Модел на потоци от данни

Владимир Димитров

Abstract: *Traditional Database Management Systems (DBMS) are suitable for execution of single queries on finite data sets stored in the database. There are many new applications, like network monitoring, financial analysis, manufacturing systems and sensors networks, which have to execute long or continuous queries on infinite and unbounded data streams. These streams could be very rapid and their characteristics and load with queries could vary with time and the resources could be bounded at the same time. This paper investigates data management for data streams class of applications.*

Key words: *Data Streams, Data Base Management Systems.*

ВЪВЕДЕНИЕ

Потоците от данни са разширение на традиционните СУБД (Системи за управление на бази от данни). Интернет превърна комуникациите между компютрите в рутинни операции, а това от своя страна доведе до появата на клас приложения, които не се вписват в рамките на традиционните СУБД. Типичната система за бази от данни е хранилище. Въвеждането на данните е част от езика за манипулация на данните или за тази цел се използва специализирана помощна програма за зареждане на данните. С други думи, зареждането на данните е под контрола на СУБД. Има, обаче, приложения, в които СУБД не може да контролира скоростта, с която пристигат данните и трябва да бъдат заредени в базата от данни. Пример за такова приложение е Уеб портал, който записва всяка итерация извършвана с него от всеки потребител взаимодействащ с него. Поредицата от URL представяща тези заявки пристига с много висока скорост и се определя само от клиентите на портала.

СИСТЕМА ЗА УПРАВЛЕНИЕ НА ПОТОЦИ ОТ ДАННИ

За да могат да бъдат изпълнявани заявки върху потоците от данни са необходими някои нови механизми. Една СУБД не може едновременно да зарежда данните от висикоскоростните потоци и заедно с това да изпълнява заявки написани на SQL. Нещо повече, не е ясно какво биха означавали някои видове заявки върху потоци. Например, какво означава съединение на два потока, след като никога няма да можем да видим края на потоците?

Една Системата за Управление на Потоци от Данни – СУПД (Data Stream Management System - DSMS) е изправена пред следните предизвикателства:

• Въпреки, че разглеждаме потоците от структурирани записи от данни заедно с конвенционално съхраняваните релации, не можем пряко да прилагаме стандартната релационна семантика за сложните непрекъснати заявки върху тези данни. За тази цел е необходима семантика и език за заявки върху потоци и релации.

• Декларативните заявки трябва да бъдат транслирани в планове на физическо ниво, които са достатъчно гъвкави за да поддържат оптимизации на фино гранулирани решения за режимиране. Такива едни планове са изградени от операции, опашки и синопсиси.

• Постигането на високо бързодействие изисква СУПД да имат възможности за споделяне на състоянието и изчисленията вътре и между плановете на заявките. Освен това, ограниченията върху потока от данни (например наредба, клъстеризация, референтна цялостност) могат да бъдат извличани и използвани за да се намали използването на ресурсите. За това е необходимо да бъдат разработени адекватни техники.

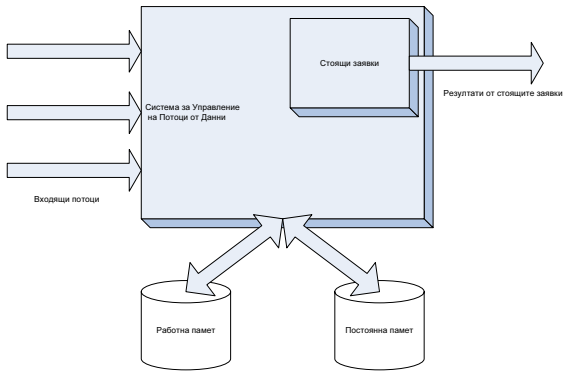
• Понеже данните, системните характеристики и натовареността със заявки може да се изменя във времето, е необходимо да се прилага адаптивен подход за

да се постигне добро изпълнение. За това може да се използва подсистема за непрекъснат мониторинг и реоптимизация.

• Когато скоростта на входящите данни превиши капацитета на СУПД да връща точните резултати за активните заявки, системата трябва да прилага режимиране на натовареността, като използва апроксимации, които намаляват точността в допустими граници. Стратегии за такива апроксимации трябва да се развият.

• Тъй като непрекъснатите заявки са продължителни по своята природа, са необходими инструменти в СУПД за администраторите и потребителите за наблюдение и манипулация на плановете на заявките по време на тяхното изпълнение.

В общи линии,структурата на Системата за Управление на Потоци от Данни – СУПД (Data Stream Management System - DSMS) е представена на следната Фиг. 1.



Фигура 1. Система за управление на потоци от данни.

СУПД приема на вход потоци от данни, а също така и заявки към нея. Заявките са от два вида: **конвенционални „горещи”** заявки и **стоящи** заявки, които се съхраняват от системата и се изпълняват непрестанно върху входящите потоци от данни.

Независимо от това дали заявките са горещи или стоящи, те трябва да бъдат изразени по някакъв начин, така че да могат да връщат отговори върху ограничени части от потоците от данни. Например, нека потоците от данни да са от сензори за радиация разположени на различни места. СУПД не е в състояние едновременно да съхранява и да търси потоците за произволен интервал от време, но тя може да използва плъзгащ се прозорец върху всеки от входните потоци. Последният може да се съхранява върху диска в „работната памет”. Такива плъзгащи се прозорци могат да бъдат зададени върху всеки от входните потоци за данните от последните 24 часа, т.е. в даден плъзгащ се прозорец ще са събрани данните от последните 24 часа за даден сензор. Данните от преди 24 часа могат да бъдат изхвърляни, обобщавани (например, като средно дневни) или копирани изцяло в постоянната памет – архивирани.

Една гореща заявка може да е за средното ниво радиация за последния час от всички сензори разположени в района на Козлодуй. На тази заявка може да се отговори, понеже разполагаме в работната памет с всички данни от всички потоци за последните 24 часа.

Стоящата заявка може да ни алармира, ако някои данни в потока излязат извън определени граници. Тъй като всеки елемент данни влиза системата, той може да бъде анализиран дали излиза извън зададените граници и ако последното се случи ще бъде генериран изход. Този вид заявки могат да бъдат изпълнявани и от самите

потоци, но е възможно да се наложи работната памет да бъде преглеждана, например, ако искаме да бъдем предупредени, когато средната стойност за последните пет минути на даден поток излезе извън границите.

МОДЕЛ НА ПОТОКА ОТ ДАННИ

Този модел на данни е полезен за обсъждане на алгоритми върху потоците от данни. Първо ще направим следните предположения за самите потоци:

• Всеки поток представлява поредица от кортежи. Кортежите имат фиксирана релационна схема (по-точно списък от атрибути), така както това е при релациите (схемата на релацията разглеждана като списък по първичната наредба). За разлика от релациите, поредицата от кортежите в потока може да не е ограничен.

• Всеки кортеж има време на пристигане, т.е. времето, в което е станал достъпен за обработка от СУПД. Последната има възможност да съхрани кортежа в работната памет, в постоянната памет или изобщо да го изхвърли. Кортежът може да бъде обработен и по най-прост начин преди да бъде съхранен.

За всеки поток може да бъде дефиниран „плъзгащ се прозорец" (или просто „прозорец"), който е множество от пристигнали напоследък кортежи. Прозорецът може да е времеви с константа т, което означава, че той се състои от кортежите пристигнали между текущия момент t и t- т. Или прозорецът може да е покоротежен, което означава, че той се състои от последните n пристигнали кортежа, където n е фиксирано цяло число. Ще описваме прозорците върху потока S с нотацията S[w], където w представлява описание на прозореца и може да бъде едно от следните: Rows n, което означава последните n кортежа от потока, или Range т, което означава всичките кортежи пристигнали за последното т време.

Нека Sensors(sensID, temp, time) е поток, чиито кортежи представят прочетената температура temp в определен момент time от сензор с идентификатор sensID. По нормално е всеки сензор да има свой поток, но потоците е възможно да се сливат в един поток извън СУПД. Изразът Sensors[Rows 1000] описва прозорец върху потока Sensors за последните 1000 кортежа. Изразът Sensors[Range 10 Seconds] описва прозорец върху същия поток, който се състои от всички кортежи пристигнали през последните 10 секунди.

ПРЕОБРАЗУВАНЕ НА ПОТОЦИТЕ В РЕЛАЦИИ

Прозорците дават възможност да преобразуваме потоците в релации. По-точно, изразите на прозорците описват релацията във всеки един момент. Съдържанието на тази релация се мени много бързо. Например, в Sensors[Rows 1000] по всяко време може да пристигне нов кортеж от потока Sensors и да бъде изтрит най-стария кортеж. За израза Sensors[Range 10 Seconds], новите кортежи се вмъкват при пристигане и се изтриват 10 секунди след това.

Изразите на прозорците могат да бъдат използвани като релации в един разширен SQL за потоци. Следният пример представя един такъв разширен SQL. Да приемем, че искаме да узнаем коя е най-високата температура регистрирана в СУПД за последния час. Формираме подходящ времеви прозорец и формулираме заявката върху него все едно, че е обикновена релация. Тази заявка изглежда така:

```
SELECT sensID, MAX(temp)
FROM Sensors[Range 1 Hour]
GROUP BY sensID;
```

Представената заявка може да е гореща, в който случай се изпълнява само веднъж върху прозореца, който трябва вече да съществува в момента на подаване на заявката. Разбира се, че на процесора на заявки в СУПД трябва да е подаден прозореца върху Sensors поне преди един час спрямо момента на подаване на заявката. СУПД трябва да е събрала достатъчно информация за да може да отговори на заявката. СУПД може да отговоря по всяко време на тази заявка стига

да изхвърля кортежите, за които е регистрирана нова по-висока температура за същия сензор. Горната заявка може да бъде и стояща, в който случай, текущата резултатна релация може да бъде разглеждана като материализирана гледка, която се променя с времето. Има и алтернативен начин на представяне резултата от стоящата заявка.

Релациите на прозорците могат да бъдат комбинирани с други релации на прозорци или с други обикновени релации, т.е. такива, които не идват от потоците. В следния пример има такова комбиниране. Нека на вход в СУПД е потока Sensors, но също така в работната памет се поддържа обикновена релация Calibrate(sensID, multm add), която задава член за умножение и член за добавяне, които се използват за коригиране на прочетеното от сензорите. Заявката:

```
SELECT MAX(mult * temp + add)
FROM Sensors[Range 1 Hour], Calibrate
WHERE Sensors.sensID = Calibrate.sensID;
```

намира най-високата калибрирана температура прочетена, от който и да била сензор за последния час. Тук извършихме съединение на релацията на прозореца Sensors с обикновената релация Calibrate.

Също така можем да съединяваме и само релации на прозорци. Следващата заявка илюстрира автосъединение чрез подзаявка, но всички SQL средства за изразяване на съединение може да бъдат използвани. Нека да искаме да видим за всеки сензор неговата максимална температура за последния час, а също така кортежите на резултата да има информация кой е последния момент, в който температурата е регистрирана. Заявката е:

```
SELECT sensID, s.temp, s.time
FROM Sensors[Range 1 Hour] s
WHERE NOT EXISTS (
      SELECT *
      FROM Sensors[Range 1 Hour]
      WHERE sensID = s.sensID AND
      (temp > s.temp OR (temp = s.temp AND time > s.time))
);
```

Тук подзаявката проверява дали няма друг кортеж в релацията на прозореца Sensors[Range 1 Hour], които се отнася за същия сензор като този на кортежа s и който или има по-висока регистрирана температура или има същата температура, но е от по-късно време. Ако няма такива кортежи, тогава кортежа s е част от резултата.

ПРЕОБРАЗУВАНЕ НА РЕЛАЦИИТЕ В ПОТОЦИ

Стоящите заявки често променят резултатните си релации. Поддръжката на последните като материализирани гледки може да доведе до много усилия по вмъкване и изтриване на кортежи, които никой няма да ги гледа. Алтернативата е да се преобразува релацията, която е резултат от заявката обратно в поток, който може да бъде обработван като всеки друг поток. Например, можем пуснем гореща заявка в даден момент за да получим стойността на резултата от заявката, когато се интересуваме от нея.

Ако R е релация, дефинираме Istream(R) като поток от вмъкваните кортежи в R. Един кортеж се появява в потока, когато стане вмъкването му в R. Аналогично, дефинираме Dstream(R) като поток на изтритите кортежи в R. Един кортеж се появява в този поток, когато бъде изтрит от R. Обновяването на кортеж може да се представи като едновременното му изтриване и вмъкване.

Нека R е конструирана със заявката от предния пример, т.е. това е релацията, която има максималната температура за последния час за всеки сензор, заедно с последното време, в което е регистрирана температурата (вмъкнат кортежа). Тогава Istream(R) ще съдържа кортеж за всяко събитие, при което е вмъкван нов кортеж в R. Забележете, че има две събития, които добавят кортежи към R: 1) Пристига кортеж за Sensors с температура, която е поне толкова висока, колкото има произволен

текущ кортеж в R със същия идентификатор (ID) на сензора. Този кортеж се включва в R и става заедно с това елемент на потока Istream(R). 2) Текущата максимална температура за сензора i е записана преди един час и има поне един кортеж за сензора i в потока Sensors през последния час. В този случай, новият кортеж за R и за Istream(R) е кортежа на Sensors за сензора i, който е пристигнал през последния час, ако няма друг кортеж за i пристигнал през последния час и който има по-висока температура и има същата температура и е от по-късно време.

Същите две събития могат да генерират кортежи и за потока Dstream(R). В 1., ако има друг кортеж в R за същия сензор, тогава този кортеж се изтрива от R и става елемент на Dstream(R). В 2., старият един час кортеж на R за сензора i се изтрива от R и става елемент на Dstream(R). Ако пресмятаме Istream и Dstream за релация като тази конструирана със заявката от примера, тогава не е необходимо да поддържаме релацията като материализирана гледка. Вместо това можем да пускаме заявки към нейните Istream и Dstream за да получаваме отговорите при необходимост. Нека да формираме Istream I и Dstream D за релацията R от примера. Когато искаме можем да пуснем заявки към тези потоци. Например, искаме да видим максималната температура записана за сензор 100 от последния час. Това ще бъде температурата в кортежа на I за сензор 100, който има time от последния час и не е изтрит от D, т.е. не е в D ограничен до изминалия час. Тази заявка може да бъде написана както следва:

```
(SELECT * FROM I [Range 1 Hour] WHERE sensID = 100 AND time >= [Now – 1 Hour])
EXCEPT
(SELECT * FROM D[Range 1 Hour] WHERE sensed = 100);
```

В тази заявка ключовата дума Now означава текущото време. Забележете, че трябва да проверим, дали кортежът е пристигнал през последния час и дали е маркиран в последния час. Тези условия не са едни и същи, възможно е кортеж на I да е пристигнал в последния час, но да е станал с максимална температура t за сензор 100 преди тридесет минути. Същата температура може да има кортеж с time от преди осемдесет минути. Причината е, че по-висока температура от t е записана от сензор 100 преди деветдесет минути. Едва преди 30 минути t става най-високата температура за сензор 100 през последните 60 минути.

ЗАКЛЮЧЕНИЕ

Представеният модел на потоците от данни е залегнал в основата на реализациите на ред СУБД. Той е основата за провеждане на изследователска дейност в тази област, тъй като още ред въпроси свързани с потоците от данни остават нерешени, например разпределената обработка, в частност в Грид среда.

Изследванията в това направление са финансирани от Фонд научни изследвания към Софийския университет по договор 200/15.05.2009 „Развитие на грид инфраструктура за научни изследвания и обучение”.

ЛИТЕРАТУРА

[1] Babu, Sh. & W. Jennifer, Continuous Queries over Data Streams. SIGMOD Record, 2001, 30 (3).
[2] Stanford Stream Data Manager, http://infolab.stanford.edu/stream.

За контакти:
Доц. д-р Владимир Димитров, Катедра “Компютърна информатика”, Факултет по математика и информатика, Софийски университет “Св. Климент Охридски”, тел.: 02- 8161 594, e-mail: cht@fmi.uni-sofia.bg

Докладът е рецензиран.