دانشگاه صنعتی اصفهان

دانشکده برق و کامپیوتر

مدیریت پنجره‌های لغزان در سیستم‌های پردازش جریان داده

پایان‌نامه کارشناسی ارشد معماری کامپیوتر

مهدی ریاحی

استاد راهنما

دکتر محمد داورپناه جزی

۱۳۸۷

دانشگاه صنعتی اصفهان

دانشکده برق و کامپیوتر

پایان‌نامه کارشناسی ارشد مهندسی کامپیوتر – معماری کامپیوتر آقای مهدی ریاحی

تحت عنوان

مدیریت پنجره‌های لغزان در سیستم‌های پردازش جریان داده

در تاریخ ۱۳۸۶/۲/۱ توسط کمیته زیر مورد بررسی و تصویب نهائی قرار گرفت .

۱- استاد راهنمای پایان‌نامه	دکتر محمد داورپناه جزی
۲- استاد مشاور پایان‌نامه	دکتر محمدحسین سرایی
۳- استاد داور	دکتر شادرخ سماوی
۴- استاد داور	دکتر کیارش بازرگان
سرپرست تحصیلات تکمیلی دانشکده	دکتر علی‌محمد دوست‌حسینی

حمد و سپاس بی‌پایان از آنّ خالق یکتاست که اندک داشته و اندوخته ناچیز خود را مرهون لطف و مرحمت بی‌کران او می‌باشم.

از دست و زبان که برآید کز عهده شکرش به در آید

بی‌شک گذراندن این دوره بدون پشتیبانی و همراهی اساتید، دوستان و خانواده عزیزم امکان‌پذیر نبود. از زحمات بی‌دریغشان صمیمانه تشکر کنم و از خداوند متعال موفقیت و بهروزی این عزیزان را خواستارم.

از استاد ارجمند و بزرگوار، جناب آقای دکتر **محمد داورپناه جزی**، استاد راهنمای پایان نامه، که در طی سال‌های اخیر افتخار شاگردی ایشان را داشته و همواره از رهنمودهای ارزشمندشان بهره‌مند گردیده‌ام، تشکر و قدردانی می‌نمایم.

ازاستاد مشاور پایان نامه، جناب آقای دکتر **محمدحسین سرایی** ، که بسیار صمیمانه و با صبر و حوصله فراوان تجربیات ارزشمند خود را در اختیار اینجانب قرار دادند تشکر می‌نمایم.

از اساتید محترم، جناب آقای دکتر سماوی و جناب آقای دکتر بازرگان که زحمت داوری این پایان‌نامه را بر عهده گرفتند و با راهنمایی‌های خود کمک ارزشمندی در تکمیل و بهبود این پایان نامه داشتند تشکر فراوان دارم.

همچنین از جناب آقای دکتر دوست حسینی، سرپرست محترم تحصیلات تکمیلی دانشکده و سرکار خانم نکویی به‌خاطر زحمات بی‌شائبه ایشان در طی این دوره کمال تشکر و قدردانی را به عمل می‌آورم.

برخود لازم می‌دانم از دوست عزیزم آقای علی صالحی که در تهیه این پایان‌نامه با راهنمایی‌ها و همکاری خود مرا یاری نمودند تشکر و قدردانی ویژه به عمل آورم. از کلیه دوستان عزیزی که در این دوره افتخار آشنایی و دوستی با آنها را داشتم، تشکر می‌کنم و سلامتی و موفقیت را برایشان آرزومندم.

مهدی ریاحی

بهار ۱۳۸۷

دانشگاه صنعتی اصفهان

تقدیم به

پدر و مادر عزیزم

فهرست مطالب

عنوان **صفحه**

چکیده

در چند سال اخیر نیاز به گونه‌ی جدیدی از پردازش داده‌ها بوجود آمده است که با پردازش داده‌هایی که توسط پایگاه‌های داده صورت می‌گیرد متفاوت می‌باشد. این نوع پردازش داده که بر روی جریانی از داده‌ها انجام می‌گیرد، پردازش داده‌های جریانی یا پردازش جریان داده نامیده می‌شود. جریان داده، یک توالی معمولاً بی‌کران از داده‌هایی است که توسط یک منبع تولید می‌شوند. درخواست‌هایی که بر روی جریان‌های داده تعریف می‌شوند، درخواست‌های پیوسته نامیده می‌شوند زیرا به صورت پی‌درپی باید توسط سیستم پردازش جریان داده اجرا شوند. با توجه به ویژگی بی‌کران بودن داده‌های جریانی و عدم امکان ذخیره‌ی تمامی داده‌های یک منبع جریان، نمی‌توان به درخواست‌هایی که باید بر روی تمام این داده‌ها انجام شوند، پاسخگویی کرد. به همین علت، اغلب بازه‌های محدودی به نام پنجره، بر روی داده‌ها تعریف می‌شود و درخواست‌ها بر روی آنها اجرا می‌شوند. پنجره‌های لغزان که معمول‌ترین نوع پنجره‌ها می‌باشند، با رسیدن داده‌های جدید و یا پس از سپری شدن مدت زمان خاصی، به جلو لغزانده شده و باعث اجرای درخواست می‌شوند. در این تحقیق روش‌هایی برای مدیریت انواع پنجره‌های لغزان ارائه شده است که بتوانند به صورت مؤثر در سیستم‌های پردازش جریان داده‌ای که نرخ داده‌ی بالایی دارند و تعداد زیادی کاربر بر روی این داده‌ها درخواست صادر کرده‌اند، مورد استفاده قرار گیرند.

شبکه‌های حسگر، یکی از اصلی‌ترین منابع تولید داده‌های جریانی می‌باشند که امروزه کاربردهای زیادی پیدا کرده‌اند. با رشد روزافزون استفاده از شبکه‌های حسگر، یکپارچه‌سازی داده‌های شبکه‌های ناهمگون و پردازش توزیع شده‌ی درخواست‌ها، از موضوعات مهم این زمینه می‌باشند و به همین علت، وجود سیستم‌هایی که مدیریت و یکپارچه‌سازی پویای شبکه‌های حسگر را فراهم کنند، ضروری می‌باشد. GSN یکی از این سیستم‌هاست و با هدف ایجاد امکان جمع‌آوری و پردازش داده‌های شبکه‌های حسگر مختلف طراحی شده است. در این تحقیق یک زیرسیستم مدیریت پنجره‌های لغزان با استفاده از روش‌های ارائه شده نیز برای GSN طراحی و پیاده‌سازی شده است. برای ذخیره‌سازی داده‌های جریانی و انجام درخواست‌های پیوسته بر روی آنها از پایگاه داده‌های رایج استفاده شده است و بسیاری از نیازمندی‌های یک سیستم پردازش جریان داده با استفاده از این پایگاه‌های داده فراهم شده‌اند.

فصل اول
مقدمه

۱-۱ مقدمه

امروزه پردازش داده‌ها یکی از مهم‌ترین بخش‌های صنعت فناوری اطلاعات می‌باشد که پایگاه‌های داده نقش اصلی را در این پردازش بر عهده دارند. با این وجود، در چند سال اخیر نیاز به گونه‌ی جدیدی از پردازش داده‌ها بوجود آمده است که با پردازش داده‌های رایج که توسط پایگاه داده‌ها صورت می‌گیرد متفاوت می‌باشد. این نوع پردازش داده که بر روی جریانی از داده‌ها انجام می‌شود، پردازش داده‌های جریانی یا پردازش جریان داده نامیده می‌شود. یک جریان داده یک توالی از داده‌هایی است که توسط یک منبع تولید می‌شوند. جریان‌های داده چند ویژگی دارند که باعث می‌شوند پایگاه داده‌های معمولی نتوانند برای پردازش آنها استفاده شوند، از جمله اینکه جریان‌های داده معمولاً بی‌کران بوده و به صورت پی‌درپی داده‌های جدید توسط منبع جریان تولید می‌شوند. به خاطر این طبیعت بی‌کران بودن داده‌های جریانی ذخیره‌سازی آنها در پایگاه‌های داده به شکلی که برای داده‌های معمولی رایج است، امکان‌پذیر نمی‌باشد. همچنین درخواست‌هایی که بر روی این جریان‌های داده تعریف می‌شوند با درخواست‌هایی که در پایگاه داده‌های معمولی بر روی داده‌ها صادر می‌شوند متفاوت می‌باشند. درخواست‌هایی که بر روی جریان‌های داده صادر می‌شوند درخواست‌های پیوسته[1] نامیده می‌شوند، زیرا به صورت پی‌درپی باید توسط سیستم پردازش جریان داده اجرا شوند. در مورد ویژگی‌های سیستم‌های پردازش جریان‌های داده مطالب بسیار زیادی وجود دارد که در فصل دوم بیشتر به آن پرداخته شده است.

یکی از مهم‌ترین منابع داده‌های جریانی، شبکه‌های حسگر می‌باشند که در طی چند سال اخیر استفاده از آنها برای کاربردهای تحقیقاتی و صنعتی بسیار گسترش پیدا کرده است. یکی از اصلی‌ترین دلایل افزایش استفاده از شبکه‌های حسگر، ارزان‌تر و کوچک‌تر شدن انواع حسگرهای کم‌مصرف می‌باشد. به لحاظ ظرفیت و توان پایین

[1] Continuous query

پردازش در این حسگرها، وجود نرم‌افزارهای پردازش داده‌ای که بر روی کامپیوترهای قوی‌تر کار می‌کنند الزامی می‌باشد. این نرم‌افزارها را که باید به عنوان یک سیستم پردازش جریان داده عمل کنند میان‌افزار[1] می‌نامند. در فصل دوم به صورت مفصل‌تری به این موضوع پرداخته شده است.

با توجه به ویژگی بی‌کران بودن داده‌های جریانی و عدم امکان ذخیره‌ی تمامی داده‌های یک منبع جریان، نمی‌توان به درخواست‌های کاربر که باید بر روی تمام این داده‌ها انجام شوند پاسخگویی کرد. به همین علت اغلب یک بازه‌ی محدود بر روی داده‌ها تعریف می‌شود و درخواست‌ها بر روی آنها صادر می‌شوند. چنین بازه‌هایی به نام پنجره نیز شناخته می‌شوند. چون درخواست‌هایی که بر روی جریان‌های داده صادر می‌شوند اغلب درخواست‌های پیوسته هستند، باید سازوکاری برای مدیریت این پنجره‌ها وجود داشته باشدکه هر بار هنگام اجرای درخواست، داده‌های جدید در این پنجره وجود داشته باشند. برای رسیدن به این هدف اغلب از پنجره‌های لغزان[2] استفاده می‌شود. به این ترتیب که بر اساس نوع بازه‌ی پنجره، دو محدوده‌ی آن در زمان‌های خاصی بر روی داده‌ها به سمت جلو حرکت داده می‌شوند. هدف اصلی این تحقیق ارائه‌ی روش‌هایی برای پیاده‌سازی پنجره‌های لغزان است به طوری که این روش‌ها به حد کافی عمومی باشند و کارایی قابل قبولی به ویژه برای سیستم‌های پردازش جریان داده‌ای که تعداد زیادی کاربر دارند و نرخ داده‌ها در آنها بالاست، داشته باشند.

۱-۲ روند مطالب پایان‌نامه

در فصل دوم مفاهیم اصلی در پردازش جریان داده به طور خلاصه جمع‌آوری شده‌اند. تفاوت‌های پایگاه داده‌های معمولی با سیستم‌های پردازش جریان داده همراه با مثال بیان شده است و دلایل عدم کاربرد سیستم‌های مدیریت پایگاه داده برای پردازش جریان‌های داده ذکر شده‌اند. خصوصیات لازم برای یک سیستم پردازش جریان داده با توجه به ویژگی‌های جریان‌های داده‌ای و درخواست‌هایی که بر روی آنها صادر می‌شوند مورد بحث قرار گرفته‌اند. به چند نمونه از سیستم‌های پردازش جریان داده‌ی موجود به صورت خلاصه اشاره شده است و ویژگی‌های سه مورد از مهم‌ترین این سیستم‌ها، از نظر تحقیقاتی، به شکل مفصل‌تری بیان شده است. در بخش دوم از این فصل به بیان ویژگی‌های شبکه‌های حسگری و کاربردهای آنها پرداخته شده است و مروری بر مهم‌ترین فعالیت‌های تحقیقاتی انجام شده دو مورد کاربرد شبکه‌های حسگری و چگونگی جمع‌آوری داده‌های آنها و پردازش این داده‌ها صورت گرفته است.

در این تحقیق علاوه بر ارائه‌ی روش‌های مدیریت پنجره‌های لغزان، پیاده‌سازی آنها نیز بر روی سیستم GSN[2] انجام شده است. GSN یک پروژه‌ی متن‌باز است که به عنوان یک میان‌افزار برای جمع‌آوری داده‌های شبکه‌های حسگر مختلف و پردازش آنها عمل می‌کند. اگرچه هدف اصلی GSN شبکه‌های حسگر است اما با توجه به قابلیت‌هایی که فراهم کرده است می‌توان آن را برای پردازش هر نوع جریان داده‌ای به‌کار برد. فصل سوم به معرفی GSN اختصاص دارد. در فصل سوم ویژگی‌های متمایز GSN، مفاهیم اصلی تعریف شده در GSN، چگونگی پیکربندی، چگونگی تعریف جریان‌های داده و ارتباط با منابع تولید کننده‌ی داده، نحوه‌ی تعریف درخواست‌های

---

[1] Middleware
[2] Global Sensor Networks (http://gsn.sourceforge.net)

[2] Sliding window

پیوسته، معماری GSN و بسیاری مسائل دیگر مطرح شده است. درنهایت مسأله‌ای که پایه‌ی اصلی این تحقیق را تشکیل می‌دهد به صورت کامل‌تر شرح داده شده است.

فصل چهارم به الگوریتم‌های ارائه شده برای مدیریت پنجره‌های لغزان و چگونگی پیاده‌سازی آنها و همچنین امکاناتی که به GSN افزوده شده است اختصاص دارد. در این فصل مدلی که برای سیستم درنظر گرفته شده است شرح داده می‌شود. تقسیم‌بندی انواع پنجره‌های لغزان و تعریف گراف و درخت لغزش و چگونگی استفاده از آنها در مدیریت پنجره‌های لغزان نیز در این فصل بیان شده‌اند. الگوریتم‌های مدیریت پنجره‌های لغزان برای انواع پنجره‌های لغزان به صورت عمومی شرح داده شده‌اند و پس از آن چگونگی پیاده‌سازی پنجره‌های لغزان بر روی GSN بیان شده است. استفاده از view برای پردازش جریان‌های داده و چگونگی نگاشت آن به نیازهای پنجره‌های لغزان، نحوه‌ی تشخیص و مدیریت داده‌های منقضی شده و بدون استفاده و هماهنگ کردن زمان نمونه‌ی در حال اجرای GSN و پایگاه داده‌ی استفاده شده نیز از مطالب فصل چهارم می‌باشند. الگوریتمی نیز برای بهینه‌سازی گراف لغزش و تا حد امکان کاهش زمان لازم برای تصمیم‌گیری در مورد زمان لغزش پنجره‌ها ارائه شده است. در نهایت به واسط گرافیکی طراحی شده برای GSN و چگونگی کاربرد آن به صورت مختصر اشاره شده است.

فصل پنجم به ارزیابی الگوریتم‌های ارائه شده اختصاص دارد. بخش اول آن به ارزیابی پیاده‌سازی الگوریتم‌ها بر روی GSN و همچنین ارزیابی صحت عملکرد GSN می‌پردازد. اگرچه همه‌ی قسمت‌های پیاده‌سازی شده در GSN به صورت نرم‌افزاری توسط آزمون‌های واحد[1] آزمایش شده‌اند، برای مشاهده‌ی عملکرد سیستم از یک سری داده‌ی واقعی استفاده شده است. برای هر کدام از انواع مختلف پنجره‌های لغزان تعدادی درخواست پیوسته تعریف شده است که بر روی جریان داده‌ای از داده‌های ذخیره‌ی شده دوباره جریانی شده‌ی یک شبکه‌ی حسگری صادر شده‌اند. با نمایش تعدادی از داده‌های تولید شده توسط این درخواست‌ها، درستی الگوریتم‌های ارائه شده و همچنین پیاده‌سازی آنها مورد بررسی قرار گرفته است. بخش دوم از این فصل به ارزیابی استفاده از گراف لغزش مطرح شده پرداخته است. برای این ارزیابی از یک شبیه‌سازی استفاده شده است و نتایج آن مورد بررسی قرار گرفته‌اند. این نتایج کارایی الگوریتم‌های ارائه شده در این تحقیق را به خوبی بیان می‌کنند.

در فصل ششم خلاصه‌ای از مطالب فصل‌های مختلف به همراه نتایج کلی به‌دست آمده از آن بیان شده است و مباحثی که در آینده می‌توان در ادامه‌ی این تحقیق به آنها پرداخت به صورت پیشنهاداتی ارائه گردیده است.

[1] Unit tests

فصل دوم
مفاهیم پایه و مرور کارهای انجام شده

۲-۱ مقدمه

در این فصل ابتدا مفاهیم پایه در زمینه‌ی پردازش داده‌های جریانی بیان شده‌اند و سپس چند نمونه از سیستم‌های پردازش جریان داده معرفی شده و به صورت خلاصه به ویژگی‌های آنها پرداخته شده است. همچنین مهمترین مفاهیم در شبکه‌های حسگر که یکی از اصلی‌ترین تولید کنندگان داده‌های جریانی می‌باشند، بررسی شده‌اند و مروری بر کارهای انجام شده در این زمینه صورت گرفته است.

۲-۲ پردازش داده‌های جریانی

۲-۲-۱ کلیات

سیستم‌های مدیریت پایگاه داده (DBMS ها) از ابتدا با هدف اصلی پشتیبانی از کاربردهای تجاری ایجاد شده و توسعه پیدا کرده‌اند. در این سیستم‌ها داده‌ها در نتیجه‌ی تراکنش‌هایی که توسط کاربر ارائه شده‌اند تغییر می‌کنند. به طور مشابه دسترسی به داده‌ها در نتیجه‌ی درخواست‌های آغاز شده توسط کاربر صورت می‌پذیرد. علاوه براین، فرض شده است که DBMS ها یک مخزن منفعل[1] می‌باشند که مجموعه‌ی بزرگی از عناصر داده‌ای را ذخیره می‌کنند. این مدل را یک مدل انسان-فعال، DBMS-منفعل می‌توان نامید [۱]. DBMS های مرسوم پاسخگوی نیازمندی‌های رده‌ی جدیدی از کاربردها که نیازمند ارزیابی درخواست‌های ماندگار[2] و طولانی مدت اجرا شونده[3] بر روی جریانی از آیتم‌های داده‌ای پی‌درپی و بلادرنگ نمی‌باشند.

یک جریان داده ترتیب معمولاً بی‌کرانی از تاپل‌ها می‌باشد که در یک تقسیم‌بندی سطح بالا می‌توان دو نوع کلی برای آن درنظر گرفت [۲]. این تقسیم‌بندی در ادامه توضیح داده شده است.

---

[1] Passive
[2] Persistent
[3] Long-running
[4] Log

- *جریان‌های داده‌ای تراکنشی*: جریان‌های داده‌ای تراکنشی گزارش وقایع[۴] ارتباطات بین موجودیت‌ها می‌باشند. به عنوان مثال، دربسیاری از وب‌گاه‌ها ارتباطات کاربر با وب‌گاه ثبت شده و سپس این گزارش‌ها برای کاربردهایی مانند پایشگری[۱] کارایی و شخصی‌سازی استفاده می‌شوند. اطلاعات جدید این ارتباطات به صورت پی‌درپی در انتهای گزارش وقایع افزوده می‌شوند. فروشندگان کارت‌های اعتباری خریدهای انجام شده توسط دارندگان کارت‌ها را پایشگری می‌کنند تا ناهنجاری‌هایی را که استفاده‌ی احتمالاً متقلبانه[۲] را نشان می‌دهند کشف کنند. گزارش وقایع تراکنش‌های کارت اعتباری یک جریان داده‌ای متوالی از داده‌های تراکنشی می‌باشد.
- *جریان‌های داده‌ای مربوط به اندازه‌گیری*: این جریان‌های داده‌ای در نتیجه‌ی پایشگری وضعیت موجودیت‌های تحت نظر ایجاد می‌شوند. بارزترین مثال در این زمینه داده‌هایی است که حسگرها به عنوان مقادیر اندازه‌گیری شده تولید می‌کنند.

به عنوان یک مثال پرکاربرد دیگر شبکه‌های تبادل ارتباطات کامپیوتری بزرگ را می‌توان نام برد که برای اعمالی نظیر مکان‌یابی گلوگاه‌ها و یا تشخیص حمله به نقطه‌ای از شبکه نیازمند پایشگری پیوسته‌ی داده‌ها می‌باشند. داده‌هایی که پایشگری می‌شوند شامل سرآیندهای پکت‌هایی هستند که در شبکه رد و بدل می‌شوند. چنین داده‌هایی تشکیل یک جریان داده‌ای اغلب با نرخ بالا می‌دهند [۲].

درخواست‌های نوعی‌ای که در چنین کاربردهایی صادر می‌شوند به صورت ماندگار و اجرا شونده در طی زمان طولانی می‌باشند؛ به این معنی که اگر کاربر درخواستی را ثبت کند سیستم مدیریت داده باید به صورت مداوم و تا زمانی که آن درخواست حذف نشده است، آن را ارزیابی و یا به عبارتی اجرا کند. چنین سیستمی را یک سیستم مدیریت جریان داده[۳] (DSMS) می‌نامند. در این سیستم‌ها هنگام بروز شرایط خاصی به صورت خودکار به کاربر اطلاع داده می‌شود و همیشه سیستم به صورت فعال در حال اجراست. از این رو می‌توان آن را یک مدل DBMS–فعال، انسان–منفعل نامید [۱]. یک مثال کاربردی برای پایشگری ترافیک در مرجع [۳] مطرح شده است که به صورت خلاصه در ادامه شرح داده می‌شود. فرض کنید که حسگرهایی در بزرگراه‌ها تعبیه شده‌اند که ترافیک جاری بر روی جاده را گزارش می‌دهند. داده‌هایی که توسط یک حسگر گزارش می‌شوند تشکیل یک جریان پیوسته را می دهند که اطلاعات زیر را در بردارد:

۱. شناسه‌ی یک اتومبیل به همراه سرعت کنونی آن.
۲. شناسه‌ی بزرگراه به همراه قسمتی از بزرگراه که اتومبیل بر روی آن حرکت می‌کند.
۳. جهت حرکت اتومبیل.

درخواست‌های مختلفی بر روی این داده‌ها می‌توان انجام داد که در ادامه چند نمونه ذکر شده است.

- برای مدیریت جریان ترافیک یک درخواست می‌تواند متوسط سرعت اتومبیل‌ها را بر روی یک قسمت بزرگراه محاسبه کند. اگر سرعت متوسط به پایین‌تر از یک حد آستانه رسید می‌توان به آن دسته از اتومبیل‌هایی که قصد حرکت بر آن قسمت بزرگراه را دارند اطلاع داد تا مسیر دیگری را انتخاب نمایند.
- بر اساس متوسط سرعت بر روی یک قسمت بزرگراه می‌توان به اپراتورها اطلاع داد که احتمالاً تصادف‌هایی

[1] Monitoring [2] Fraudulent [3] Data Stream Management System

رخ داده که باعث کند شدن حرکت بر روی آن قسمت شده است.

- بر روی جاده‌های دارای عوارض[1] جریان داده‌ای که از حسگرها می‌آید می‌تواند برای کم کردن خودکار مقدار عوارض از حساب اتومبیل‌هایی که نشانه‌ی[2] آنها به وسیله‌ی حسگرها تشخیص داده شده است، استفاده شود.
- یکی از اهداف در مدیریت ترافیک پیشرفته این است که با استفاده از نرخ عوارض متغیر، ازدحام در ترافیک را مدیریت کرد. برای رسیدن به این هدف، برنامه‌های کاربردی می‌توانند عوارض را به صورت تابعی از متوسط سرعت حرکت بر روی بزرگراه محاسبه کنند و از حساب اتومبیل‌ها بکاهند.

براساس مثال مطرح شده، تطبیق‌پذیری قابلیت‌های DBMS های رایج با نیازمندی‌های کاربردهای جریانی در جدول ۲-۱ بررسی شده است [۲].

جدول ۲-۱: مقایسه‌ی نیازمندی‌های کاربردهای جریانی با قابلیت‌های DBMS های متداول.

موضوع	در DBMS های متداول	در کاربردهای جریانی	مثال
نوع درخواست	کاربردهای DBMS های رایج درخواست‌های تک-دفعه‌ای[3] را صادر می‌کنند؛ به این معنی که پس از صدور درخواست، DBMS آن را ارزیابی می‌کند و نتایج را بر اساس داده‌های موجود در آن زمان به برنامه‌ی کاربردی برمی‌گرداند.	در کاربردهای جریانی لازم است پس از ثبت یک درخواست، به صورت مداوم آن درخواست ارزیابی گردد مگر اینکه درخواست از سیستم حذف شود.	در کاربرد مدیریت ترافیک جاده، لازم است متوسط سرعت اتومبیل‌ها در سیستم به صورت پیوسته و بر اساس مقادیر جدیدی که حسگرها گزارش می‌دهند، ارزیابی شود.
مشخصه‌ی زمان	در کاربردهای رایج، داده‌ها لزوماً دارای یک مشخصه‌ی زمان نیستند. به‌روز رسانی یک خصیصه مقدار قبلی آن را جای‌نویسی[4] می‌کند.	در کاربردهای جریانی مفهوم به‌روز رسانی یک داده وجود ندارد، بلکه داده‌ها به شکل ترتیبی از مقادیر در طی زمان می‌باشند. همچنین درخواست‌ها بر روی داده‌های جریانی عموماً بر روی یک تاریخچه‌ی زمانی از مقادیر صادر می‌گردند.	در کاربرد مدیریت ترافیک جاده، ممکن است متوسط سرعت اتومبیل‌ها بر روی ۵ دقیقه‌ی آخر مورد نیاز باشد.
ویژگی داده‌هایی که درخواست‌ها بر روی آنها انجام می‌شوند.	درخواست‌ها بر روی داده‌هایی که محدود هستند (اگرچه ممکن است از نظر تعداد زیاد باشند) و در هنگام ارزیابی درخواست تغییر نمی‌کنند، ارزیابی می‌شوند. اگر داده‌ها هنگام ارزیابی درخواست‌ها تغییر کنند، DBMS سازوکارهایی را استفاده می‌کند تا نتایج ارزیابی در یک لحظه‌ی زمان خاص، صحیح باشد. به عبارتی دیگر، در زمان اجرای درخواست، عملیات بر روی یک تصویر لحظه‌ای[2] از داده‌ها در آن زمان	برای درخواست‌های جریانی، مجموعه داده‌ها به صورت پیوسته در حال رشد هستند و درخواست‌ها باید بر روی چنین داده‌های بی‌کرانی ارزیابی شوند.	در مثال کاربرد مدیریت ترافیک جاده، حسگرها همیشه فعال هستند و در هر چند لحظه داده‌های جدید را ارسال می‌کنند. چنین داده‌هایی را حتی اگر برای مدت چند روز درنظر بگیریم می‌توان بی‌نهایت دانست.

[1] Toll [2] Tag [3] One-time [4] Overwrite
[2] Snapshot [2] Failure [3] Active databases
[4] Temporal databases [5] Continuous queries

	انجام می‌گیرد.		
دقت داده‌ها	در DBMS های متداول فرض می‌شود که داده‌ها دقیق هستند.	در کاربردهای جریانی سیستم مدیریت داده‌ها باید امکان مدیریت داده‌های نامطمئن را داشته باشد. علت این عدم اطمینان در داده‌ها می‌تواند شکست[۲] و یا خطای حسگرها باشد، همچنین تأخیرهایی که در شبکه‌ی انتقال داده‌ها وجود دارد باعث برهم خوردن ترتیب داده‌ها می‌شود.	در سیستم مدیریت ترافیک جاده، این امکان وجود دارد که حسگرها در تشخیص سرعت‌ها دچار خطا شده و تعدادی داده‌ی خطادار گزارش کنند. به هم خوردن ترتیب داده‌ها نیز در این سیستم می‌تواند پیش‌آید.
فعال بودن و یا منفعل بودن سیستم	DBMS های متداول اغلب منفعل می‌باشند. کاربر یک تراکنش را آغاز می‌کند، DBMS آن را اجرا می‌کند و سپس نتایج را به او برمی‌گرداند.	بسیاری از کاربردهای جریانی طبیعت پایشگری دارند؛ به این معنی که این کاربردها وضعیت موجودیت‌هایی را پیوسته زیرنظر می‌گیرند و در صورت بروز شرایط خاص واکنش نشان می‌دهند. از کاربردی که یک واحد تولید مواد شیمیایی را پایشگری می‌کند ممکن است خواسته شود تا در صورتی که مقدار اندازه‌گیری شده توسط یک حسگر (مثلاً اندازه‌ی دما) بسیار زیاد شود یا اگر یک حسگر مقداری را به تعداد دوبار در آخرین ۲۴ ساعت خارج از محدوده‌ی مشخصی ضبط کرده باشد، به اپراتور هشدار دهد [۱]. کاربردها می‌توانند چندین جریان را پایشگری کنند و در نتیجه هشدار دهی را در صورت برقراری شرایط پیچیده انجام دهند.	در مثال مدیریت ترافیک اگر متوسط سرعت بر روی یک قسمت از بزرگراه کمتر از حد آستانه‌ای شود باید به صورت خودکار به اتومبیل‌هایی که قصد حرکت بر روی آن قسمت دارند اطلاع داده شود و مسیرهای جایگزین به آنها پیشنهاد گردد.

اگرچه تحقیقاتی در مورد پایگاه داده‌های فعال[۳] [۴] و پایگاه داده‌های زمانی[۴] [۵] برای افزودن برخی از قابلیت‌های ذکر شده به DBMS های متداول صورت گرفته است، ولی به دلیل عدم قابلیت گسترش مناسب برای رفع نیازهای کاربردهای جریانی استفاده‌ی عملی چندانی ندارند. چالش‌هایی که کاربردهای جریانی بوجود می‌آورند باعث می‌شود که مجموعه‌ای از خواسته‌های عملیاتی بر روی سیستم‌های مدیریت داده‌های جریانی گذاشته شود [۱، ۲، ۶]. مهمترین این خواسته‌ها در ادامه ارائه شده‌اند.

۱. **مفهوم زمان و درخواست‌های پیوسته**: مدل داده‌ای باید مفهوم زمان را دربر بگیرد و همچنین سازوکار درخواست باید از تعریف درخواست‌های پیوسته[۵] بر روی چنین داده‌هایی پشتیبانی کند.
۲. **تولید نتایج درخواست‌ها از روی داده‌های ناکامل**: به خاطر طبیعت بی‌کرانی داده‌های جریانی و همچنین امکان نامطمئن بودن داده‌ها، یک سیستم مدیریت داده‌های جریانی باید دارای قابلیت محاسبه‌ی جواب‌ها بر اساس داده‌های ناکامل و نامطمئن باشد.
۳. **پشتیبانی از عملگرهایی که به کل داده‌ها نیاز دارند**: برخی از عملگرها در DBMS ها، مانند عملگر *Sort* و توابع

جمعی (*AVG*، *MAX* و ...) و *Join*، نیاز به وجود تمام داده‌ها دارند. این عملگرها را عملگرهای مسدود کننده[1] می‌نامند. واضح است که به دلیل بی‌کران بودن داده‌های جریانی، سیستم مدیریت جریان نمی‌تواند از این عملگرهای مسدود کننده که نیازمند پردازش کل جریان قبل از تولید جواب می‌باشند، استفاده کند. بنابراین، سیستم مدیریت جریان باید راه حلی برای این مشکل ارائه دهد.

۴. **ارزیابی درخواست‌ها تنها با یک گذر بر روی داده‌ها**: به خاطر محدودیت‌های ذخیره‌سازی و نیز کارایی سیستم، بازگشت به عقب بر روی یک جریان داده (مگر در صورت وجود پنجره بر روی جریان) امکان‌پذیر نمی‌باشد. الگوریتم‌های جریانی برخط مجبورند تنها یک گذر بر روی داده‌ها داشته باشند.

۵. **ایجاد طرح‌های درخواست به صورت وفقی**[2]: از آنجا که درخواستها به صورت طولانی مدت و ماندگار هستند، سیستم مدیریت جریان باید توانایی تطبیق با تغییرات شرایط را داشته باشد. بر این اساس، طرح‌های اجرای درخواست[3] باید برخلاف طرح‌های ایستای متداول به صورت وفقی و پویا باشند.

۶. **دارا بودن قابلیت واکنشی**[4] **گسترش‌پذیر**: سیستم‌های مدیریت جریان باید دارای قابلیت واکنشی باشند. مسأله‌ای که در اینجا مطرح است این است که سیستم باید بتواند قابلیت گسترش برای هزاران راه‌انداز[5] را داشته باشد. حتی در آن دسته از DBMS های مرسوم که مفهوم راه‌انداز در آنها وجود دارد چنین قابلیت گسترشی وجود ندارد.

۷. **مدیریت منابع برای تضمین کیفیت سرویس**: علاوه بر وجود قابلیت واکنشی، در بسیاری از کاربردهای جریانی فراهم کردن این قابلیت به صورت بلادرنگ (یا تقریباً بلادرنگ) نیز ضروری می‌باشد. برای برقراری این قابلیت، سیستم باید دارای یک مدیریت هوشمند منابع و نیز راهبردهای تنزل[6] مؤثر باشد به طوری که بتواند تضمین‌هایی برای کیفیت سرویس (QoS) فراهم کند.

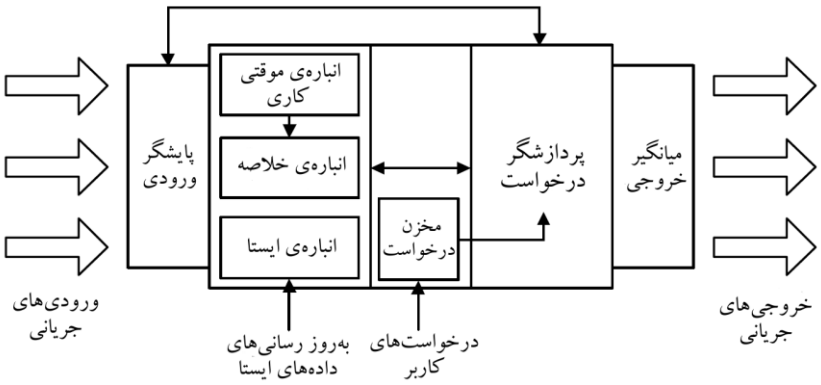
۸. **درنظر داشتن محدودیت منابع**: بسیاری از کاربردهای جریانی به صورت ذاتی توزیع شده هستند. به عنوان مثال حسگرهای به‌کار رفته برای پایشگری یک زیستگاه ممکن است در ناحیه‌ی پهناوری توزیع شوند. توزیعی بودن سبب می‌شود که ملاحظات پهنای بند در پردازش درخواست‌ها اهمیت پیدا کند. به‌علاوه، معمولاً حسگرها دارای باطری‌ها یا منابع تغذیه‌ی با طول عمر محدود هستند و در پردازش درخواست‌ها باید استفاده‌ی مفید و مؤثر از توان باطری‌ها درنظر گرفته شود.

سیستم‌های مدیریت جریان داده به صورت کلی از معماری نشان داده در شکل ۲-۱ استفاده می‌کنند [۶]. پایشگر ورودی داده‌های جریانی را از منابع دریافت می‌کند و ممکن است بر حسب نیاز نرخ داده‌ها را تنظیم کند (مثلاً با دور انداختن برخی از داده‌ها). داده‌ها به طور معمول در سه بخش شامل انباره‌ی موقتی کاری (به عنوان مثال برای درخواست‌های پنجره‌ای)، انباره‌ی خلاصه[2] (برای synopsis ها) و انباره‌ی ایستا (برای متاداده‌هایی مانند مکان فیزیکی هر منبع) ذخیره می‌شوند. درخواست‌های پیوسته‌ای که از طرف کاربران صادر می‌شوند در مخزن درخواست ثبت می‌گردند و ممکن است برای اجرا به صورت اشتراکی گروه‌بندی شوند. ممکن است امکان پذیرش

---

[1] Blocking operators
[2] Adaptive
[3] Query execution plan
[4] Reactive
[5] Trigger
[6] Degradation
[2] Synopsis
[2] Re-optimize
[3] Query languages
[4] Transient
[5] Timestamp
[6] Declarative language

درخواست‌های تک-دفعه‌ای برای وضعیت کنونی داده‌ها نیز در سیستم فراهم گردد. پردازشگر درخواست، درخواست‌ها را از مخزن درخواست گرفته و با استفاده از داده‌های ذخیره شده آنها را اجرا می‌کند. پرداشگر درخواست می‌تواند با پایشگر ورودی نیز ارتباط داشته و بر اساس تغییرات ورودی و در صورت لزوم، طرح‌های درخواست را باز-بهینه[۲] سازد. نتایج یا برای کاربران به صورت جریانی ارسال می‌شوند و یا به صورت موقت در میانگیر خروجی ذخیره می‌شوند.



شکل ۲-۱: معماری تجریدی مرجع برای یک سیستم مدیریت جریان داده [۶].

۲-۲-۲ مدل‌های داده‌ی جریانی و زبان‌های درخواست[۳]

یک جریان بلادرنگ داده، یک توالی از آیتم‌های داده می‌باشد که در ترتیب مشخصی دریافت می‌شوند. هر آیتم از جریان داده می‌تواند به صورت یک تاپل رابطه‌ای یا یک نمونه‌ی شیء مدل‌سازی شود. در اغلب سیستم‌هایی که بر اساس مدل رابطه‌ای طراحی شده‌اند آیتم‌های داده به صورت یک تاپل گذرای[۴] دارای مَهر زمانی[۵] یا یک شماره‌ی ترتیب مدل می‌شوند. در چنین سیستم‌هایی از یک زبان اعلانی[۶] SQL-مانند برای درخواست دادن بر روی جریان داده‌ها استفاده می‌شود. به طور کلی دو نوع مهر زمانی برای داده‌ها می‌توان در نظر گرفت [۲]:

۱ -*مهر زمانی ضمنی* که توسط سیستم مدیریت جریان هنگام دریافت هر تاپل بر روی آن قرار داده می‌شود. این مهر زمانی، اغلب زمان سیستم در لحظه‌ی ورود تاپل می‌باشد. ممکن است بسته به مورد استفاده‌ی سیستم، از یک شماره‌ی ترتیب نیز به عنوان مهر زمانی تاپل بتوان استفاده کرد.

۲ -*مهر زمانی صریح* که یک خصوصیت از خود داده می‌باشد و توسط تولید کننده‌ی داده بر روی آن قرار داده می‌شود.

در بسیاری از موارد تنها بخشی از هر جریان داده در هر لحظه مورد نیاز می‌باشد. در این موارد از مفهوم پنجره بر روی جریان داده استفاده می‌شود. بر اساس سه معیار زیر می‌توان انواع پنجره‌ها را رده‌بندی کرد [۶].

۱ -**جهت حرکت نقاط انتهایی**: دو نقطه‌ی ثابت، یک پنجره‌ی ثابت و دو نقطه‌ی لغزان، یک پنجره‌ی لغزان[۱] را

---

[1] Sliding window
[2] Eager
[3] Lazy
[4] Procedural languages
[5] Intrusion detection

تعریف می‌کنند. اگر یک نقطه ثابت و نقطه‌ی دیگر لغزان باشد، یک پنجره‌ی *landmark* تعریف می‌شود.

۲ - **پنجره‌ی فیزیکی در مقابل پنجره‌ی منطقی**: پنجره‌های فیزیکی یا زمان-مبنا بر اساس یک بازه‌ی زمانی تعریف می‌شوند، درحالی‌که پنجره‌های منطقی یا تعداد-مبنا (و یا تاپل-مبنا) بر حسب تعداد تاپل‌ها تعریف می‌شوند. این نحوه‌ی بیان کردن پنجره‌ها بسطی از مفهوم SQL-99 برای بیان کردن پنجره‌های فیزیکی یا منطقی بر روی رابطه‌ها می‌باشد.

۳ - **بازه‌ی به‌روز رسانی**: پنجره‌ها می‌توانند به صورت مشتاق[۲] با رسیدن هر تاپل و یا به صورت تنبل[۳] (یا پردازش دسته‌ای) به‌روز شوند. اگر بازه‌ی به‌روز رسانی بزرگتر از اندازه‌ی پنجره باشد، پنجره را *tumbling window* گوییم.

به طور کلی سه نوع زبان برای درخواست‌های پیوسته پیشنهاد شده است. دسته‌ی اول زبان‌های رابطه-مبنا مانند CQL [۷]، AQuery [۸] و StreaQuel [۹] می‌باشند. اغلب این زبان‌ها بر اساس SQL بوده و قابلیت‌هایی برای پنجره‌ها و مدیریت ترتیب و زمان به نحو آن افزوده‌اند. دسته‌ی دوم زبان‌های شیء-مبنا می‌باشند که در سیستم پایشگری شبکه‌ی Tribeca [۱۰] و پایگاه داده‌ی حسگر Cougar [۱۱] استفاده شده‌اند. دسته‌ی سوم زبان‌های رویه‌ای[۴] هستند که در آنها کاربر باید جریان داده و عملیات را مشخص کند. سیستم Aurora [۱] از این نوع زبان استفاده می‌کند.

۲-۲-۳ سیستم‌های مدیریت داده‌های جریانی نمونه

در سال‌های اخیر تعداد قابل توجهی سیستم جریانی در مراکز علمی و تحقیقاتی تولید شده و یا در حال تکمیل است. به برخی از این سیستم‌ها در ادامه اشاره شده است.

۱. **Hancock** [۱۲]: سیستم Hancock در آزمایشگاه‌های AT&T ساخته شده است. Hancock یک زبان شبه C فراهم کرده است که توسط آن می‌توان درخواست‌هایی را بر روی داده‌های تراکنشی انجام داد.

۲. **Gigascope** [۱۳]: این سیستم نیز در آزمایشگاه‌های AT&T ساخته شده است. Gigascope یک زبان SQL مانند به نام GSQL فراهم کرده و برای کاربردهای شبکه‌ای، مانند تحلیل ترافیک، کشف نفوذ[۵]، تحلیل پیکربندی مسیریاب، پایشگری شبکه و غیره طراحی شده است.

۳. **NiagaraCQ** [۱۴]: این سیستم در دانشگاه Wisconsin-Madison برای پردازش درخواست‌های پیوسته بر روی مجموعه‌های پویای داده بر روی وب‌گاه‌ها طراحی شده است. مدل داده‌ای و زبان درخواست در NiagaraCQ براساس XML و XML-QL می‌باشد.

۴. **Tapestry** [۱۵]: درخواست‌های پیوسته در سیستم Tapestry به منظور فیلتر کردن براساس محتوا بر روی پایگاه داده‌های فقط-افزودنی[۱] ایمیل‌ها و پیغام‌های تابلوی اعلانات[۲] به‌کار رفته‌اند. یک زیرمجموعه‌ی محدود شده از SQL برای تضمین ازریابی کارآمد درخواست‌ها و نیز فراهم کردن نتایج درخواست فقط-افزودنی استفاده شده است.

۵. **Aurora** [۱]: این سیستم با همکاری دانشگاه‌های Brandeis، Brown و MIT ایجاد شده و توسعه پیدا کرده

---

[1] Append-only
[2] Bulletin board
[3] STanford stREam data Manager
[4] Utility

است. کاربر درخواست‌های خود را به صورت گرافیکی می‌تواند بیان کند و هر درخواست مجموعه‌ای از تعدادی جعبه و اتصالات آنها می‌باشد.

۶. **TelegraphCQ** [۹]: این سیستم که در دانشگاه Berkeley طراحی شده و توسعه پیدا کرده است، از پایگاه داده‌ی متن‌باز PostgreSQL برای پیاده‌سازی استفاده کرده و قابلیت پردازش درخواست‌های پیوسته را به آن افزوده است.

۷. **STREAM**[3] [۷، ۱۶]: پروژه‌ی STREAM در دانشگاه Stanford توسعه پیدا کرده است و بر اساس سه نوع عملگر که تبدیلات بین جریان و رابطه را انجام می‌دهند زبان درخواست خود CQL را ایجاد کرده است.

به دلیل اینکه سه سیستم Aurora، TelegraphCQ و STREAM نسبت به سایر سیستم‌های ذکر شده جامع‌تر بوده و تحقیقات منتشر شده‌ی بیشتری بر روی آنها صورت گرفته است، در ادامه مهمترین ویژگی‌های این سه سیستم به صورت مختصر بیان می‌شوند.

سیستم Aurora

در Aurora کاربر درخواست‌های خود را به صورت گرافیکی توسط یک سری جعبه و اتصالات آنها بیان می‌کند. جعبه یک عنصر گرافیکی است که نماینده‌ی یک عملگر بر روی جریان داده می‌باشد و پیکان‌ها جهت جریان داده‌ها را تعیین کرده و اتصال خروجی‌های یک جعبه را به عنوان ورودی‌های جعبه‌ی دیگر بر عهده دارند. Aurora درخواست‌های پیوسته، دیدها و درخواست‌های ad-hoc را پشتیبانی می‌کند. یک گراف QoS دو بعدی توسط کاربر به هر خروجی نسبت داده می‌شود که میزان سودمندی[4] خروجی را برحسب تعدادی ویژگی مربوط به کارایی و کیفیت مشخص می‌کند. شکل ۲-۲ مدل درخواست استفاده شده در Aurora را نمایش می‌دهد. در قسمت بالای این شکل یک درخواست پیوسته مشاهده می‌شود. در شبکه‌هایی که درخواست‌های پیوسته را تشکیل می‌دهند، داده‌ها پس از پردازش نهایی دور ریخته می‌شوند. نقاطی که به شکل دایره‌ی توپر نشان داده شده‌اند، نقاط اتصال نامیده می‌شوند و تغییرات پویا را در شبکه امکان‌پذیر می‌سازند. جعبه‌های جدید را می‌توان در این نقاط به شبکه اضافه کرد و یا برخی جعبه‌ها را از آن حذف کرد. در این نقاط امکان ذخیره‌سازی تاریخچه‌ی محدودی از داده‌ها وجود دارد که توسط کاربر قابل تنظیم می‌باشد. view یک شبکه است که در آن داده‌ها از تعدادی جعبه گذشته و نتیجه‌ای که در نهایت تولید می‌شود ذخیره‌سازی می‌گردد. سایر برنامه‌های کاربردی می‌توانند به انتهای این view ها متصل شده و از داده‌های آنها استفاده کنند. درخواست ad-hoc نشان داده شده در پایین شکل نیز به یک نقطه‌ی اتصال متصل شده است و تا زمانی‌که به صورت صریح این اتصال قطع نشود، این درخواست بر روی داده‌های ذخیره شده در این نقطه‌ی اتصال اجرا می‌شود.