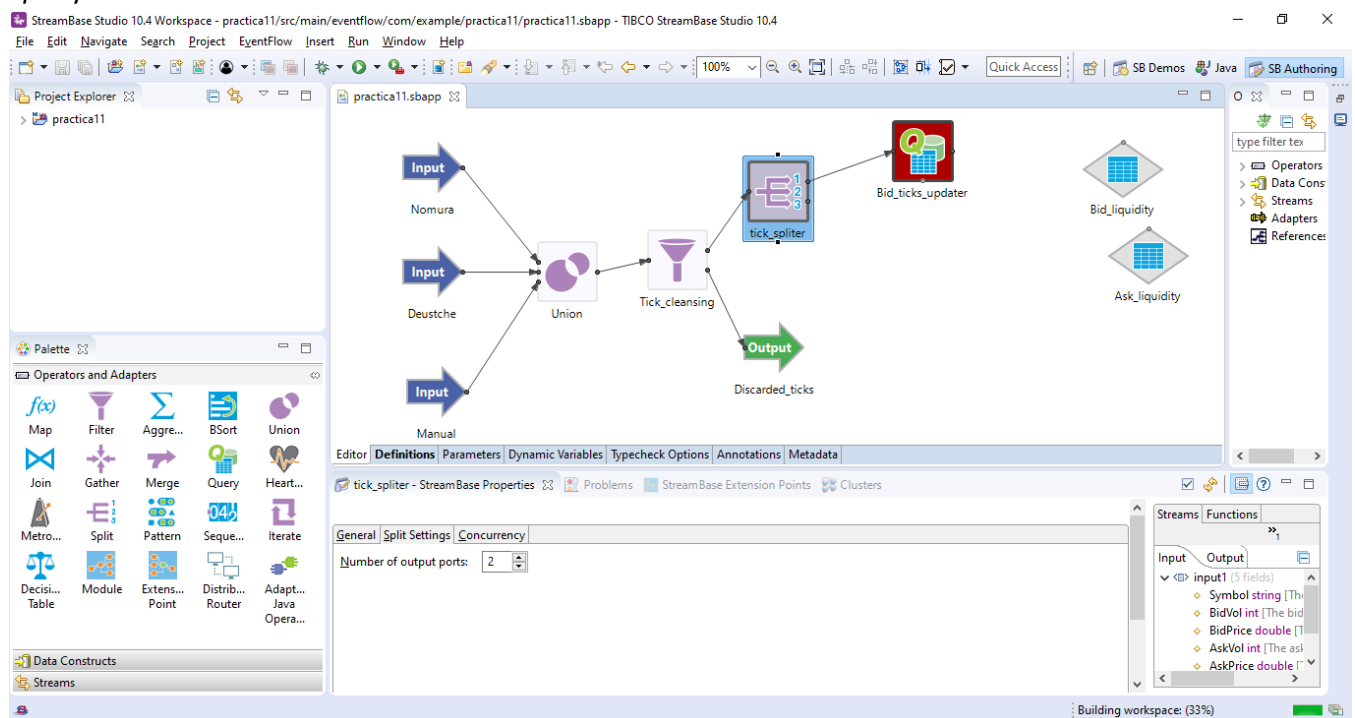


Almacenando quotes en el Liquidity Book

Ahora vamos a meter la actualización de las tablas de liquidez según llegue un tick, pero primero debemos dividir el flujo de ticks para que se pueda actualizar por separado el bid y el ask:

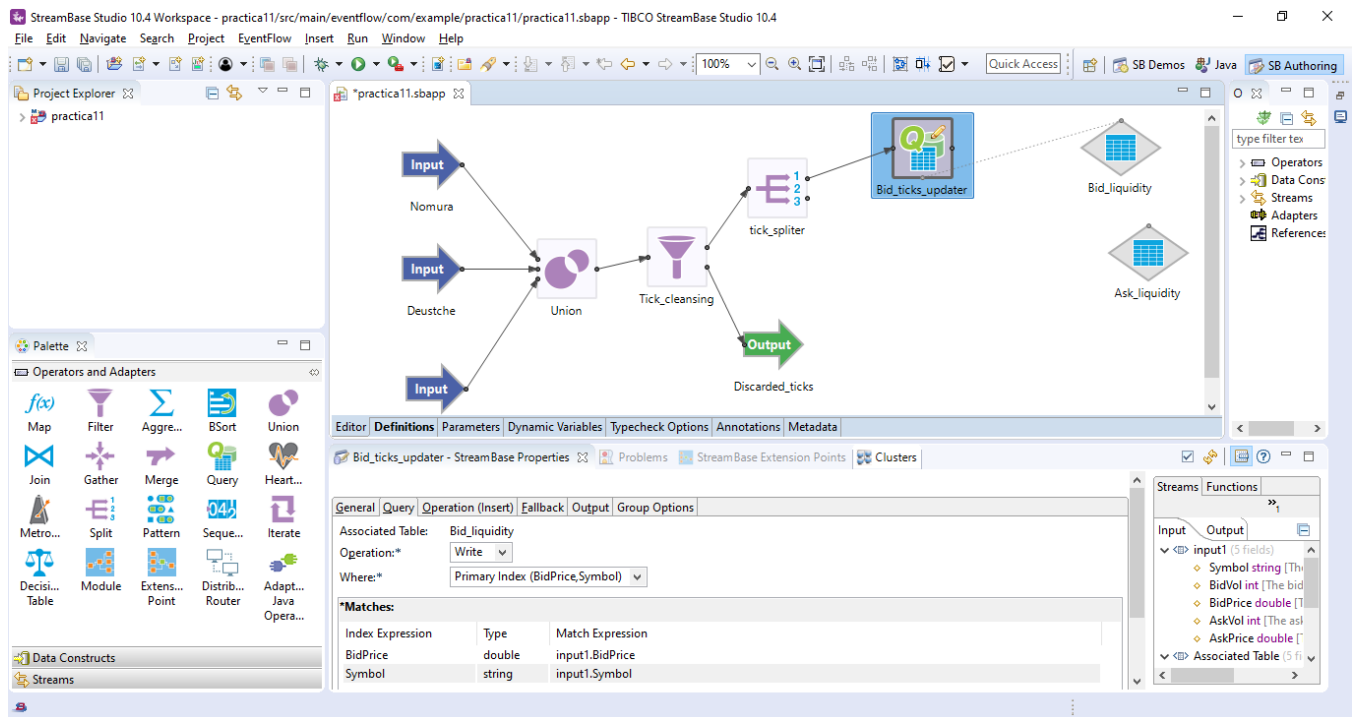


Para actualizar la tabla debemos utilizar el operador “Query”, lo añadimos y lo conectamos con el *Split* y con la tabla



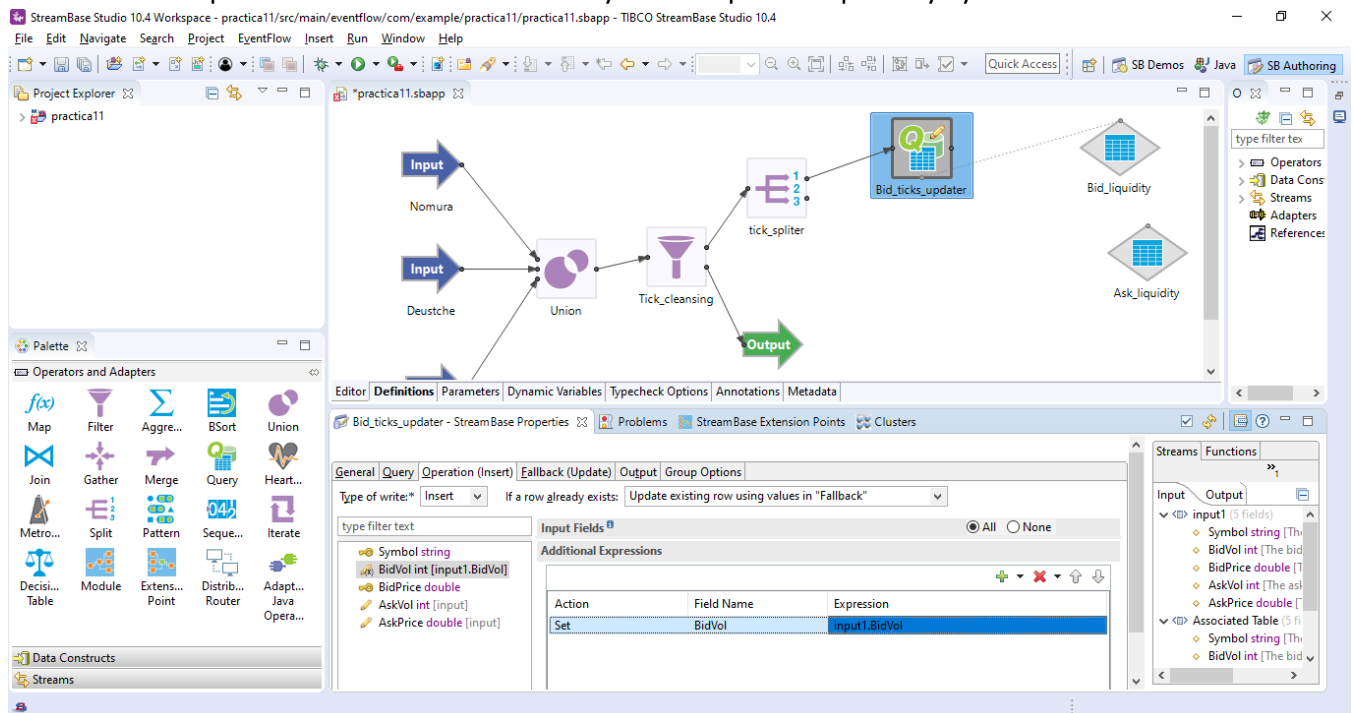
Ahora falta incorporar la información para la Query actualice la tabla, por eso muestra el error saliendo en rojo.

Primero se indica que operación realiza, lectura, escritura o borrado, seleccionaremos escritura e indicaremos los campos por lo que actualizará la tabla (será la clave primaria, Symbol y BidPrice):

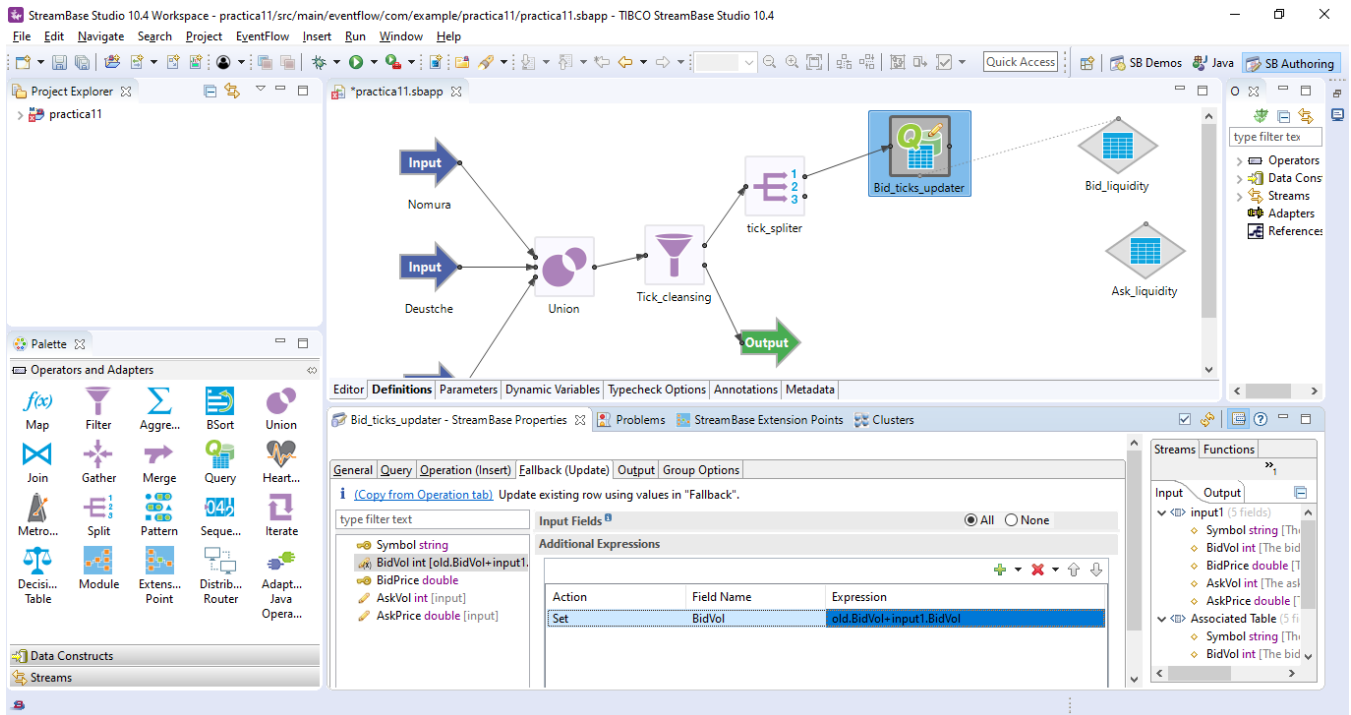


Ahora debemos indicar cómo se realiza la escritura en la pestaña "Operation", que debe:

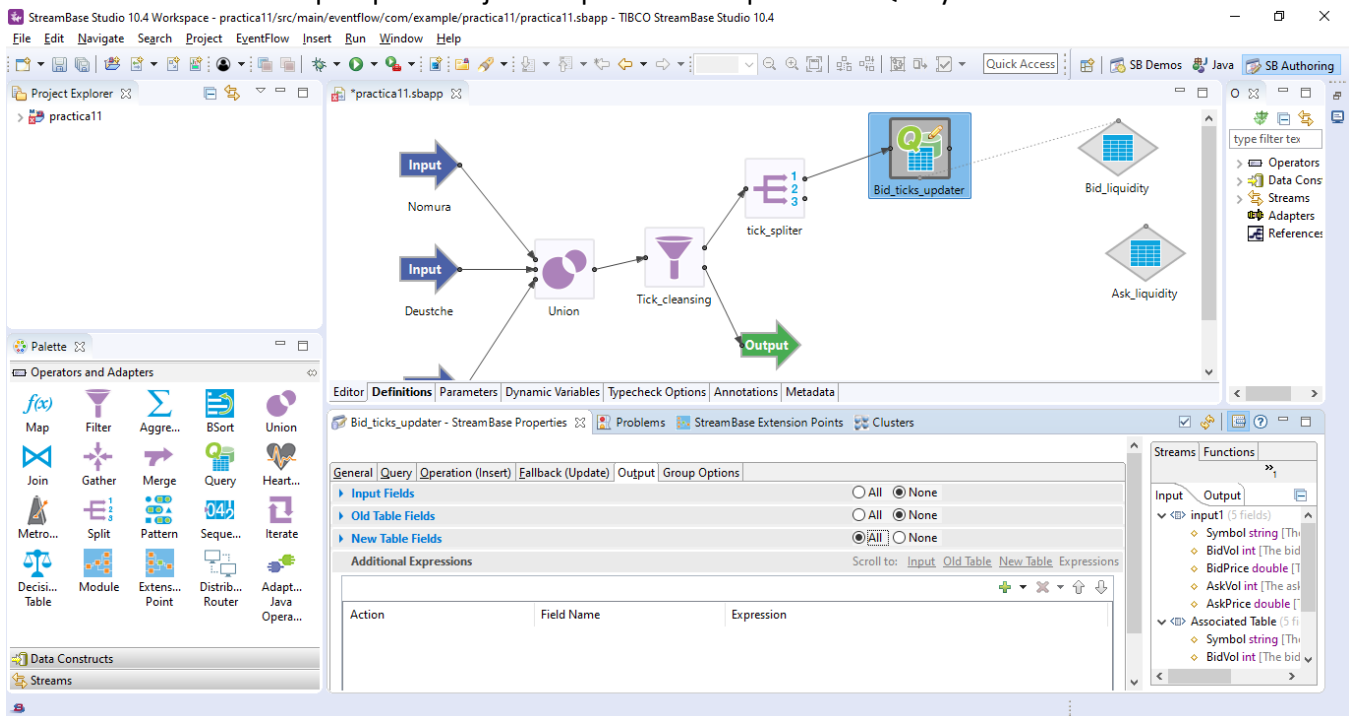
- Insertar una tupla nueva con el BidVol si no hay entrada para ese precio y Symbol



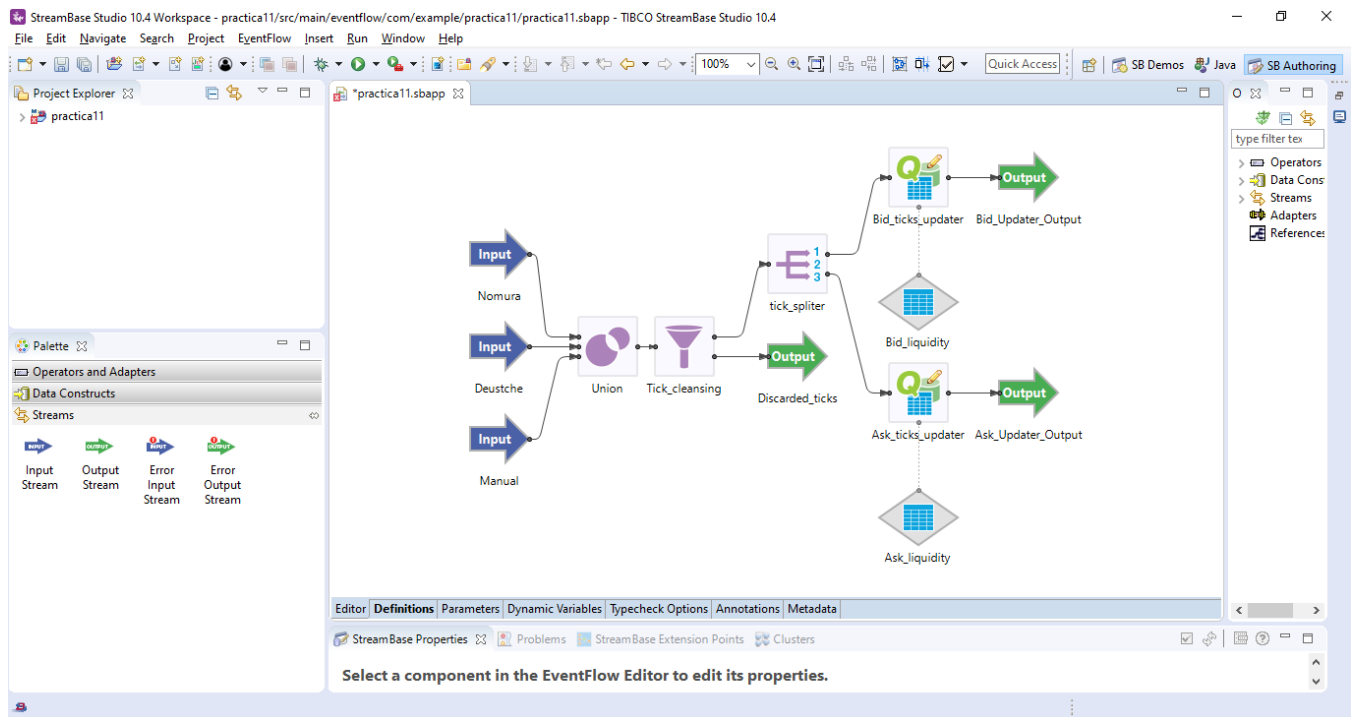
O Actualizar el BidVol si existe una entrada para ese Symbol y BidPrice, utilizando la información de actualización en la pestaña "Fallback", de modo que sume el volumen del tick que esté tratando al volumen ya existente



Y ahora, para poder saber que se ha actualizado en la tabla, añadimos un flujo de salida a la Query e indicamos como salen tuplas por el flujo en la pestaña "Output" de la Query":

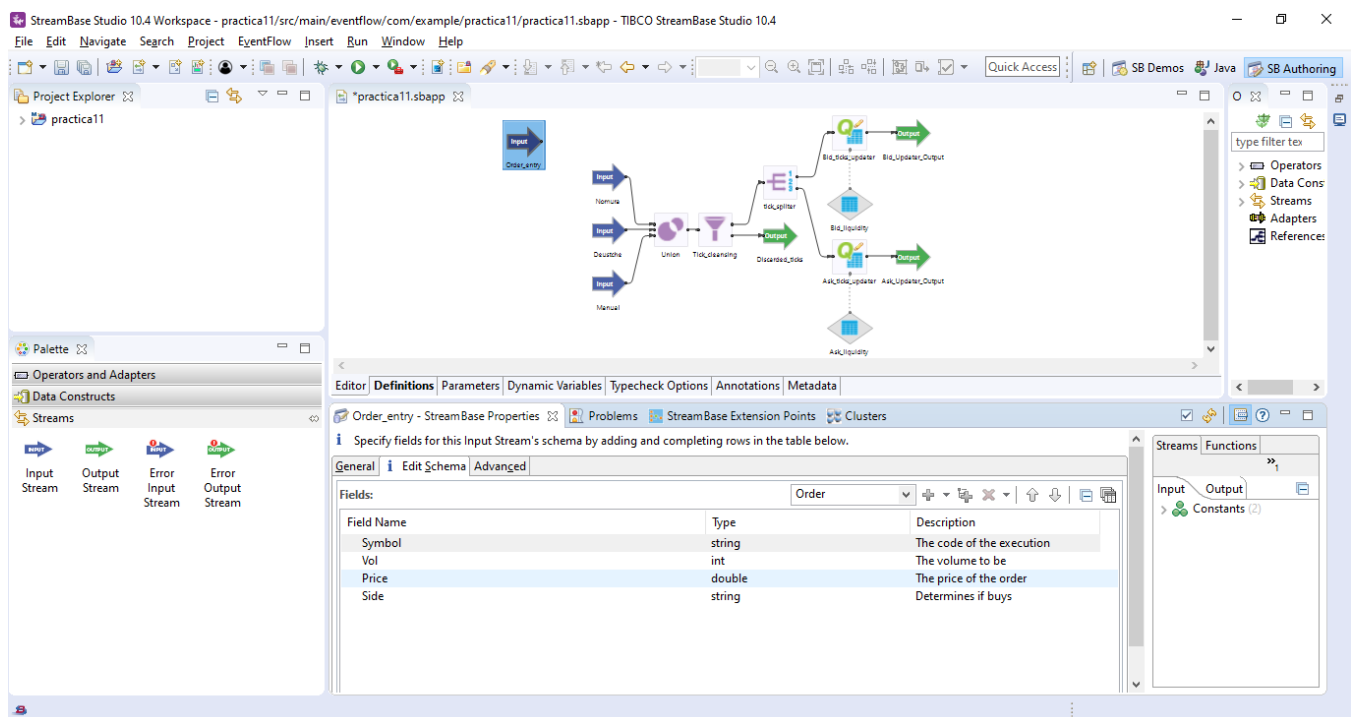


Y repetimos los mismos pasos para el lado Ask y reorganizamos los elementos para mayor facilidad (Control + L):



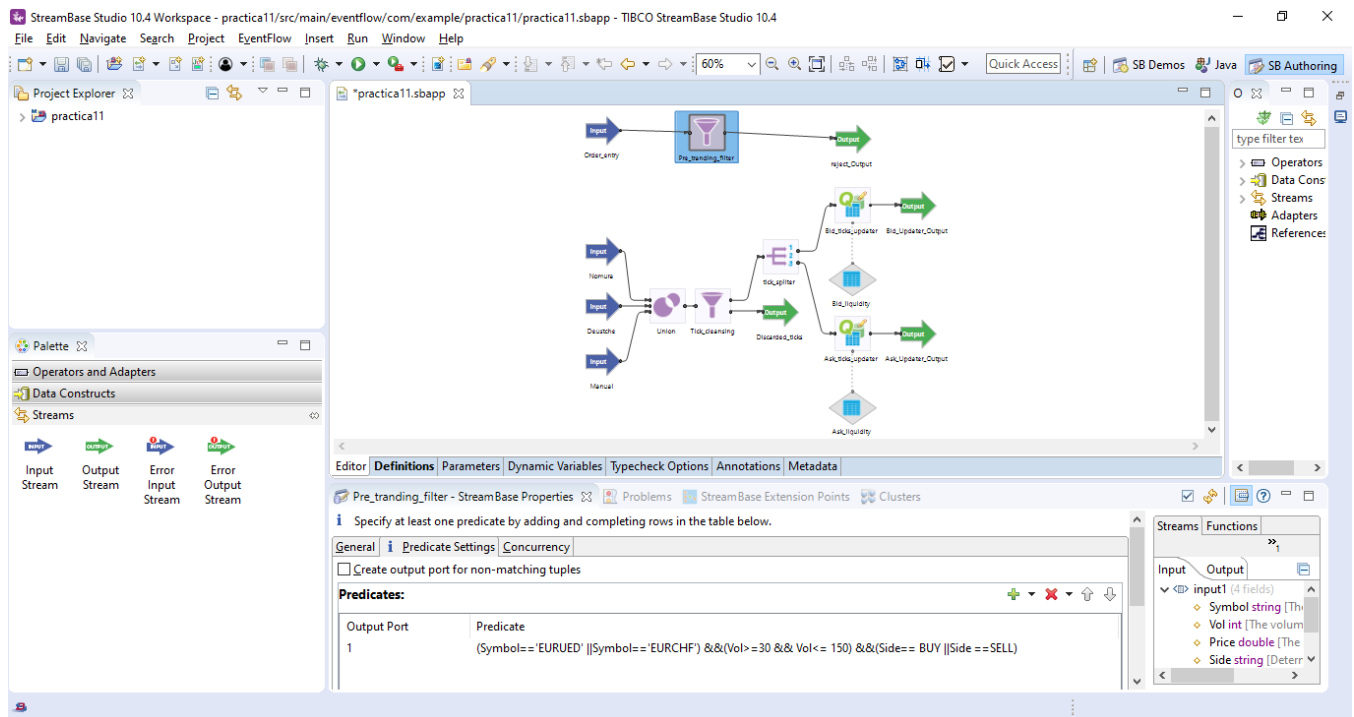
Creación de flujos de entrada de órdenes

Añadimos el flujo y establecemos el esquema como el de Order, ya definido en el primer paso:



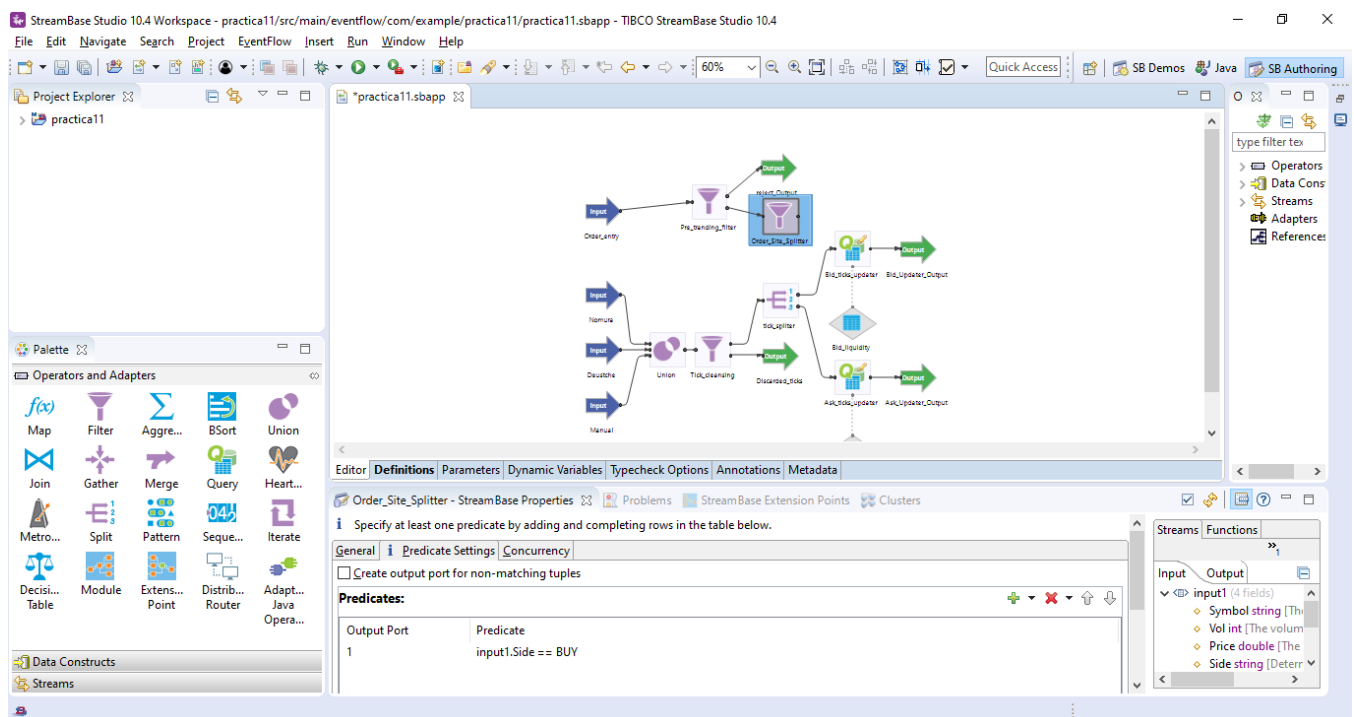
Aplicación de filtros pre-trading

Añadimos el filtro y conectamos el flujo de entrada al filtro. Establecemos el criterio de filtro, que será, filtrar por el Symbol, solo admitiremos EURUSD y EURCHF, el resto se rechazará, también, por “jugar”, aplicaremos un filtro de volumen, rechazando aquel volumen de las órdenes que sean inferiores a 30 unidades y superiores a 150 unidades. Por supuesto, solo admitiremos compras o ventas.



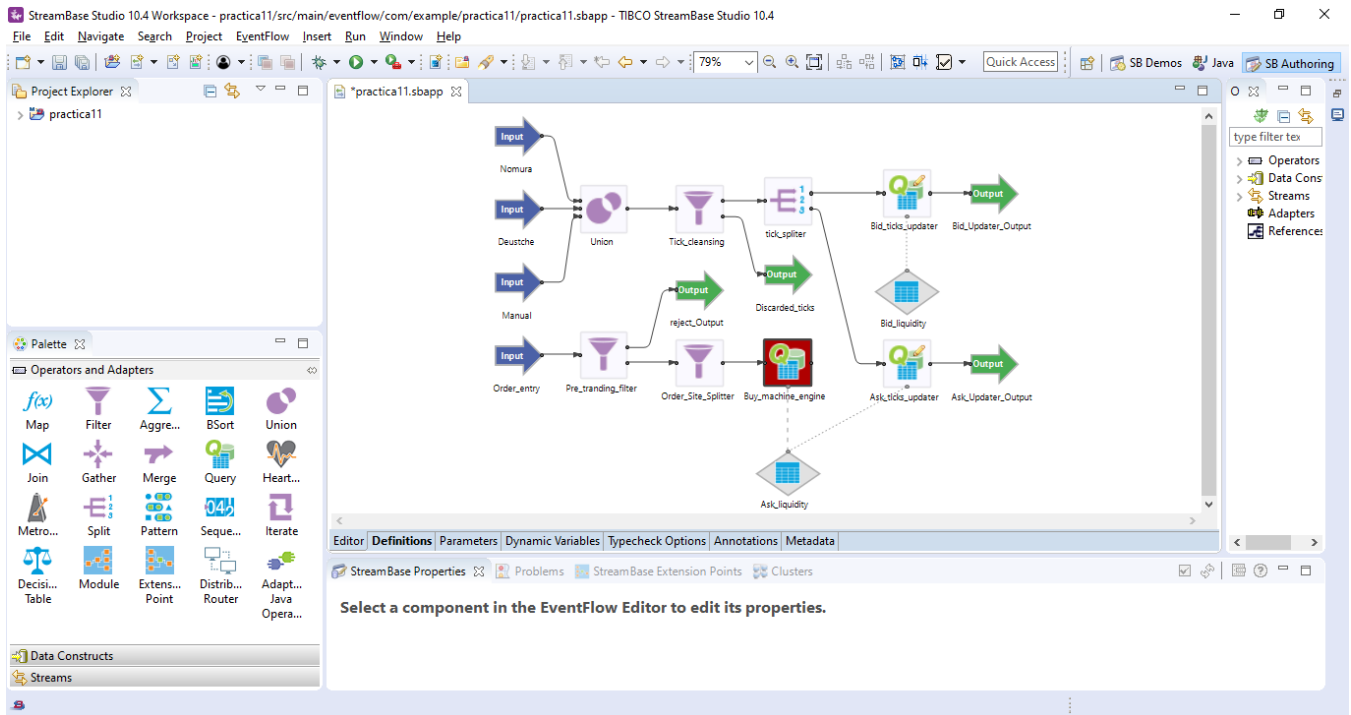
Matching de órdenes contra el Liquidity Book

Para añadir los motores de case de las órdenes contra la liquidez, vamos a dividir el flujo según el sentido de la orden, por lo que debemos añadir un filtro que divida las órdenes de compra y las de venta en flujos diferentes:

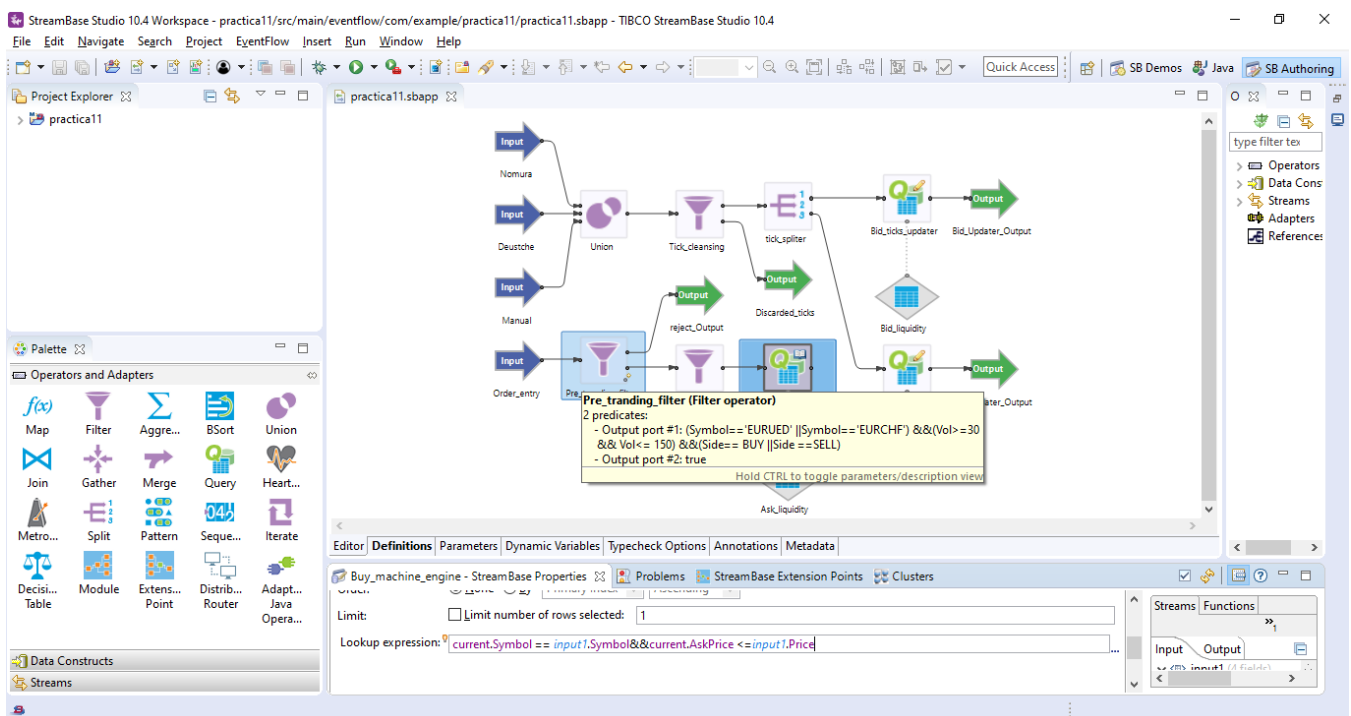


Ahora vamos a procesar las órdenes de compra, que tendrán que buscar liquidez en la tabla donde se almacenen los quotes tipo ASK, y las órdenes de venta tendrán que casar con la liquidez BID.

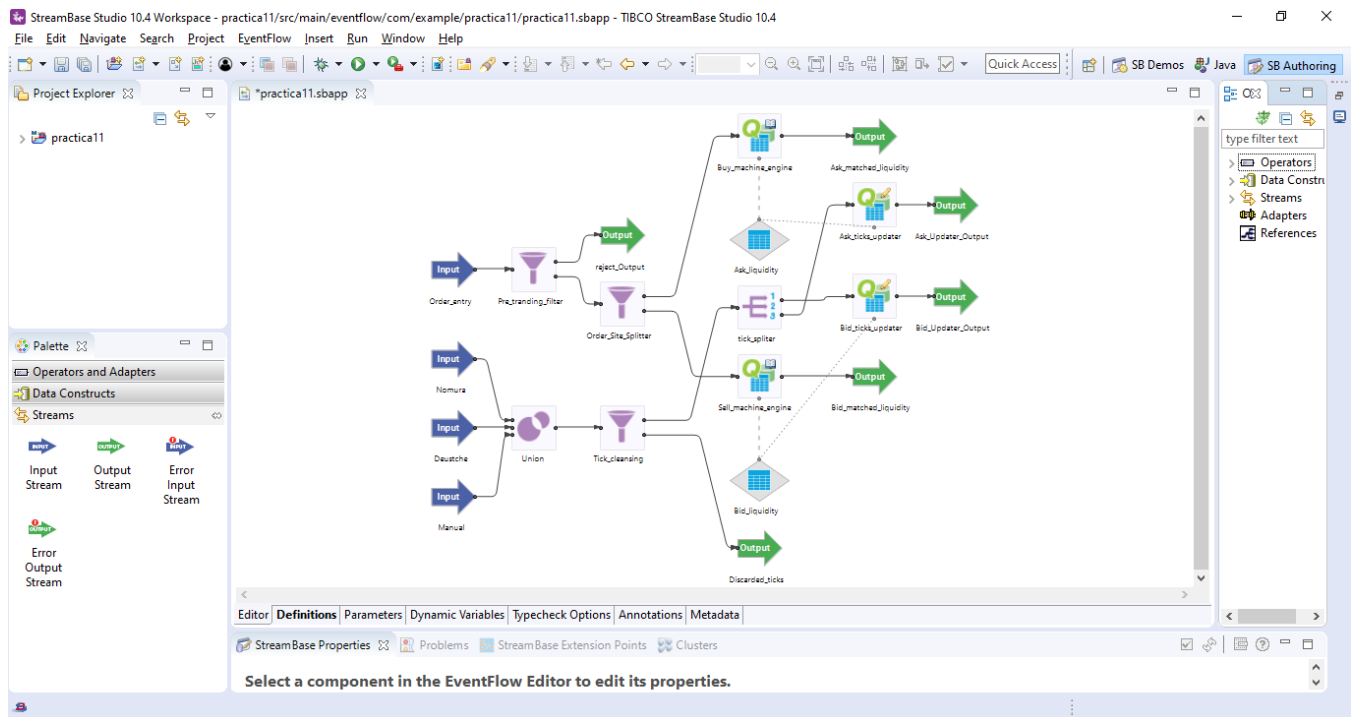
Para buscar liquidez a las órdenes de compra vamos a añadir una query que debe buscar en la tabla ASK_LIQUIDITY (ojo, esto va a ser un lío de conexiones entre operadores, tablas y demás, reordena todo lo que quieras para que quede claro):



Y ahora especificamos la operación, que será solo lectura:



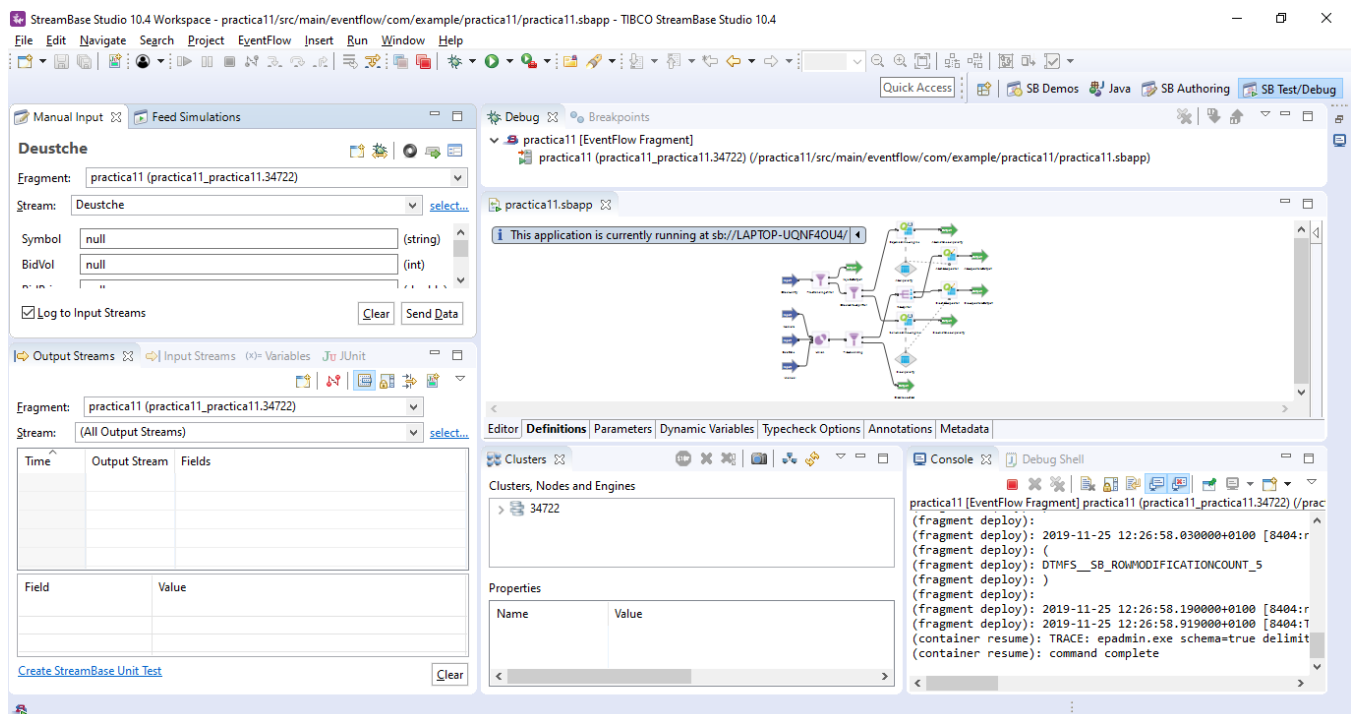
Ahora vamos a añadir un flujo para poder ver contra que liquidez cruzaría la orden y definimos como va a ser la salida del motor de cruce de órdenes:



Ejecutando el proyecto

Para ejecutar el proyecto pulsaremos sobre el botón verde con la imagen del “play” y el entorno cambiará la perspectiva a la de depuración y ejecución si así se le permite.

Al pasar a esta perspectiva, la mitad izquierda de la pantalla es la que nos interesará más, en la parte superior están los flujos de entrada y en la parte inferior los flujos de salida:



Inyección de liquidez manual

Primero se realiza una inyección de liquidez de forma “Manual”, colocando los datos solicitados y lo agregamos dándole a sen data, luego se observa el output que la liquidez esta reflejada

StreamBase Studio 10.4 Workspace - practica11/src/main/eventflow/com/example/practica11/practica11.sbapp - TIBCO StreamBase Studio 10.4

File Edit Navigate Search Project EventFlow Insert Run Window Help

Manual Input Feed Simulations

Manual

Fragment: practica11 (practica11_practica11.34722)

Stream: Manual

Symbol: EURUSD (string)

BidVol: 1000 (int)

BidPrice: 0.75 (double)

☒ Log to Input Streams

Clear Send Data

Output Streams Input Streams Variables JUnit

Fragment: practica11 (practica11_practica11.34722)

Stream: (All Output Streams)

Time	Output Stream	Fields
15:22:40	Bid_Updater_Output	Symbol=EURUSD, BidVol=1000, BidPrice=0.75, AskVol=2000, AskPrice=0.85
15:22:40	Ask_Updater_Output	Symbol=EURUSD, BidVol=1000, BidPrice=0.75, AskVol=2000, AskPrice=0.85

Field Value

Create StreamBase Unit Test

practica11 [EventFlow Fragment] pra
(fragment deploy): 2019-1
(container resume): TRACE
(container resume): comma

Entrada de órdenes

StreamBase Studio 10.4 Workspace - practica11/src/main/eventflow/com/example/practica11/practica11.sbapp - TIBCO StreamBase Studio 10.4

File Edit Navigate Search Project EventFlow Insert Run Window Help

Manual Input Feed Simulations

Order_entry

Fragment: practica11 (practica11_practica11.34722)

Stream: Order_entry

Symbol: EURUSD (string)

Vol: 800 (int)

Price: 1 (double)

☒ Log to Input Streams

Clear Send Data

Output Streams Input Streams Variables JUnit

Fragment: practica11 (practica11_practica11.34722)

Stream: (All Output Streams)

Output Stream	Fields
0. Ask_Updater_Outp...	Symbol=EURUSD, BidVol=1000, BidPrice=0.75, AskVol=2000, AskPrice=0.85
0. Bid_Updater_Output	Symbol=EURUSD, BidVol=1000, BidPrice=0.75, AskVol=2000, AskPrice=0.85
0. Ask_matched_liqu...	Symbol=EURUSD, BidVol=1000, BidPrice=0.75, AskVol=2000, AskPrice=0.85
0. Bid_matched_liqu...	Symbol=EURUSD, BidVol=1000, BidPrice=0.75, AskVol=2000, AskPrice=0.85

Field Value

Create StreamBase Unit Test

practica11.sbapp

This application is currently running at sb://LAPTOP-UQNF40U4/

Editor Definitions Parameters Dynamic Variables Typecheck Options Annotations Metadata

Clusters, Nodes and Engines

practica11_practica11

practica11_practica11

com_example_pr

Properties

Name Value

General Query Operation Fallback (Current Table File...) Output Group Options

Input Fields

Current Table Fields

From "current": All None Prefix: Suffix:

Action Field Name Expression

Additional Expressions

Function...

Cho

type filter

Aggr

Ext

Stat

Win

Simple I

BSOI

Error