دانشگاه اصفهان
دانشکده فنی و مهندسی
گروه کامپیوتر

معماری نرم‌افزاری پایگاه‌های داده‌ی متحرک[1]

تحقیق درس: پایگاه‌داده پیشرفته

استاد گرامی: دکتر براآنی

تهیه‌کننده: محسن مؤمنی[†]

چکیده

پایگاه‌های داده‌ی متحرک، در سال‌های اخیر و با گسترش فراوان کاربردهای وسایل کامپیوتری متحرک[2]، رفته‌رفته به عنوان یکی از انواع اساسی و پرکاربرد پایگاه‌های داده مطرح می‌گردند. مسائل و مشکلاتی که در طراحی و پیاده‌سازی این انواع خاص پایگاه داده، علاوه بر مشکلات کلی موجود در پایگاه‌های داده توزیع شده، که پایگاه‌های داده‌ی متحرک نوع خاصی از آن ها هستند، در این مجال مطرح می‌گردد. در قسمت اصلی مقاله اما تأثیر این مسائل، بر طراحی معماری نرم‌افزاری پایگاه‌های داده‌ی متحرک، مورد بررسی قرار گرفته است.

پس از مروری کوتاه بر مسائل اصلی مطرح در مورد پایگاه‌های داده ی متحرک و عملیات خاص آن‌ها، معماری نرم‌افزاری پایگاه‌های داده‌ی متحرک به تفصیل مورد بررسی قرار گرفته و مسائل و معضلات آن را بررسیده‌ایم.

---

[1] Software architecture of mobile databases

[†] mohsenmomeni@yahoo.com

[2] Mobile devices

۱ انگیزه‌های تحقیق در زمینه‌ی پایگاه‌داده‌ی متحرک

مدیریت داده‌ی مؤثر به عنوان یکی از مهم‌ترین و شاخص‌ترین مسائل مربوط به توسعه‌ی نرم‌افزارهای کاربردی متحرک شناخته شده است. در مثال‌های فراوانی می‌توان نشان داد که چگونه مدیریت صحیح و مورد اعتماد داده‌ها در طراحی یک نرم‌افزار متحرک، می‌تواند مسائل و مشکلات اصلی آن را رفع نماید و آن کاربرد را قابل استفاده سازد.

امروزه کاربردهای فراوانی برای تجهیزات و وسایل کامپیوتری متحرک پیشنهاد شده، که در غالب آن‌ها مسئله‌ی پایگاه‌داده –ساختار، مدیریت و ارتباطات آن– مسئله‌ای اساسی در طراحی و پیاده سازی نرم‌افزار کاربردی بوده‌است. در این مقاله می‌کوشیم در ابتدا مروری کوتاه بر اصلی‌ترین و بنیادی‌ترین مشکلات پایگاه‌داده‌ی متحرک داشته باشیم و سپس وارد موضوع اصلی مورد بحث این مقاله، یعنی معماری نرم‌افزاری پایگاه‌های داده‌ی متحرک شده و آن را نسبتاً به تفصیل مورد بررسی قرار دهیم.

۲ مسائل اساسی در طراحی پایگاه‌داده‌ی متحرک

پایگاه‌های داده‌ی متحرک (که اخیراً فراوان مورد بررسی قرار گرفته‌اند) و پایگاه‌های داده‌ی توزیع شده‌ی سنتی[3] (که مدت زیادی است موضوع تحقیقات و بررسی‌ها هستند) گر چه دو موضوع متفاوت از یکدیگر را تشکیل می‌دهند و اختلافاتی نیز با هم دارند، اما در هر حال هر دو از معماری‌های تقریباً مشابهی بهره می‌جویند و بسیاری از ایده‌های طراحی و پیاده‌سازی مفاهیم، در آن‌ها با یکدیگر اشتراکات وسیعی دارند. بسیاری از مسائل مورد بحث در مدیریت‌داده‌های توزیع‌شده، در سیستم‌های پایگاه‌داده‌ی متحرک نیز مورد توجه هستند و به‌همین سان ایده‌های جدید در هر یک می‌تواند برای دیگری نیز قابل اجرا و استفاده باشد.

اما در اینجا بیش‌تر بر روی مسائل و نکاتی تأکید می‌کنیم، که بیش‌تر در مورد پایگاه‌های داده‌ی متحرک مطرح می‌شوند و متناسب با ویژگی‌های خاص این انواع پایگاه‌داده می‌باشند. درباره‌ی پایگاه‌های داده‌ی متحرک، سه مسئله‌ی اصلی، علاوه بر مسائل مطرح در توزیع‌شدگی و پایگاه‌های داده‌ی توزیع‌شده، مورد توجه است:

۱) نسبتاً غیر قابل اعتماد بودن اتصال‌ها. به دلیل محدودیت باطری اغلب تجهیزات متحرک، این تجهیزات معمولاً با قطع اتصال‌های فراوانی در حین کار خود روبه‌رو هستند. علاوه بر این پهنای

[3] Conventional distributed databases

باند در دسترس محدودی وجود دارد و سرور پایگاه‌داده هم ظرفیت پاسخگویی محدودی دارد. این مسائل نباید باعث شوند، پرس‌وجو با عدم پاسخ یا کندی در پاسخگویی روبه‌رو شود.

۲) محدودیت‌های گنجایشی برای ذخیره‌سازی داده. محدودیت دیگر پایگاه‌های داده‌ی متحرک، محدودیت گنجایش ذخیره‌سازی اطلاعات در آن‌هاست. با وجود پیشرفت‌های فراوانی که در سال‌های اخیر در تکنولوژی flash disk‌ها صورت گرفته، اما هنوز این عناصر با محدودیت ظرفیت و محدودیت برآوردن داده‌ها[4] روبه‌رو هستند. حتا در صورتی که حجم ذخیره سازی از محدودیت‌های امروزی فراتر رود، نیز باز حجم داده‌ها به قدری وسیع است، که این مشکل همچنان در غالب پایگاه‌های داده‌ی متحرک پابرجا خواهد بود و اجازه‌ی رونوشت کامل از پایگاه‌داده را در پایگاه‌های داده‌ی متحرک نخواهد داد.

۳) امنیت و رمزگذاری. که در محیط گسترده‌ای که پایگاه‌داده‌ی متحرک در آن می‌تواند مورد دستیابی قرار گیرد، یک امر حیاتی است. مسائل امنیت و رمزنگاری در مورد پایگاه‌های داده‌ی متحرک، شکل حادتری به خود می‌گیرند، زیرا در اینجا وسایل و تجهیزات متحرک می‌توانند به آسانی وارد ارتباط گردند و شناسایی، تشخیص و اطمینان کاربران در طول نشست تبادل داده بسیار مهم است.

۳ مُدهای عملیاتی در پایگاه‌داده‌ی متحرک

به خاطر محدودیت منابع، چندین مد عملیاتی برای پایگاه داده‌ی متحرک وجود دارد. این مدها در پروتکل‌هایی که برای کار در محیط متحرک نوشته می‌شوند، باید لحاظ گردند. به همین خاطر گاهی پروتکل‌هایی که برای محیط معمولی نوشته شده‌اند، به آسانی قابل تبدیل به پروتکلی برای محیط متحرک نیستند. سه مد عملیاتی اصلی در این محیط عبارتند از:

۱) مُد اتصال کامل[5]. که در آن کامپیوتر متحرک، به شکل دائمی به سرور متصل است. در این مد پروتکل hand-over مورد استفاده قرار می‌گیرد، که در آن در هنگام انتقال کامپیوتر متحرک، سرور جدیدی به صورت شفاف برای آن تعیین می‌شود و ارتباط جدیدی شکل می‌گیرد.

---

[4] load
[5] Fully connected mode

۲) *مُـد عدم اتصال*[6]. در این مُد disconnectionهای متعدد باید لحاظ گردند.یک راه استفاده از پروکسی است، که اطمینان از انجام پرس‌وجو، حتا در حالتی که کـامپیوتر متحـرک disconnect شده است، را تأمین کند و تقاضای به‌روزرسانی در اتصال دوباره از پروکسی انجام شود.

۳) *مد اتصال ضعیف (جزئی)*[7]. در این حالت، قطعه‌ی متحرک با استفاده از پهنای باندی پـایین و متناوب با سرور ارتباط دارد. پروتکل اتصال ضعیف برای محدود کردن ارتباط در حالت پهنای باند محدود (که معمولاً وقتی قطعه‌ی متحرک بر لبه‌های یک سلول قرار دارد روی می‌دهد)، نیاز است.

۴ تلخیص داده[8] و روش‌های آن

تلخیص داده‌ها در پایگاه‌های داده‌ی متحرک، به خاطر ظرفیت محدود آن‌ها یکی از اعمـال اساسـی است. این کار برای کاهش حجم داده‌ی انتقالی انجام می‌گیرد. در کل روش‌های مختلـف تلخـیص داده‌ها را می‌توان به دو دسته ی کلی تقسیم نمود. روش‌هایی که در این خلاصـه‌سـازی بـه میـزان اهمیت داده‌ها برای کـاربر و تعـدد اسـتفاده از آن‌هـا، اهمیـت مـی‌دهنـد و روش‌هـای context sensitive نامیده می‌شوند و روش‌هایی که به این مورد اهمیت نمی‌دهند و insensitive نامیـده می‌شوند. داده‌ی حساس به بستر، اطلاعاتی است که ارتباطی معنایی با کاربر استفاده کننده از آن‌هـا دارد. و از اطلاعات آن می‌توان در کاهش اندازه‌ی پایگاه‌داده استفاده نمود. در [۴] روش‌های نـوع نخست به تفصیل مورد بررسی قرار گرفته است.

در اینجا روش‌هایی از هر دو نوع اختصاراً معرفی می‌گردند.

چندین روش مختلف برای تلخیص داده‌ها وجود دارد. کلاً می‌توان آن‌ها را در ۴ دسته‌ی زیر قـرار داد:

۱) *تکنیک‌های تغییر شما*[9]

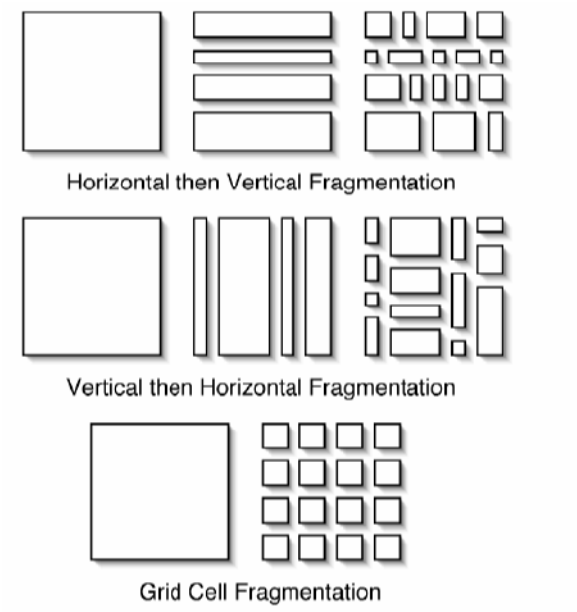
که در آن‌ها با ایجاد تغییر در شِمای رابطه‌های یک پایگاه داده اندازه‌ی آن کاهش می‌یابد. خـود این روش سه تکنیک دارد:

1)Projection (vertical) methods
2)Selection (Horizontal) methods
3)Hybrid

که روش سوم خود می‌تواند به صورت ترکیب‌های مختلفی از روش‌های ۱ و ۲ مورد اسـتفاده قرار گیرد. سه نوع مختلف از روش سوم در (شکل ۱) دیده می‌شوند.

---

[6] Disconnected mode
[7] Partial or Weak connection mode
[8] Data Summarisation
[9] Schema



شکل ۱- روش‌های تکه‌تکه‌سازی ترکیبی [۱]

۲) تکنیک‌های تغییر دامنه

استفاده از سلسله مراتب‌های معنایی، انتزاع حین گردآوری، کاهش داده‌ی تدریجی و جانشین‌ها تکنیک‌های مختلف این دسته هستندع که تا کنون در کاربردهای مختلف استفاده گردیده‌اند. خلاصه‌ای از چگونگی انجام این روش‌ها در [۱] آمده است.

۳) تکه‌تکه‌کردن و تخصیص داده

اغلب قطعه‌قطعه‌سازی دادا و تخصیص آن دو فرآیند جداگانه پنداشته می‌شوند. در بسیاری از روش‌های قطعه‌قطعه‌سازی در قطعه‌قطعه‌کردن مجموعه‌ی انتخابی، یک تراکنش یا پرس‌وجو به عنوان پایه‌ی تقسیم‌بندی رابطه‌ی سراسری قلمداد می‌گردند. صفت یا ماتریس پیش‌بینی استفاده در این روش برای تقسیم‌بندی (افقی یا عمودی یا هر دو) استفاده می‌شود.

۴) فشرده‌کردن

روش‌های مختلفی برای فشرده‌سازی داده‌ها معرفی گردیده‌اند. این روش‌ها و کاربردهای خاص آن‌ها برای داده‌ها و پایگاه‌های داده در [*Severance, 1983*]، [*Cannane et al. 1999*]، [Roth & Van Horn, 1993]، [Cormack, 1985] و چندین منبع دیگر معرفی گردیده‌اند. در کل فشرده سازی این حُسن را دارد، که علاوه بر کاهش حجم داده، به دلیل تغییر ساختار آن، مزایایی برای امنیت نیز فراهم می‌کند. اما کاربرد تنهای ان چندان در پایگاه‌های داده‌ی متحرک مؤثر نیست و می‌تواند در کنار دیگر روش‌ها و در ترکیب با آن ها به کار گرفته شود.

۵ عملیات در پایگاه‌داده‌ی متحرک

عملیات‌های مهم در پایگاه داده‌ی متحرک را می‌توان در یک تقسیم‌بندی به موارد زیر تقسیم نمود:

۱) پردازش پرس‌وجو و بهینه‌سازی.

برای پردازش پرس‌وجوها سه روش کلی شناخته شده‌اند ***[Roddic, 1997]*** (۴):

نخست، آنکه به پردازشگر پرس‌وجو اجازه دهیم، پایگاه‌داده‌ی مناسب برای پاسخ را، خود از میان پایگاه‌داده‌ی خلاصه‌شده[10] و پایگاه‌داده‌ی اصلی تعیین کند. نوعی ***انتخاب هوشمندانه*** باید برای کاهش هزینه‌ها در این روش صورت گیرد.

در روش دوم، که ***روش دانه‌درشت***[11] نامیده می‌شود، به هر دوی پایگاه‌داده‌های خلاصه و اصلی *query* ارسال می‌شود و نخستین پاسخ دریافت شده مورد استفاده قرار می‌گیرد. این روش هزینه‌ی زیادی را به سیستم تحمیل می‌کند، اما این حُسن را دارد که به صورتی تضمین‌شده به‌ترین سرعت را در کار خود دارد.

در روش سوم، یا ***روش دانه ریز***[12]، *query* به یک‌سری *segment* تقسیم می‌گردد و این *segment*ها برای پردازش به صورت سریال یا موازی به دو پایگاه ارسال می‌گردند. این روش استفاده‌ی مؤثر از پایگاه‌داده‌ی محلی را تضمین می‌کند و هزینه ی کمی را تحمیل می‌نماید. اجرای موازی این روش احتمالاً سریع‌تر خواهد بود، اما ممکن است، شامل ارتباط‌های تکراری زائد فراوانی باشد.

جواب‌های برآمده از پایگاه‌داده ی خلاصه شده ممکن است، یکی از ۴ حالت زیر را داشته باشند:

Complete and sound، Potentially anderstated، Potentially overstated، wrong.

سه الگوریتم پرس‌وجو در ***[Ganguly & Alonso, 1993]*** ارائه گریده است:

۱) ***namely adaptation of partial order dynamic programming***

۲) ترکیبات خطی[13]

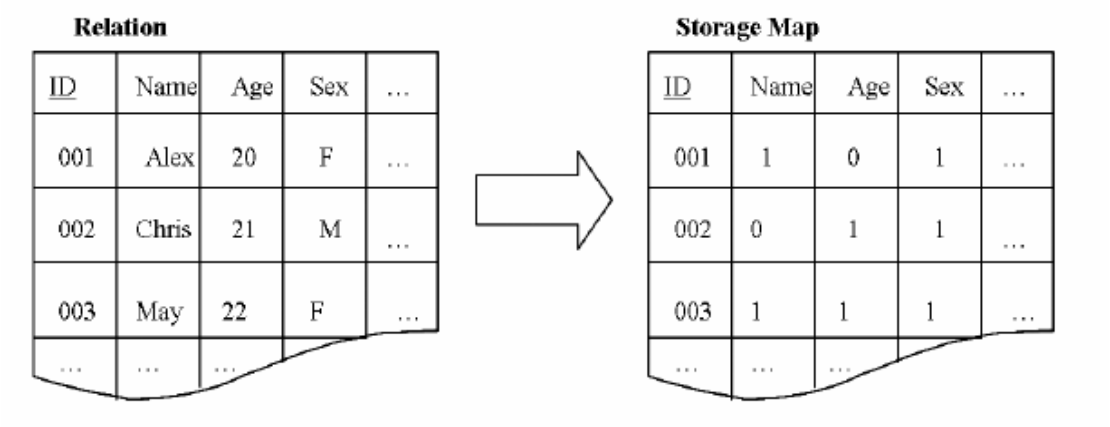
۳) الگوریتم مجموعه‌ی خطی[14].

در ***[Roddic & Chan, 2003]*** روشی مبتنی بر نقشه‌ی ذخیره‌سازی[15] برای پردازش پرس‌وجوها ارائه گردیده است. در این روش برای هر رابطه یک نقشه‌ی ذخیره‌سازی همانند شکل ۲ ساخته

---

[10] summarized
[11] Coarse grained
[12] Fine Grained
[13] Linear combinations
[14] Linearset algorithm
[15] Storage map

می‌شود و در هنگام تصمیم‌گیری برای استفاده از پایگاه خلاصه شده یا پایگاه اصلی، تصمیم‌گیری از روی آن انجام می‌شود و داده‌های مورد نیاز، طبق الگوریتمی دانه‌ریز آورده می‌شوند.



*شکل ۲ **storage map** (منبع:۱)*

۲) شماهای تکرار داده[16]. شمای تکرار بیانگر دو چیز است: یک پایگاه داده‌ی تکرارشده[17] به چه طریقی به روز رسانی می‌شود، و این به‌روزرسانی چگونه در بقیه‌ی تکرارها انتشار می‌یابد.

دو مکانیسم ارتباط وجود دارد: در روش نخست یک سایت اصلی[18] انتخاب می‌شود که همه‌ی به روزرسانی‌ها از طریق آن شکل می‌گیردو این به روزرسانی‌ها انتشار می‌یابد. در روش دوم از راهبرد "به‌روز رسانی همگانی[19] استفاده می‌گردد که در آن عمل به روز رسانی در سایت منبع[20] انجام می‌گیرد و سپس به تمام سایت‌هایی که به آن داده نیاز دارند منتشر می‌گردد.

در پایگاه‌های داده‌ی توزیع‌شده دو پروتکل اصلی برای انتشار وجود دارد. ***Eager*** و ***Lazy***. روش ***eager*** که در پایگاه‌های داده ی توزیع‌شده به خوبی عمل می‌کند، برای پایگاه‌های داده‌ی متحرک نامناسب است. این نیز به خاطر تعدد ***disconnection*** در پایگاه‌داده‌ی متحرک است. ترکیبی از دو روش توسط ***Lubinsky*** معرفی گردیده است.

---

16 Data Replication Schemes
17 Replicated Database
18 Primary cite
19 Update everywhere
20 origin

۳) کنترل همروندی. برای اطمینان از سازگاری[۲۱] پایگاه‌داده در ***DBMS***، در حالتی که چندین کاربر توانایی دسترسی و تغییر آن‌ها را دارند، مکانیسم‌های کنترل همروندی مورد نیاز هستند. البته این امر بستگی به نحوه‌ی به‌روزرسانی پایگاه‌داده نیز دارد. مثلاً اگر امکان به‌روزرسانی تنها برای سایت مرکزی یا اصلی وجود داشته باشد، مدیریت ***two-phase locking*** می‌تواند سازگاری را فراهم نماید.برای پایگاه‌داده‌ی توزیع‌شده، روش رای‌گیری[۲۲] در این مورد به‌کار می‌رود. که در آن در طول مدت به‌روزرسانی، داده قفل می‌شود. این چنین روشی برای پایگاه‌داده‌ی متحرک نمی‌تواند روش مناسبی باشد.در ***[Jing et al, 1995]*** روش قفل دو فازی بهینه[۲۳] به عنوان روشی مناسب برای پایگاه‌های داده‌ی متحرک معرفی گردیده است.

در معماری بایو که در ادامه مطرح می‌کنیم، یک سیستم سازگاری ضعیف که سازگاری میان تکرارها را تضمین می‌کند، برای آن در ***[Terry et al, 1995]*** ارائه شده است. در معماری بایو دسترس‌پذیری بالایی برای داده‌ها ایجاد شده و این امر به هزینه‌ی سازگاری ضعیف در آن به‌وجود آمده است. در این سیستم به کاربر و سرور هر دو اجازه‌ی به‌روزرسانی داده می‌شود.

۴) پشتیبانی تراکنش. پشتیبانی تراکنش تکیه بر در دسترس بودنِ همه‌ی اطلاعات لازم برای کامل‌شدن عملیات دارد. اگر داده‌ای در دسترس نباشد، تراکنش ***fail*** می‌شود و یا باید برای در دسترس قرار گرفتن داده صبر کند.

تراکنش‌ها در پایگاه‌داده‌ی متحرک بسیار متفاوت از تراکنش‌های پایگاه داده‌ی متمرکز یا توزیع‌شده است. از این خصوصیات متفاوت، یکی احتیاج به تقسیم تراکنش به قسمت‌های مختلف و اجرای محلی و از راه دور آن، در برخی موارد است. یا پشتیبانی تراکنش توسط پایگاه‌های داده‌ی ثابت، محاسبه ی اجزاء مختلف تراکنش در سرورهای مختلف در هنگام حرکت کامپیوتر متحرک و یا دوام تراکنش در طول قطع اتصال‌های متعدد همگی تفاوت‌هایی را در خصوصیات پایگاه‌داده‌ی متحرک ایجاد می‌کنند.

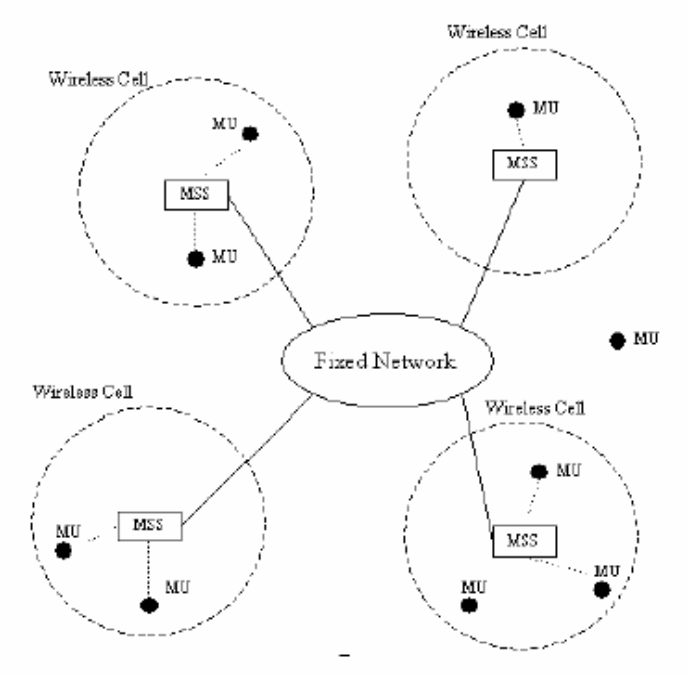
برای پردازش تراکنش در پایگاه‌داده ی متحرک روشی معناگرا در ***[Walborn & Chrysanthis, 1995]*** معرفی گریده، که ایده ی اشیاء قابل تقسیم و قابل ضبط را مطرح نموده است. که سازگاری را در کل پایگاه افزایش می‌دهد و تأثیرِ ***caching*** را بیش‌تر می‌سازد.

---

[21] consistency
[22] voting
[23] An Optimistic Two Phase Locking

روش‌های متعدد دیگری نیز برای انجام تراکنش‌ها معرفی گردیده که توضیح آن‌ها از مجال این مقاله بیرون است. مثلاً تراکنش‌های ***Kangaroo*** یا مدل تراکنش ***open-nested*** یا مدل تراکنش ***multi-version*** یا روش‌های دیگری که در کاربردهای مختلف معرفی گردیده‌اند.

۵) recovery سیستم. در سیستم‌های پایگاه‌داده‌ی مرکزی یا توزیع‌شده‌ی مبتنی بر کپی کلی داده‌ها، در هنگام وقوع ***failure*** برای کپی اصلی، ***clientها*** دیگر نمی‌توانند، در به‌روز رسانی داده‌ها اثری داشته باشند. اگر مشتری توانایی ***cache***کردن داده‌ها به صورت محلی را نداشته باشد، دسترسی به خواندن داده‌ها نیز وجود ندارد. در این موارد معمولاً مرکزی جدید، نقش کپی اصلی را به عهده می‌گیرد و معمولاً این کار از طریق الگوریتم انتخاب[24] انجام می‌گیرد. و تا زمانی که هماهنگ‌کننده‌ی اصلی ***recover*** نشده است، کار هماهنگی را به عهده می‌گیرد. در پایگاه‌های داده ی متحرک نیز در مورد ایستگاه‌های ثابت (سرورهای ساکن) می‌توان از چنین روشی بهره برد.اما به خاطر تعدد ***disconnection*** در شبکه برای کامپیوترهای متحرک، در مورد آن‌ها چنین عملکردی معمول نیست. در شبکه‌های ***ad-hoc*** شبکه متناوباً به قسمت‌های مختلفی تقسیم می‌شود و توپولوژی آن تغییر می‌یابد. یک الگوریتم خوشه‌بندی برای توسط ***Aggarwal*** ودیگران، برای تشخیص کامپیوترهای متحرک دسترس‌پذیر شبکه، برای طراحی و ساختیک شبکه ی پراکنده معرفی شده است.



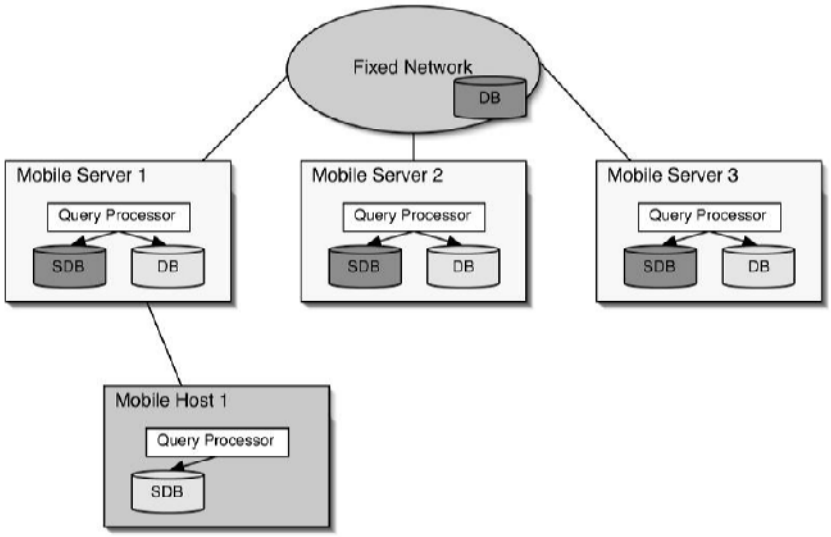
شکل ۳ محیط ارتباطی نوعی نرم‌افزارهای متحرک

[24] Election algorithm

۷ معماری پایگاه‌داده‌ی متحرک

محیط ارتباطی در مورد نرم‌افزارهای متحرک اثر قابل توجهی بر ساختار و معماری آن‌ها دارد. چنانکه در شکل ۳ دیده می‌شود، محیط این تجهیزات، بی‌سیم است و از یکسری سلول‌های بی‌سیم، که در هر یک از آن‌ها یک سرور قرار دارد و کامپیوترهای متحرک در آن حوزه، با آن سرور خاص ارتباط برقرار می‌کنند، تشکیل شده است.

معماری نوعی پایگاه‌های داده‌ی متحرک، دربردارنده‌ی یک پایگاه داده‌ی اصلی و تکه‌های کوچکی از آن در پایگاه‌های داده‌ی متحرک است. در شکل زیر (شکل ۴) معروف‌ترین معماری در نظر گرفته شده برای پایگاه‌های داده‌ی متحرک (Madria *et al*, 1998) نشان داده شده است. این معماری را می‌توان یک معماری سرویس‌دهنده/مشتری در نظر گرفت که در آن پایگاه داده‌ی اصلی در یک سرور واقع شده و پایگاه‌های داده‌ی خلاصه شده و کوچک در مشتری‌ها یا همان پایگاه‌های داده‌ی متحرک، مستقر هستند.



شکل۴ – معماری کلی پایگاه‌داده‌های متحرک معمولی (*Madria et al*, 1998)

معماری سرویس‌دهنده/مشتری دیگری به نام بایو[۲۵] (شاخه‌ی فرعی) طراحی گردیده (Demers *et al*, 1994) که از معماری سرویس‌دهنده–مشتری‌ای به‌علاوه‌ی سرورهای سَبُک[۲۶] بهره می‌گیرد. که این سرورهای سَبُک به مشتریان معینی اجازه‌ی ذخیره‌ی داده‌هایشان در cashهای آن‌ها را

[25] Bayou
[26] Lightweight servers

می‌دهند. چنین تکنیکی دسترسی به داده را افزایش می‌دهد، اما به دلیل خصوصیت سازگاری ضعیف[۲۷] آن، تشخیص ناسازگاری‌ها و conflictها در آن فوق‌العاده مشکل‌ساز می‌گردد.

دیگر معماری‌های پایگاه‌داده علاوه بر دو عنصر فوق (سرویس‌دهنده و مشتری) شامل عنصر دیگری به نام عامل[۲۸] نیز هستند، و بنابراین یک معماری سرویس‌دهنده/عامل/مشتری دارند. نقش عامل می‌تواند تفاوت‌های زیادی در کاربردهای مختلف داشته باشد. این عامل هم می‌تواند در طرف مشتری و هم می‌تواند در طرف سرویس‌دهنده مستقر باشد و هم می‌تواند بین آن‌ها منتقل شود.

در معماری‌های بالا توپولوژی شبکه ثابت در نظر گرفته شده است. یعنی فرض شده که سرورها ثابت هستند و کامپیوترهای متحرک میان سرورهای مختلف می‌توانند منتقل شوند. اما در سیستم شبکه‌ی نامنظم[۲۹] که در (Fife & Gruenwald, 2003) پیشنهاد گردیده، سیستم بی‌سیم می‌تواند چندین بار توپولوژی خود را تغییر دهد. گره‌های این شبکه هم مشتری و هم سرور هستند و هر دوی این‌ها بی‌سیم و متحرک. این امکان تحرک برای سرورها موجب می‌شود، گره‌ها بتوانند در یک شبکه‌ی تکی یا یک شبکه‌ی گسترده‌تر باشند. مثالی از این شبکه بلوتوث است. شبکه‌ی نامنظم به معماری peer-to-peer منجر می‌گردد. که در آن مشتری‌ها می‌توانند با دیگر مشتری‌ها نیز ارتباط داشته باشند و داده‌ها را به اشتراک بگذارند. این امر به دسترسی‌پذیری بیش‌تر، اما طبق معمول به پیچیدگی برقرارسازی عدم ناسازگاری می‌انجامد. (البته اگر مشتری‌ها قادر به به‌روزکردن داده‌ها باشند)

۱-۷ معماری‌های Client-Server

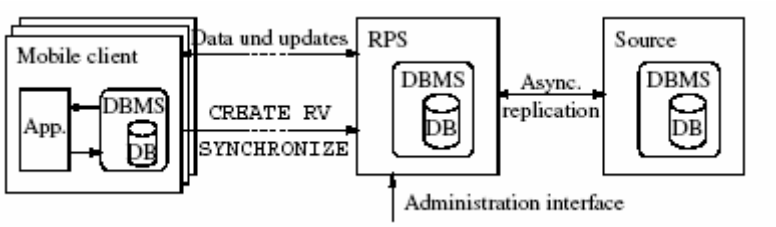
با توجه به محیط ارتباطی‌ای که برای تجهیزات متحرک، نشان دادیم، معماری سخت‌افزاری این محیط، بر معماری نرم افزاری نیز اثر می‌گذارد و ما را ملزم به استفاده از معماری C/S در محیط آن می‌کند. در این معماری همانگونه که گفتیم، در هر سلول (یا هر گره) یک سرور قرار گرفته، که تجهیزات متحرک با ورود به دامنه‌ی ان تحت سرویس‌دهی آن سرور خاص قرار می‌گیرند. در این معماری بسته به اینکه از چه تجهیزاتی استفاده کنیم، ممکن است، سرور خود دارای معماری مرکز گرا، یا توزیع‌شده باشد. و بدین سان تکرار داده در سرورها می‌تواند حالات مختلفی را که در پایگاه‌های داده‌ی توزیع‌شده دیده‌ایم داشته باشد. تکرار کلی داشته باشیم، یا fragmentation کلی و یا ترکیبی از این‌ها.

---

[27] Weak consistency property
[28] agent
[29] Ad-hoc

در محیط‌های پایگاه‌داده‌ی متحرک با معماری C/S سرویسی برای تکرار داده‌ها[30] مورد نیاز است، تا دسترس‌پذیری[31] داده را در مُد عدم اتصال نیز تأمین کند.

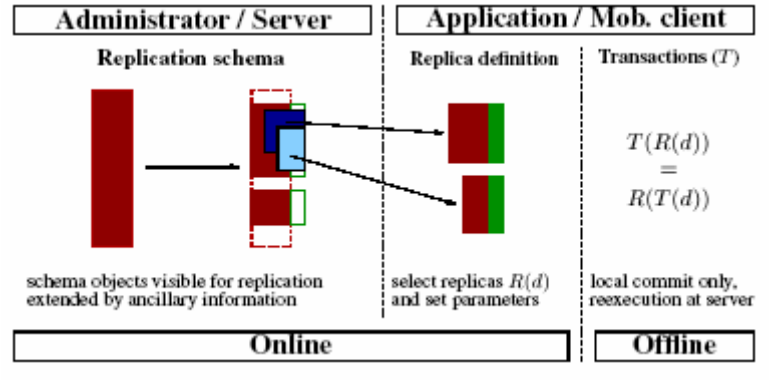
در [۲] راهبردی مشتری‌گرا برای پایگاه‌داده‌های رابطه‌ای ارائه شده است. که با استفاده از آن برنامه‌های کاربردی متحرک می‌توانند داده‌های تکراری و اختیارات تکرار را به صورتی وابسته به موقعیت درخواستی تعریف کنند. در این راهبرد یک معماری سه لایه برای پایگاه داده در نظر گرفته می‌شود. [Gollmick, 2002] به تشریح مزایای استفاده از سرور پروکسی تکرار[32] (RPS) پرداخته است. در این معماری بین مشتری صاحب پایگاه‌داده‌ی متحرک و سرور ثابت، RPS قرار می‌گیرد. در RPS به خاطر امنیت بیش‌تر، تسهیلات scalability و گام‌های توسعه‌ی بیش‌تر، یک کپی از داده‌ی اصلی نگه‌داری می‌شود. (شکل ۵)



شکل ۵ – معماری سه‌لایه (منبع: ۲)

در چنین راهبردی سه فعالیت اصلی مورد نیاز است: تعریف تکرارها، همگام‌سازی و پردازش تراکنش‌ها. دو فعالیت نخست، باید online انجام گیرند و فعالیت سوم offline انجام می‌گیرد.
در [۲] تعریف تکرارها به دو گام تقسیم شده است. تعریف شِمای تکرار و تعریف replica.
تعریف شِمای تکرار توسط admin و بر روی سرور انجام می‌گیرد. اما replicaها در واقع viewهایی هستند که توسط مشتری تعریف می‌شوند و نقطه‌ی کلیدی راهبرد مشتری‌گرای ارائه شده در [۲] هستند.سرویس تکرار مسئول تأمین داده برای جدول‌های replica که به صورت محلی هستند، می‌باشد.

[30] replication
[31] availability
[32] Replication proxy server



شکل ۶- فرآیند تکرار داده در معماری C/S (منبع:۲)

در مجموع روش‌های متعددی برای پیاده‌سازی ساختار c/s در کاربردهای مختلف معرفی گردیده است. مثلاً در [۱۰] یک ***scalable server*** مسئول تأمین ***scalability*** و ***expandability*** در سیستم پایگاه داده است.

۲-۷ معماری‌های Client-Agent-Server

معماری‌های C/A/S همانطور که از اسم شان پیداست، دارای سه‌جزء نرم‌افزاری سرور، مشتری و عامل می‌باشند. اما در مقالات مختلف، کارکردهای متفاوتی برای این عامل در نظر گرفته شده است و بسته به این کارکرد، در معماری معرفی شده، عامل در طرف مشتری، در طرف سرور یا در انتقال میان آن‌ها در نظر گرفته شده است. یک معماری وابسته به پهنای باند، که میانجی[۳۳] نامیده شده، در ***[Zenel & Duchamp, 95]*** معرفی گردیده است. در این معماری تنها به داده‌های ضروری اجازه‌ی گذر از خط داده می‌شود و دیگر داده‌ها، فیلتر شده با تأخیر ارسال می‌شوند.

[Heurer & Lubinsky, 1996] عاملی اطلاعاتی معرفی نمودند، که هوشمندانه انتقال داده‌های خاص چندرسانه‌ای را به خوبی به انجام می‌رسانید.

[Lauzac & Chrysanthis, 1998a , 1998b] نیز به معرفی عاملی که ***view holder*** نامیده می‌شد، و برای انجام به‌روز رسانی در طرف مشتری پایگاه داده به کار گرفته شده بود، پرداختند.

در بررسی ساختار معماری‌های پایگاه‌داده‌ی متحرک عامل‌گرا به بررسی معماری‌ای می‌پردازیم که در منبع [۹] معرفی گردیده است. از عامل‌ها به خاطر قابلیت‌های فراوان آن‌ها در محیط پایگاه‌داده‌ی متحرک، به خوبی می‌توان برای تسهیل امور بهره گرفت. در معماری معرفی شده در [۹] از HAFS[۳۴] استفاده می‌شود. HAFS یک سیستم مبتنی بر عامل است، که هدف آن تامین امنیت و محافظت از داده‌ها در یک محیط شامل منابع داده‌ای پراکنده است. در معماری این سیستم به هر

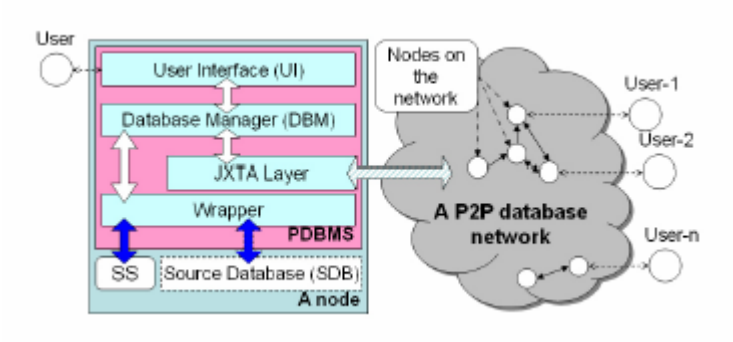
[33] intermediary
[34] Highly available federated systems

HAFS خواه ساکن باشد و خواه متحرک، یک عامل امنیت منتسب می‌شود، که مسئول تأمین امنیت برای اشیاء داده‌ای ذخیره شده در آن HAFS است. هر وسیله‌ای که عاملی را تولید کند، به عنوان سرور عامل شناخته می‌شود. در محیط متحرک، معمولاً میزبان‌های ثابت و سرورها به عنوان سرور عامل محسوب می‌گردند. اما در شبکه‌های ad-hoc حتا برخی از کامپیوترهای متحرک نیز می‌توانند به عنوان سرور عامل، به ایجاد عامل مبادرت ورزند. هر عامل می‌تواند یک یا چند منبع داده را محافظت نماید. که این به تجهیزات کوچک، اجازه می‌دهد، نیاز به ساخت عامل نداشته باشند. سرور عامل نه‌تنها اجازه‌ی ارتباط بین عامل‌ها را می‌دهد، بلکه امکان ساکن شدن آن‌ها در مشتری‌ها را نیز فراهم می‌کند.

۳-۷ معماری‌های Peer-to-Peer

در شبکه‌های ad-hoc به دلیل ساختار ارتباطی میان تجهیزات متحرک، توپولوژی سیستم می‌تواند چندین بار تغییر یافته و دگرگون شود. در چنین محیطی سرورها نیز می‌توانند تحرک داشته باشند. چنین ساختار محیطی‌ای منجر به معماری P2P می‌گردد. در معماری P2P هر جزء نرم‌افزاری هم سرور است و هم مشتری و این ساختار availability بالایی را فراهم می‌کند، زیرا مرکزیت ساختار C/S و وابستگی آن به پایداری سرور را نیاز ندارد.

در [۳] معماری نرم افزاری شکل ۷ به‌عنوان یک معماری نرم افزاری برای یک پایگاه‌داده‌ی متحرک در محیط متحرک و البته به همراه دیگر اجزاء مورد نیاز برای برنامه ی کاربردی نشان داده شده است.



شکل ۷ – معماری پایگاه‌داده و برنامه‌ی کاربردی در شبکه‌ی P2P. (منبع:۳)

در [۵] نیز دسته پروتکل خاصی برای داده‌ی دینامیک معرفی گردیده، که در محیط ad-hoc نیز علاوه بر محیط پایگاه‌داده‌ی توزیع‌شده‌ی متحرک معمولی قابلیت‌ها و بهبودهای فراوانی را در دسترسی دینامیک و رفتار متحرک نشان می‌دهد.

۸ جمع‌بندی

در این مقاله کوشیدیم، معرفی مختصری از پایگاه‌داده‌های متحرک ارائه کنیم. مسائل و مشکلات مطرح درباره‌ی آن‌ها را مورد بررسی قرار دهیم و در قسمت اصلی به بررسی معماری‌های نرم‌افزاری ارائه‌شده برای پایگاه‌های داده‌ی متحرک بپردازیم و به تفصیل آن‌ها را مورد بررسی قرار دهیم. سه نوع اصلی معماری پایگاه‌های داده‌ی متحرک یعنی انواع C/S، C/A/S و P2P را بررسی نموده، به برخی مسائل مطرح در آن‌ها اشاره کردیم.

۹ منابع و مأخذ

1) **Darin Chan and John F. Roddick, 2005,** "*Summarisation for Mobile Databases*", Australia, Journal of Research and Practice in Information Technology*, Vol. 37, No.3, August 2005.*

2) **Christoph Gollmick, 2003,** "*Replication in Mobile Database Environments: A Client-Oriented Approach",* Proceedings of the 14th International Workshop on Database and Expert Systems*, © 2003 IEEE.*

3) **Fausto Giunchiglia and Ilya Zaihrayeu**, "*Coordinating Mobile Databases*"**,** *University of Trento, Italy***.**

4) **Darin Chan and John F. Roddick, 2003,** "*Context-Sensitive Mobile Database Summarisation*", Flinders University of South Australia, Australian Computer Society.

5) **Yanli Xia and Abdelsalam (Sumi) Helal, 2003,** *"A Dynamic Data/Currency Protocol for Mobile Database Design and Reconfiguration",* University of Florida, USA, ACM, *SAC* , March 9-12, 2003.

6) **Kam-yiu Lam, Tei-Wei Kuo, Wai-Hung Tsang and Gary C.K Law, 2000,** "*CONCURRENCY CONTROL IN MOBILE DISTRIBUTED REAL-TIME DATABASE SYSTEMS*", University of Hong Kong.

7) **Eric Jui-Lin Lu, Yung-Yuan Cheng, 2004,** "*Design and implementation of a mobile database for Java phones*", Elsevier 2004 B.V. All rights reserved.

8) **Val´erie Issarny, Ferda Tartanoglu, Jinshan Liu, 2004,** *"Software Architecture for Mobile Distributed Computing",* France, Proceddings of IEEE/IFIP WICSA'2004.
(http://www.extra.research.philips.com/euprojects/ozone/).

9) **Carlos Sanchez1, Le Gruenwald, 1999,** *"An agent based architecture using XML for Mobile Federated Database Systems".*

10) **Roger Bamford, Rafiul Ahad, Angelo Pruscino,** *"A Scalable and Highly Available Networked Database Architecture",* Oracle Corporation, California.