**UNIVERSITATEA BABEŞ-BOLYAI CLUJ-NAPOCA**

**FACULTATEA DE MATEMATICǍ ŞI INFORMATICǍ**

**SPECIALIZAREA INFORMATICĂ**

**LUCRARE DE LICENŢĂ**

**Flux continuu de date pe baze de date**

**Conducător ştiinţific**

Dr. SURDU Sabina, Lector Universitar

**Absolvent**

**Crișan Camelia Daniela**

**2018**

Contents

[Capitolul 1 3](#_Toc516571318)

[Introducere 3](#_Toc516571319)

[Capitolul 2 4](#_Toc516571320)

[Descrierea aplicației 4](#_Toc516571321)

[2.1 Events 4](#_Toc516571322)

[2.2CEP(Complex Event Processing) 4](#_Toc516571323)

[2.3 Ferestrele de evenimente(Window events) 5](#_Toc516571324)

[2.4 Hopping Windows 6](#_Toc516571325)

[2.5 Snapshot Windows 7](#_Toc516571326)

[2.6 Count Windows 7](#_Toc516571327)

[2.7 Aggregations 7](#_Toc516571328)

[2.8 Query 7](#_Toc516571329)

[2.9 LINQ 7](#_Toc516571330)

[Capitolul 3 8](#_Toc516571331)

[Tehnologii utilizate 8](#_Toc516571332)

[3.1Streaminsight 8](#_Toc516571333)

[3.2 Event flow debugger 9](#_Toc516571334)

[Capitolul 4 11](#_Toc516571335)

[Studiu experimental 11](#_Toc516571336)

[4.1 Studiu 11](#_Toc516571337)

[4.2Comparatie cu studiul experimental Aurora 12](#_Toc516571338)

[Capitolul 5 13](#_Toc516571339)

[Concluzii 13](#_Toc516571340)

[6.Bibliografie 13](#_Toc516571341)

# Capitolul 1

## Introducere

Bazele de date au evoluat și au început să crească din ce în ce mai mult datorită fluxului imens de informație primit.Ținând cont că în ultimul deceniul tehnologia și-a făcut loc în viețile noastre ca o parte esențială a acesteia, am devenit dependenți de utilizarea acestora și a informațiilor pe care acestea le dețin, astfel acesta este unul din fluxurile importante de date. Accesul omniprezent la date ușurează utilizatorilor munca zilnică și viața personală, astfel eliminarea barierelor tehnologice de scară și economie sunt realizate pe diferite platforme prin calul utilitar.Aceste tendințe aduc un impact major asupra aplicațiilor IT, un grad sporit de conștientizare a timpului, deoarece care fiecare punct de date care sosește reprezintă un punct de contact al clientului astfel se poate observa cum fiecare trend se diminuează odată cu timpul. Datorită concurenței în ceeea ce privește aplicațiile și a modului în care evolueză, acestea necesită o trecere la o pocesare bazată pe evenimente bine pozitionate într-o perioadă de timp. Astfel am ales să studiez impactul fluxurilor de evenimente, în diferite condiții de evaluare, pentru a putea observa viitoarele oportunități de optimizare a aplicațiilor și a datelor existente.

# Capitolul 2

## Descrierea domeniului aplicației

### 2.1 Evenimente

Pentru a putea vorbi despre evenimente și procesarea acestora trebuie mai întâi să înțelegem ce sunt evenimentele. Evenimentul este o acțiune trecută care poate fi reprezentată ca obiect ( în programare) și care poate fi analizată în diferite moduri. De obicei fiecare flux de evenimente are anumite caracteristici, de exemplu un flux de evenimente provenite de la mașini din trafic. Toate acestea vor conține același tip de informație, dar fiecare v-a avea un timp de apariție diferit, o viteză a mașinii diferită și o marcă diferită. Astfel vom putea defini, un eveniemnt generic care să fie o specificare pentru un set de astfel de evenimente și să le încapsuleze intr-o sferă semantică, grupându-le ca tipuri de evenimente.

Evenimentul poate fi descris ca și o colecție de atribute împărțite pe trei categori : antet, încărcătura utilă și conținut deschis (opțional). Antetul este constituit din meta-informația evenimentului spre exemplu timpul de apariție al acestuia. Încărcătura conține o colecție de atribute specifice fiecărui tip de eveniment în parte, fiecare din aceste atribuite poate fi un tip de dată simplu precum un întreg (int), un sir de caractere (string) sau poate fi o structură complexă de date (obicecte).

Printre atribute se regăsesc timpul de apariție, este un tip de dată care se definește ca și un punct in timp, precizia acestuia fiind dată de granularitatea temporală care se aplică tipului de eveniment. Granularitatea temporală, denotă un atom temporal dintr-un punct de vedere al aplicației precum : minutul, ora sau ziua. Din acestea rezultă timpul de apariție fiind definit ca si atribut in care precizia timpului este dată de granularitatea temporală a evenimentului. [3]

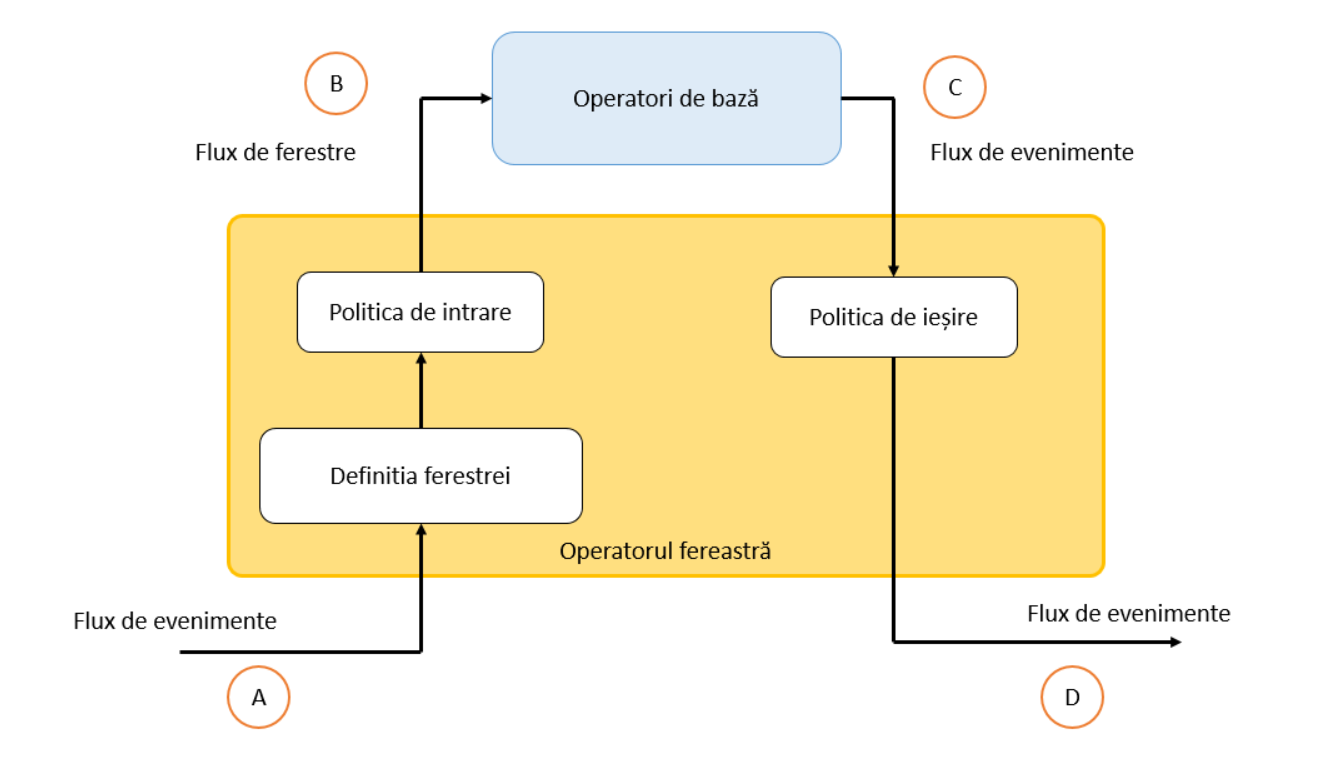
### 2.2CEP(Complex Event Processing)

Procesarea de evenimente se bazează pe analiza a ceea ce se întâmplă in fluxul de date și pe rezultate concrete pe baza acestora. Procesarea complexă de evenimente ne permite analiza și corelarea cantităților mari de date sub formă de evenimente cu scopul detectării unor șabloane sau situații de interes în timp real. Structura este formată din mai multe tipuri de adaptori care pot face legătura între sursele de date și sistemul de analiză.

### 2.3 Ferestrele de evenimente(Window events)

În general, aplicațiile care procesează eveimente in timp real, necestiă o performanță ridicată in calculul diferitelor operatii pe subseturi de date. In Streaminsight aceste subseturi sunt definite ca si ferestre, acestea sunt de 3 feluri : fereastră hopping(Hopping Window), fereastră de numarare(Count Window), fereastră instant(Snapshot Window). Fluxul de evenimente pentru a putea fi procesat este transformat într-un stream fereastră **IQWindowedStreamable<T> , astfel operațiile de bază vor putea fi executate. Aceste evenimente sunt grupate după tipul de ferestră folosit : timp sau număr, ferestrele fiind caracterizate după operatorul acesteia.**

**Operațiile de bază readuc stream-ul de ferestre la un stream de evenimente, aceste operații se împart în două categorii : agregările și operațiile de generare de evenimente de ieșire pentru un set de evenimente de intrare.În cadrul aggregărilor avem operații precum sum, avg, count sau operații definite de utilizator. Într-un window stream pot fi aplicate una sau mai multe aggregări, atât asupra ferestrei cât și a datelor din acesta, fiind posibilitatea de a crea un alt flux de date care să conțină date aggregate. Operațiile de generare pot fi TopK sau operații definite de utilizator, acestea sunt utilizate peste un flux de ferestre și generează zero sau mai multe evenimente în cadrul unei ferestre ca și rezultat al unul calcul.**

 **Fiecare fereastră este alcătuită din trei părți: definția specifică,tranformarea temporală a datelor de input și transformarea temporală a datelor de output. Configurația unei ferestre:**

Configurația unei ferestre (adaptată după documentația oficială, în limba română)

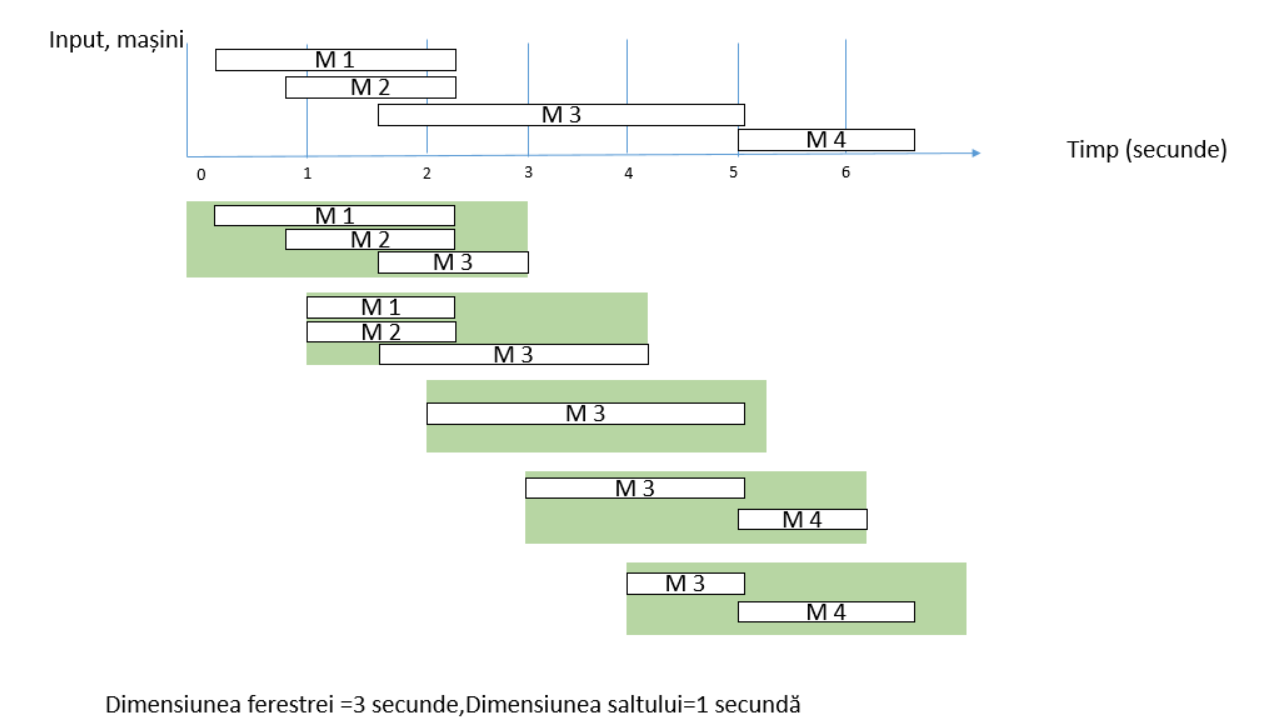
Tansformările temporale sunt făcute după anumite politici, politica de input este singulară și se referă la timpul de intrare și ieșire al eveimentelor, în care ficeacre operație poate vedea timpul evenimentului doar in interiorul ferestrei, iar ca politică de ieșire, fiecare tip de fereastră are politica ei speicifică. Ferestrele snapshot adaptează timpul final al evenimentului după timpul final al ferestei.Rezultatele date de ferestrele hopping sunt așezate în ordine după timpul final al frestrei, iar rezultatul dat de ferestrele de numărare

### 2.4 Hopping Windows

Acest tip de ferestră definește un subset de evenimente care se încadrează într-un interval de timp și care aceptă să se efectueze diferite calcule sau aggregări asupra datelor. Timpul acestor evenimetne este împărțit în intervale de timp regulare.

Numele de hopping vine de la saltul temporal făcut de o fereastră în momentul începerii unei ferestre, adică timpul de întârziere față de fereastra anterioară. Aceste ferestre sunt compuse din două intervale temporale: saltul temporal și dimensiunea ferestrei, care este creeată odată la fiecare salt temporal. Atunci când saltul temporal și dimensiunea ferestrei sunt egale, ne aflăm într-un caz particular, deoarece ferestrele nu se suprapun, acest model de fereastră se numește fereastră thumbling.

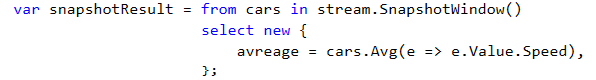
Evenimente sunt împărțite de această fereastră în bucăți temporale pentru a putea fi procesate, aceasta se datoreaza politicii de input a datelor. Operațiile se efectuează chiar dacă față de fereastra anterioară nu s-a modificat structura evenimentelor, deoarece aestea sunt impărțite în bucăți temporale. De exemplu:



### 2.5 Snapshot Windows

Acest tip de fereastră definește deasemenea evenimente dintr-o perioadă de tmp asupra cărora se pot efectua diferite operații. Totuși diferența față de hopping windows este impărțirea timpului, acesta fiind delimitat de începutul și de sfârșitul evenimentelor, astfel dimensiunile temporale vor fi inegale. Dimensiunile ferestrelor sunt egale cu cele temporale, dar doar cân in acel moment temporal există cel puțin un eveniment.

Definirea acestui tip de fereastră nu are nevoie de nici o componentă deoarece dimensiunile ferestrelor vor fi alocate dinamic după apariția evenimentelor, acestea fiind adaptate de politica de intrare a datelor. Pot fi folosite pentru implementarea ferestrelor de alunecare, acestea fiind foarte utile în momentul în care vrem să aplicăm aggregări asupra unui operator de tipul group, deoarece fluxul de evenimente este comprimat.

 Aceasta este construcția unui calcul al mediilor vitezelor mașinilor dintr-un interval de timp, în cazul meu am calculat viteza medie a mașinilor într-un interval de 2h :

### 2.6 Aggregations

### 2.7 Stream

Stream-ul, adică fluxul de date este o secvență de evenimente continue disponibile o perioadă de timp.Acesta se poate defini ca un set de două valori in care prima valoare este un tuplu, iar cea de-a doua reprezintă timpul apariției tuplului.

### 2.8 Query

#### 2.8.1 CQL

Asupra fluxurilor de date, se pot creea interogări simple pentru care este suficient un limbaj relațional precum SQL, dar cum aceste query-uri au început să devină din ce în ce mai complexe introducând în structura lor ferestre(ex. Thumblig window), aggregări sau join-uri între fluxuri, aceste limbaje simple au fost depășite, astfel s-a creeat un nou tip abstract pentru interogări continue numit Continoious Query Language(CQL).

Acesta se poate împărți în trei caterogi: relation-to-relation care are ca și input una sau mai multe relații si ca output are tot o relație, stream-to-relation care are ca input un flux si produce ca output o relatie și reation-to-stream care are ca input o relație si generează un flux ca și date de output.

Un plan de interogare este compilat pentru o interogare continuă cand acesta este înregistrat, acesta proceseaza cozi de operatori. Acești operatori pot fi de inserare sau de ștergere. Elementele cozilor dintr-un query sunt reprezentate de fluxuri sau relații și are doi operatori operatorul care produce și cel care consumă. Fiecare plan are alocat câte un rezumat care de cele mai mule ori este folosit pentru a afla starea curentă a relației.Execuția se reazlizează pe baza unui plan care depinde de timpul apariției elementului procesat.

Pentru a crește performanța, avem nevoie de câteva înbunătățiri la interogările acestea brute. Deoarece într-un singur plan de interogare pot apărea mai multe rezumate, acestea uneori pot fi redundante astfel ele trebuie înlocuite de niște stub-uri care implementează aceeași interfață. Constrângerile sunt un mod bun de a crește performanța, deoarece dacă acesrea sunt specificate dinainte, iar datele care vin au un anumit șablon se poate reduce dimensiunea rezumatului.

#### 2.8.2 Query development în Streaminsight

Streaminsight acceptă modelarea datelor după modelul realității, adică un eveniment se întâmplă la o anumită perioadă de timp, astfel interogările pe aceste date trebuie să aibă o fundație algebrică bine consolidată

### 2.9 LINQ

### 2.10 Dynamic SQL

# Capitolul 3

## Tehnologii utilizate

### 3.1 Streaminsight

#### 3.1.1 Despre

Tehnologia principală utilizată este Streaminsight, aceata este o platformă care poate fi folosită pentru a dezvolta si implementa aplicații care proceasează evenimente complexe(CEP), aceasta are o arhitectură de nivel înalt în privința procesării fluxurilor de date și datorită implementării bazate pe Microsoft .NET Framework, aceasta permite dezvoltarea rapidă si eficientă a unor aplicații de procesare a evenimentelor. Fluxurile de evenimente conțin în general date care trebuie anazilate sau procesate, precum datele din aplicațiile de tranzacționare financiară, analiză web sau analiză operațională. Astfel a venit solutia, Steaminsight poate dezvolta CEP care oferă direct rezultatul fără a fi nevoie de foarte multe resurse asftel reducându-se costurile axtragerii, analizei si corelărilor de date și îți permite să monitorizezi, să gestionezi și să furnizeze datele pentru anumite condiții.Această tehnologie este folosită în special pentru a monitoriza date din mai multe surse pentru modele, tendințe, excepții și oportunități semnificative.

Are o bună optimizare în ceea ce privește performanța si transferul de date, deoarece acesta implementează o arhitectură care suportă paralelizarea la nivel înalt a interogărilor continue asupra datelor. Transferul de date este facilitat de utilizarea memoriei cache și de calculul incremental al rezultatelor. Datele nefiind stocate, procesându-se la momentul apariției acestuia, se obține un timp de latență scăzut.

Aplicațiile care folosesc această tehnologie sunt scrise de obicei sub platforma Microsoft.NET folosind limbajul C#, în care se regăsește limbajul interogărilor integrat (LINQ) care, deoarece este foarte cunoscut și foarte folosit poate reduce costurile de dezvoltare ale aplicației.

Tipurile de șcenarii pe care StreamInsight le suportă sunt : aplicațiile DLL hostate în care se poate interga complet, aplicații de tip server care este folosit de mai mulți clienti sau în aplicații autonome sau hostate, care sunt alcătuite din mai multe servere.

Monitorizarea și administrearea funcțiilor sunt incluse in StreamInsight, asigură un cost scăzut, iar managementul interfeței și vizualizarea diagnsticului date de acesta asigură monitorizarea și administrarea unei aplicații CEP. Framework-ul de administrare suportă și aplicațiile hostate. Pentru analiză și diagnosticare a aplicațiilor autonome se folosește event flow debugger.

Această tehnologie, poate suporta anumite cerințe precum procesarea a peste 10.000 de evenimente pe secundă, un timp de latență scăzut în cazul agregărilor, avertizări în situațiile în care ceva s-a defectat, nu merge conform standardelor sau mentenanță susținută pe condiții.

Pentru un client cu o aplicație care necesită o procesare a unui flux de date impresionant, la o latență scăzută, precum aplicațiile web, acestea includ anumite cerințe precum abilitatea de a procesa peste 100.000 date/secundă, abilitatea de a naviga și prezenta o pagină pe baza fluxului de analiză și descoperirea șabloanelor în fluxurile de date cu un timp de răspuns imediat.

Pentru a putea administra rețele imense, precum utilitățile, internetul, aceste aplicații au anumite specificații. Aplicație trebuie să poată proesa peste 100.000 date/secundă care provin din milioane de fluxuri de date, acestea trebuie să poată fi impărțite pe mai multe nivele de agregare, eficientizarea acestora prin mutarea în rețele inteligente și un răspuns imediat la diferitele variații de date.

Ca și arhitectură partea de StreamInsight e văzută ca perte de server în care datele venite de la generatorii de fluxuri sunt procesate cu ajutorul interogărilor și generează rezultate pentru consumatori, precum aplicații web. Fluxurile de intrare sunt continue, ele fiind procesate în ordinea apariției acestora în concordanță cu fiecare interogare, iar ca și date de output acestea pot fi distribuite pentru diferite acțiuni.

Procesarea datelor este efectuată cu ajutorul a trei tipuri diferite de date. O listă de eveniemnte bine ordonată, care implementează interfața Ienumerable<>, poate fi accesată în mod iterativ folosind framework-ul LINQ-to-Objects, acesta se numește secvență de date de tragere (pull). O secvență de date de același tip, ordonate, care implementează interfața Iobservable<> și care oferă secvența pentru unul sau mai mulți consumatori, acest acces realizându-se cu ajutorul framework-ului Reactive LINQ, acestea se numesc secevnte de date de impingere (push). Stream-urile temporale sunt sevente de elemente cu caracteristici unice, ele implementează interfața IStreamable<>.

Fluxurile de date temporlae sunt specifice StreamInsight-ului, fiecare dată componentă a acestui flux infinit de date este constituită din două părți încărcătura utilă și timpul evenimentului, atât timpul de început cât și de sfârșit.Aceste fluxuri au anumite caracteristici precum fiecare eveiment din cadrul fluxului are un timp de apariție, printre evenimentele fluxului se regăsesc o serie de evenimente speciale care definesc gradul de completare al fluxului odată cu trecerea timpului, acestea fiind Current Time Increment(CTI) events, aceste date pot fi privite și ca niște bariere temporale deoarece, după ce un astfel de evniment apare în flux cu un anumit timp, nici o altă dată care îl urmează nu poate avea un timp de apariție mai mic.

În secțiunea 2.1 am descris în mod general un eveniment, acum vom putea vedea care din caracteristicile lui se aplică in Streaminsight. Datele din fluxurile temporale sunt încapsulate în evenimente, acestea fiind unitățile de bază ale unei aplicații dezvoltate în această tehnologie.În antet sunt incluse tipul evenimentului precum și unul sau mai multe momente de timp, iar încărcătura utilă este construită ca o structură .NET, câmpurile definite în acesta fiind bazate pe tipurile de date existente in sistemul .NET. Inserarea și CTI(incrementarea timpului curent) sunt cele două tipuri de evenimente pe care StreamInsight le poate susține.Inserarea adaugă evenimentele în fluxul de date,împreună cu timpul se start si de final al acestuia. În schimbevenimentele de tipul CTI au un singur câmp acesta fiind timpestampul actual, astfel indică faptul că in fluxul de date nu mai există alte evenimente de procesat a căror timp de apariție nu mai corespunde.

Ca și entități ale unei aplicații, există cinci tipuri de bază. Sursa, care generează datele, suportă interogările de tipul LINQ în fluxurile continue sau în secvente de evenimetne, acestea pot implementa interfețele specializate pentru asta precum IQueryable, IQbservable și IQStreamable. Ca și consumator avem sink-ul, acesta este implemtnat după tipul sursei, dacă sursa implementează interfețele IObservable/IStreamable, atunci sink-ul v-a implementa un observer sau dacă sursa implementează IStreamable sink-ul v-a mai putea implementa interfața IStreamableSink. Subiectul este atat generator cât și consumator de date, fiind de folos și în cazul partajării calculelor între consumatori și producători, față de sursă acesta nu poate implementa în mod direct interfața IStreamable. Conexiunea între surse și sink-uri este realizată de binding, care este un executabil, iar execuția acestuia se numește proces.

#### 3.1.2 Monitorizare

#### 3.1.3

### 3.2 Event flow debugger

Majoritatea aplicațiilor produc fluxuri mari de date, iar aplicațiile de administrare ale acestora trebuie sa aibă o latență scăzută și relații complexe între fluxuri. Datorită elementelor temporale din fluxurile de date, determinarea unui răspuns al operațiilor cât mai corect este esențială, dar și determinarea cauzelor problemelor interogărilor într-o eventuală eșuare.

Astfel s-au creeat nevoi pentru un nou tool care să poată manageria foarte multe date, dar care să reducă problema spațiului de căutare. Cerințele de riguroase de consitență trebuie deasemenea administrate, iar soluția sau diagnosticul trebuie să fie ușor de intuit.

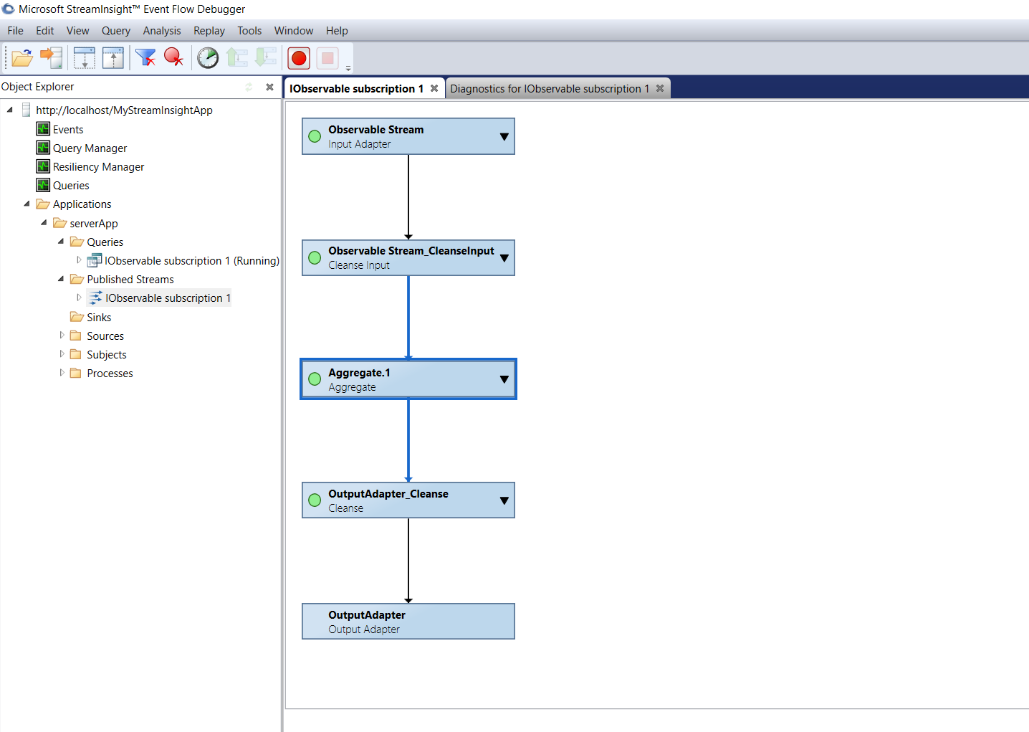
Event flow debugger este o interfață grafică autonomă dată de Microsoft StreamInsight, care te ajută ca developer sau ca administrator al unei aplicații de procesare complexă a evenimentelor, să faci debugging asupra unei interogări creeate peste un flux de date

Trebuie înțeles faptul că acest tool nu este un debugger de controlare al fluxului. Controlul fluxului construiește programele in diferitele limbaje de progrmare, dar pentru ca acestora să li se facă debugging, acceptând breakpoints și rulând codul în mod normal până la întâlnirea unuia, se pot vedea stările actuale ale sistemului, vairabilele existente în cod, se poate intra sau ieși din anumite funcții totul până la finalul execuției, terminate în mod normal sau de utilizator. Pe altă parte event flow debugger analizează eveniementele raportându-se la trecerea timpului și se mută de la o interogare CEP la o alta, de la un nivel diferit. Aici debuggingul arată cum un eveniment are efect asupra fluxului de date și cum se generează evenimente noi după ce au fost manipulate evenimentele de intrare. Astfel față de un debuggin al fluxului de control acesta arată modul în care operatorii afectează fluxul de date și nu cum funcționează ei mai exact. Astfel vom putea înțelege impactul evenimentelor aspura celui analizat și implacutl fiecărui eveniment în parte asupra celorlalte evenimente.

Acest tool autonom își propune să facă debugging pe o urmă a unui flux de evenimente bazându-se pe intergistrarea în timp real a unei anumite interogări, fiind conectat la un server în timp real sau această urmă mai poate fi generată în afara degubber-ului, într-un fișier și apoi încărcată pentru a putea fi analizată. Debugger-ul își mai propune să monitorizeze serverul, astfel să se poată obține identificarea stării despre obiectele aplicației și despre listele Object Explorer. Cu ajutorul interfeței putem vedea modoul de funcționarea al inteorgărilor, le putem proni sau opri, le putem activa sau dezactiva în timpul execuției.

Se poate conecta la o aplicație StreamInsight ca si un client sau local pentru a înregistra evenimentele rezultate după efectuarea a unei sau mai multor interogări, dar debugger-ul se mai poate folosi fără a fi conectat la un server, pentru a analiza interogările pe urmele de date obtinute offline. Astfel obținem o felxibilitate în modul de lucru, deoarece putem face deubgging pe o anumită interogare cât timp fluxul este în desfășurare sau putem să o testăm pe urmele de date obținute offline.

Înregistrarea interogărilor eveniemntelor în timp real se poate face dacă la event flow debugger este conectat un server live de StreamInsight. Ca evenimentele să fie stocate de debugger, serverul trebuie să fie unul autonom și să ruleze pe același computer.

Totuși pentru ca o aplicație să fie conectată la degubber aceasta trebuie să iși activeze serviciile web, clientul trebuie să aibă permisiuni asupra serverului pentru a se putea conecta la serevr. Apoi din interfață se v-a conecta la o adresă endpoint care a fost inițializată în server. După ce s-a realizat conexiunea în secțiunea de Object Explorer, vom putea vedea o ierarhizare a obiectelor unde pot fi găsite interogările pentru a se face deubgging. Apoi procesul de înregistrare poate începe. Putem observa partea de ierarhizare :

StreamInsight poate fi atât o aplicație autonomă cât și una hostată, dar dacă aceasta nu are disponibil un serviciu web nici un fel de client nu se poate conecta, astfel nici debuggerul nu v-a putea face legătura cu acesta. Astfel s-a creeat necesitatea unui mecanism de diagnosticare și rezolvare a problemelor în cazul acestor situații. Pentru rezolvarea acestei probleme se stochează istoricul evenimentelor venite de la un server cu ajutorul utilitarului trace.cmd care este inclus în StreamInsight. Pașii de manipulare ale acestui isotric pentru ca mai apoi datele să fie procesate de event flow debugger sunt : creerea unui fișier cu extensia .etl care să înregistreze datele venite de la server, acestea sunt înregistrate cu ajutorul comenzii trace.cmd start, în momentul în care s-a înregistrat suficientă informație se poate opri înregistrarea cu ajutorul comenzii stop, după aceea datele obținute vor putea fi încărcate manual in event flow debugger pentru a putea analiza datele. În acest caz secțiunea de diagnosticare a obiectelor pe nivele nu poate fi afișată. Utilitarul folosit în acest caz este un script bazat pe comanda Windows Logman, care se foloseste de Event Trace for Windows (ETW) pentru colecatera datelor. În timpul procesului de încărcare unele date se pot pierde, dar debugger-ul v-a anunța acest eveniment, acest lucru se poate evita dacă zona tampon a fișierului își mărește dimensiunea sau se alocă mai multe zone tampon pentru un fișier.

Pentru analizarea interogărilor event flow debugger are anumite funcționalități cheie. Pentru înțelegerea limbajului integrat al interogărilor (LINQ) și a modului în care interogările din acest limbaj acționează asupra fluxului de date, debugger-ul pune la dispoziție vizualizarea planurilor de interogare, a operatorilor si a stream-urilor de evenimente. Datele din fluxul de date sunt puse la dispoziție pentru a putea fi analizate, atât cele de intrare cât și de ieșire, împreună cu calculele intermediare aferente. Pentru fiecare dată se pune la dispoziție timpul de start și de sfârșit si încărcătura utilă. Operatorul Group partiționează fluxul de date, iar operatorul Apply îl modifică, cu toate acestea fluxul poate fi urmărit în debugger. Problema spațiului de căutare și corelarea între stagiile fluxului de evenimente este redusă de abilitatea de analiză globală a acestuia. Ajută la înțelegerea propagării evenimentelor printr-o interogare a fluxului, deoarece poate trece prin pașii de execuție ai interogării raportându-se la trecerea timpului. Asupra timpului si încărcăturii unui eveniment se răsfrânge impacutul interogărilor, acesta putând fi analizat și înțeles prin vizualizarea schimbărilor stărilor evenimentelor. Putem observa viitoare date care trebuie procesate și care vor afecta rezultatul curent.

Una din caracteristicile debugger-ului este reluarea, aceasta te ajută să treci de la un operator la altul și să poți observa progresul de-a lungul fluxului sau poți să aplici puncte de întrerupere aspura grafului de interogare care activează acțiunile până la întâlnirea acestuia. Pentru a putea găsi ceea ce a provocat starea actuală a evenimentului, trebuie să putem să ne intoarcem la sursa problemei și să găsim secvența de operații dorită. O altă caracteristică este analiza propagării evenimentelor, astfel se v-a putea observa efectul eveniementelor asupra fluxului de date, odată cu schimbarea stării acestuia, dar și efectul asupra altor evenimente sau cauza pentru care s-au generat evenimente noi.

Interogările din StreamInsight sun descompuse în noduri de excuție, acestea fiind reprezentate de operatori. Operatori suportați de query-uri sunt Select (Filter), Project, Import, Export, Group-and-Apply, Join, Multicast, Union, Top-K, AlterLifetime, Advancetime și Cleanse [4]. Operatorii din graful de interogare s-ar putea să nu corespundă in totalitate cu cei din interogarea LINQ, deoarece unii opeatori sunt introduși în mod automat precum cel introdus de optimizatorul de interogare Clenease, care sortează datele și care nu se regăsește in interogarea LINQ. Totuși acestea nu îngreunează ințelegerea grafului interogării.

Fiecare grilă de evenimente are un tabel operator, fiecare tabel având ca denumire numele operatorului pe care îl reprezintă și tipul acestuia. Filtrul, fiind un text box, acceptă cod C#, care să ofere condiționarea dorită pentru a obține anumite rezultate. Câmpurile evenimentului este o altă componentă a grilei de evenimente, acestea conțin tipurile evenimentelor definite de utilizator pe tot parcursul evenimentelor interne. Aceste câmpuri sunt atribuite în mod automat, dar pot fi modificate prin adăugarea sau eliminarea unor câmpuri.Tabela creeată cu aceste câmpuri v-a fi afișată în ordinea stabilită de debugger, evenimentele putând fi exportate intr-un fișier pentru o analiză viitoare excutată de alte programe.

Când vine vorba de operatorul de grupare și aplicare îl includem într-o caterogie specială, gruparea după cum ii spune și numele grupează evenminetele la momentul intrării asupra cărora se aplică mai mulți operatori, pentru datele de ieșire se folosește operarotul de grupare unificat. În momentul extinderii unui astfel de nod se vor afișa toate fluxurile de date grupate după o valoare numită cheie pentru care s-a efectuat operatorul de aplicare.

Alte uitlități ale debuggerului sunt posibilitatea de a vedea graful de interogare în ansamblu, trecând sau nu prin operatori. Este foarte ușor de folosit, deoarece acțiunile posibile sunt reprezentate atât prin butoane cât și în opțiunile meniului. În momentul conectării debuggerului la server , orice eroare survenită pe parcurs v-a fi returnată, în limba engleză, toate celelate erori definite de utilizator vor fi folosite pentru diagnosticarea diferitelor probleme. Operatorii pot fi vizualizați deodată sau pot fi inchiși cu ajutul funcționalităților de extindere sau restrângere. Pentru o mai ușoară folosire a debugger-ului, meniul acestuia pune la dispoziție o varietate de instrumente prin care ne putem personaliza anumite specificații precum data și ora pentru zona de timp pentru toate operațiile din graful de interogare sau putem alege segemnetele de date care să fie încărcate din fișiere sau dimensiunile de încadrare ale înregistrării unui flux de evenimente în momentul în care acesta este conectat la un server.

Acesta nu este numai un instrument de depanare, el mai deserveste și ca un instrument de moitorizare pentru serverul de StreamInisght. În interfața acestuia vom putea regăsi pe lângă operatori niște statistici ale obiectelor cu privire la stările acestora în momentul rulării. De asemenea se pot vizualiza date despre anumite interogări care au fost înregistrate de către server, precum date despre întărzierea, consumul si producerea de date, dar și memoria utilizată pentru efectuare unei interogări.

# Capitolul 4

## Studiu experimental

### 4.1 Studiu

Ca prim experiment, am început prin a studia un flux de evenimente, în care fiecare eveniment este reprezentat de o masină care are ca și date viteza cu care merge și numele mașinii. Pentru a putea vedea impactul asupra diferitelor ferestre de timp, trebuie calculat mai întai care ar fi dimensiunea optimă a ferestrelor. Astfel la terminarea unui flux de date, după ce acestea au fost salvate într-o bază de date uniform distribuită, se vor prelucra date, de la 1...n până cand marja de eroare dintre rezultatul obtinut si rezultatul mediu obtinut după terminarea interogrilor este minim.

### 4.2Comparatie cu studiul experimental Aurora

# Capitolul 5

## Concluzii

# 

# Bibliography

**There are no sources in the current document.**

# 6.Bibliografie

[1] [*Jason H*](https://github.com/JasonWHowell)*,*https://msdn.microsoft.com/en-us/azure/stream-analytics/reference/hopping-window-azure-stream-analytics, Last Updated: 4/27/2018

[2] *Torsten Grabs, Roman Schindlauer, Ramkumar Krishnan, Jonathan Goldstein, and Rafael Fernández ,* „Introducing Microsoft StreamInsight”, Published: September 2009, Revised: May 2010

[3] *O. Etzion and P. Niblett* , „Event Processing in Action”, Manning Publications, 2010.

[4] *N. Leavitt*, "Complex-Event Processing Poised for Growth," in Computer, vol. 42, no. , pp. 17-20, 2009. doi:10.1109/MC.2009.109

[5] https://msdn.microsoft.com/en-us/library/ff518532(v=sql.111).aspx