فصل ۱۴

تراکنش

اغلب اوقات مجموعه ای از چندین عملیات در یک پایگاه انجام می‌شود تا از دید کاربران پایگاه داده ،یک واحد باشد. غالبا مجموعه ای از عملیات در پایگاه داده، از دید کاربر به شکل یک عملیات واحد مشاهده می‌شود. به عنوان مثال،انتقال وجه از یک حساب جاری به یک حساب پس انداز از نظر مشتری یک عملیات مجزا در سیستم پایگاه داده است اما در واقع از چندین عملکرد تشکیل شده است. واضح است که این عملیات باید بصورت کامل انجام شود یا در صورت وقوع خطا اصلا انجام نگیرد.

مجموعه عملیاتی که یک واحد منطقی و مجزای کاری را تشکیل می‌دهد، تراکنش می‌نامیم. سیستم پایگاه داده باید اجرای صحیح تراکنش ها برخلاف خطاها را درصورتیکه یک تراکنش انجام شود یا انجام نشود تضمین کند. علاوه بر این سیستم پایگاه داده باید اجرای همزمان تراکنش ها را به روشی که مانع بی ثباتی آنها شود مدیریت کند. به عنوان مثال در انتقال سرمایه یک تراکنش تراز کلی مشتری را محاسبه می‌کند و ممکن است تراز بررسی حساب را پیش از اینکه توسط تراکنش ها انتقال سرمایه دهد مشاهده کند اما تراز پس انداز را پس از اینکه حساب معتبر شد مشاهده کند،بدین ترتیب نتیجه صحیح به دست نمی‌آید.

این فصل مفاهیم اساسی فرآیند تراکنش بانکی را معرفی می‌کند. جزئیات فرآیند تراکنش موازی و ترمیم خطاها به ترتیب در فصل های ۱۵ و۱۶ آمده است. موضوعات بیشتر فرآیند تراکنش در فصل ۲۶ مورد بحث قرار می‌گیرد.

۱۴–۱ مفهوم تراکنش

تراکنش واحدی از اجرای برنامه هایی است که احتمالا به داده های مختلف دسترسی می‌یابد و آنها را بروزرسانی می‌کند. معمولا یک تراکنش توسط برنامه کاربردی که به زبان کاربردی داده در سطح بالا (معمولا SQL) یا زبان برنامه نویسی (به عنوان مثال C++ یا جاوا) با دسترسی پایگاه داده تعبیه شده در JDBC یا ODBC است آغاز می‌شود. یک تراکنش محدود به اعلان های (یا هشدارهای عملکردی) شروع یا پایان تراکنش است. تراکنش تمام عملیات اجرایی از آغاز تا پایان تراکنش را دربرمی‌گیرد.

این مجموعه از مراحل باید برای کاربر به عنوان یک واحد مجزا و غیرقابل تقسیم انجام شود.به دلیل اینکه یک تراکنش غیرقابل تقسیم است یا تمام آن باید اجرا شود یا به طور کلی اجرا نشود. بنابراین اگر اجرای یک تراکنش آغاز شود اما به هر دلیلی خطا رخ دهد، هر تغییری در پایگاه داده انجام شود ممکن است باعث شود تراکنش بی خاتمه پایان یابد. این الزامات صرف نظر از شرایطی مانند خطای تبادل(به عنوان مثال اگر تقسیم بر صفر شود)،خرابی سیستم عامل یا توقف سیستم کامپیوتر اجرا می‌شود. همانگونه که مشاهده می‌کنیم، تضمین این الزامات دشوار است زیرا بعضی تغییرات ممکن است هنوز در متغیرهای حافظه اصلی مبادله

ذخیره شده باشند در حالیکه تغییرات دیگر در پایگاه داده نوشتن شده است و روی دیسک ذخیره می شود. به این ویژگی "همه یا هیچ کدام" به عنوان **اتمیک** اشاره می‌شود.

علاوه بر این به دلیل اینکه تراکنش یک واحد مستقل است،عملیات آن نمی تواند با عملیات پایگاه داده دیگر و یا بخشی از مبادله جدا شود. با وجود اینکه می‌دانستیم واقعیت کاملا متفاوت است قصد داریم اثر سطح کاربر این تراکنش ها را شرح دهیم. حتی یک SQLمجزا هم در برگیرنده دسترسی های مجزا به پایگاه داده است و یک تراکنش ممکن است شامل چندین حالت از SQL باشد.بنابراین سیستم پایگاه داده باید اقدامات ویژه به عمل آورد تا تضمین کند تراکنش ها به درستی بدون دخالت پایگاه های داده که به طور همزمان اجرا می‌شوند عمل می‌کنند.به این ویژگی به عنوان **ایزولاسیون** اشاره می‌شود. حتی اگر سیستم اجرای صحیح یک تراکنش را تضمین کند چنانچه سیستم به طور متوالی خراب شود این هدف کوچکی است و در نتیجه سیستم تراکنش را فراموش می‌کند. بنابراین عملکرد یک تراکنش باید در حین خرابی هم ادامه یابد. به این ویژگی **پایایی** می گویند.

به دلیل سه ویژگی بالا، تراکنش ها روش ایده آلی از تبادل ساختار یک پایگاه داده می‌باشند. این امر منجر به اعمال یکی از الزامات خود تراکنش ها می‌شود. یک تراکنش باید ثبات پایگاه داده را حفظ کند. چنانچه یک تراکنش از اعداد اتمی در ایزولاسیون اجرا شود از یک پایگاه داده نامتناقض شروع می‌شود،پایگاه داده باید مجددا در انتهای تراکنش به ثبات برسد. این الزام به ثبات به فراتر از محدودیت تمام داده هایی که پیش از این مشاهده کردیم وارد می شود.(مانند محدودیت های کلیدی اصلی،صحت ارجاعی،محدودیت های بررسی و مانند این). در مقابل انتظار می رود تراکنش ها به فراتر از تضمین حفظ محدودیت های ثبات نرم افزار های کاربردی وارد شود که برای استفاده از ساختار SQL، داده جامع بسیار پیچیده ای هستند. چگونگی انجام این عمل و مسئولیت برنامه نویسی که کدهای تراکنش را می‌نویسد چیست. این ویژگی **ثبات** نامیده می‌شود. به منظور توضیح دقیق مواردی که در بالا ذکر شد به یک سیستم پایگاه داده ای نیاز داریم که ویژگی های زیر از تراکنش را حفظ کند:

- اتمیک:تمام عملیات تراکنش به درستی در پایگاه داده انجام می‌شود یا در هیچ یک از تراکنش ها وجود ندارند.
- ثبات: اجرای یک تراکنش بصورت انفرادی(بدون هیچ تراکنش همزمانی) ثبات پایگاه داده را حفظ می کند.
- ایزولاسیون: حتی تراکنش های چندگانه ای که ممکن است همزمان اجرا شوند،سیستم را برای هر دو تراکنش T_1و T_2شروع و پایان اجرای تراکنش TIوT_2 ضمانت می کند.بنابراین هرتراکنش از تراکنش های در حال اجرای همزمان در سیستم ناآگاه است.
- پایایی: پس از اتمام با موفقیت تراکنش،تغییراتی برای ادامه عملیات پایگاه داده باید انجام شود حتی اگر سیستم با خطا مواجه شود.این ویژگی ها اغلب ویژگی های ACID نامیده می‌شوند،این لغت اختصاری از حرف اول هر کلمه این چهار ویژگی مشتق شده است.

همچنان که مشاهده خواهیم کرد،تضمین ویژگی ایزولاسیون ممکن است اثر چشمگیری بر عملکرد سیستم داشته باشد. به این دلیل بعضی کاربردها با ویژگی ایزولاسیون سازگار هستند. ما این سازگاری ها را پس از اولین تحقیق اجرای دقیق ویژگی های ACID بررسی می‌کنیم.

۲-۱۴ یک مدل ساده تراکنش

به دلیل اینکه SQL زبان پیچیده و قدرتمندی است،تحقیقات خود را از تراکنش با یک زبان پایگاه داده ساده آغاز می‌کنیم و بر زمانی تمرکز می کنیم که داده از دیسک به حافظه اصلی و از حافظه اصلی به دیسک انتقال می‌یابد. برای انجام این کار،عملیات حذف و خواندن را نادیده می‌گیریم و بررسی آنها را به بخش ۸-۱۵ موکول می‌کنیم.تنها عملیات حقیقی داده به زبان ساده ما به عملیات محاسباتی محدود می‌شود. پس از آن در مورد تراکنش هایی با زمینه ای واقعی برمبنای SQL با مجموعه عظیمی از عملیات بحث می‌کنیم. اقلام داده در مدل ساده شده ما شامل یک مقدار داده واحد است(در مثال ما یک عدد). هرداده با یک اسم شناسایی می‌شود.(عموما یک حرف واحد در مثال های ما A,B,C است).

ما مفهوم تراکنش را با استفاده از یک نرم افزار بانکی ساده نشان می دهیم که شامل چندین حساب و مجموعه ای از تراکنش ها می‌شود که به حساب دسترسی دارد و آن ها را بروز می‌کند. تراکنش های دسترسی داده از دو عملیات استفاده می‌کنند:

- خواندن(X): در حافظه اصلی متعلق به تراکنشی است که عملیات خواندن را اجرا می کند، داده X از پایگاه داده به یک متغیر منتقل و همچنین X نامیده می‌شود.
- نوشتن(X): مقدار متغیر X حافظه اصلی تراکنش را انتقال می‌دهد که نوشتن با داده X در پایگاه داده اجرا می‌شود.

دانستن این موضوع که تغییر داده ها تنها در حافظه اصلی است یا ذخیره سازی پایگاه داده روی دیسک اهمیت دارد. درسیستم پایگاه‌داده واقعی‌عملیات نوشتن الزاما نتیجه به روزرسانی سریع داده ها بر دیسک نیست،عملیات نوشتن ممکن است به طور موقت در جایی دیگر ذخیره شود و پس از آن روی دیسک اجرا شود. اما در حال حاضر فرض می‌کنیم که عملیات نوشتن پایگاه داده را سریعا به روز رسانی می‌کند. در فصل ۱۶ دوباره به این موضوع اشاره خواهیم کرد.

T_1را به عنوان تراکنشی در نظر بگیرید که ۵۰ دلار از حساب A به حساب B انتقال می‌دهد. این تراکنش ها را می توان همانند زیر تعریف کرد:

```
T1:read(A);
   A:=A−50;
   write(A);
   read(B);
   B:=B+ 50;
   write(B).
```

اکنون با هم ویژگی های ACID را بررسی میکنیم(به منظور تسهیل اجرا آنها را در ردیف های مختلف A-C-I-D بررسی میکنیم).

- ثبات: ثبات پایگاه داده نیازمند جمع Aو B است که باید توسط اجرای تراکنش بدون تغییر بماند. بدون نیاز به ثبات، پول ممکن است در تراکنش کم یا اضافه شود! اگر پایگاه داده پیش از اجرای تراکنش ثابت شود به سادگی این موضوع را می توان تائید کرد،پایگاه داده پس از اجرای تراکنش ثابت می‌ماند.

تضمین ثبات برای یک تراکنش مجزا مسئولیت برنامه نویس نرم افزاری است که تراکنش را کدنویسی می‌کند. این عمل ممکن است توسط آزمایش مرزهای یکپارچه که در بخش ۴-۴ بحث کردیم تسهیل شود.

• اتمیک: فرض کنید درست پیش از اجرای تراکنش T_i مقادیر حساب های Aو B به ترتیب ۱۰۰۰و ۲۰۰۰ دلار باشد. اکنون فرض کنید در طول اجرای تراکنش T_i خطایی رخ می‌دهد که مانع می شود T_i با موفقیت اجرای تراکنش خود را تکمیل کند. علاوه بر این فرض کنید که خطا پس از عملیات نوشتن (A) اما پیش از عملیات نوشتن (B) اتفاق افتاده است. در این مورد مقادیر حساب A وB که در پایگاه داده بازتاب می‌شود ۹۵۰ و ۲۰۰۰ دلار است. سیستم ۵۰ دلار را به دلیل این خطا کم می‌کند. در کل متوجه خواهیم شد که جمع AوB به طور مکرر حفظ نمی‌شود.

بنابراین به دلیل خطا حالت سیستم به طور مکررحالت واقعی از جهانی را منعکس می کند که پایگاه داده قصد تصرف آن را دارد. ما چنین حالتی را وضعیت نامناسب می‌گوییم.ما باید چنین حالت های نامناسبی که قابل مشاهده در سیستم پایگاه داده نیستند را تضمین کنیم. اما توجه کنید که سیستم باید در بعضی نقاط در حالت نامناسب باشد. حتی اگر تراکنش T_i برای تکمیل اجرا شود. نقاطی وجود خواهدداشت که مقدار حساب A ۹۵۰ دلار و مقدار حساب B۲۰۰۰ دلار است که به طور واضح این وضعیت نامناسبی است. اما این حالت سرانجام با حالت مناسبی که مقدار حساب A ۹۵۰ دلار است و مقدار حساب B ۲۰۵۰ دلار است جایگزین می‌شود . بنابراین اگر تراکنش هرگز آغاز نشده بود یا برای تکمیل تضمین شده بود،چنین حالت نامناسبی جز در حین اجرای تراکنش قابل مشاهده نیست. این دلیل نیاز به خاصیت اتمیک است،اگر ویژگی های اتمیک موجود باشد تمام اقدامات تراکنش در پایگاه داده اجرا می‌شوند یا هیچ کدام اجرا نمی‌شوند.

نظریه اساسی که اتمیک را پشتیبانی می‌کند: سیستم پایگاه داده مسیر مقادیر قدیمی هر داده ای(روی دیسک) را حفظ می کند و تراکنش همانند آن عملیات را نوشتن می کند. این اطلاعات روی فایلی که لاگ نامیده می‌شود نوشته می‌شود. اگر تراکنش این اجرا را کامل نکند سیستم پایگاه داده مقادیر قدیمی لاگ را ذخیره می کند تا فرض شود تراکنش هرگز اجرا نشده است. مباحث بیشتر در مورد این نظریه در بخش ۴-۱۴ موجود است. تضمین خاصیت اتمیک به عهده سیستم پایگاه داده است خصوصا با بخشی از پایگاه داده کنترل می‌شود که سیستم ترمیم نامیده می‌شود و با جزئیات در بخش ۱۶ شرح داده شده است.

• پایایی: نخستین باری که اجرای تراکنش با موفقیت تکمیل و کاربری که تراکنش را آغاز کرده است متوجه می‌شود انتقال وجه انجام شده است،در این مورد است که هیچ خرابی سیستمی منجر به از دست دادن اطلاعات مربوط به تراکنش نمی‌شود. ویژگی پایایی تضمین می کند زمانی که تراکنش با موفقیت کامل شد حتی اگر پس از اجرای کامل تراکنش خطایی رخ دهدتمام به روز رسانی ها در پایگاه داده ادامه دارد.

اکنون فرض می کنیم که خطا ی یک سیستم کامپیوتری ممکن است باعث از دست رفتن داده در حافظه اصلی شود اما داده ای که روی دیسک نوشتن می‌شود هرگز از بین نمی‌رود. در فصل ۱۶ در مورد حفاظت دربرابر از بین رفتن داده بحث می‌کنیم.مجددا پایایی را با این دو تضمین میکنیم:

۱- به روز رسانی های انجام شده توسط تراکنش هایی که پیش از پایان تراکنش روی دیسک نوشتن می‌شوند.
۲- اطلاعاتی در مورد به روزرسانی های انجام شده توسط تراکنش و نوشتن روی دیسک کافی است تا پایگاه داده را برای به روزرسانی مجد در زمانی که سیستم پایگاه داده پس از خطا دوباره راه اندازی شده است فعال کند.

سیستم ترمیم پایگاه داده در فصل ۱۶ شرح داده شده است و علاوه بر این ویژگی اتمیک نهایی را تضمین می‌کند.

• ایزولاسیون :حتی اگر ویژگی های ثبات و اتمیک برای هر تراکنش تضمین شوند چنانچه چندین تراکنش همزمان اجرا شوند عملیات آنها ممکن است در یک حالت نامناسب باقی بماند.

به عنوان مثال همانگونه که پیش از این مشاهده کردیم زمانی که تراکنش انتقال داده از A به B با کل کسورات نوشته شده به A و افزایش جمع کل به B در حال اجرا است پایگاه داده به طور موقت در وضعیت نامناسبی قرار می گیرد. چنانچه دومین تراکنش در حال اجرای همزمان AوB را در این نقطه میانی بخواند و A+B را محاسبه کند شاهد مقدار نامناسبی خواهد بود. علاوه بر این اگر این دومین تراکنش سپس به روز رسانی های AوB بر مبنای مقادیر نامتناقضی که می‌خواند انجام دهد،پایگاه داده ممکن است در حالت نامتناقض باقی بماند حتی اگر تراکنش ها کامل شود.

روشی برای جلوگیری از این مسئله اجرای تراکنش های هم زمان، اجرای تراکنش ها به صورت سری است(یکی پس از دیگری). اما اجرای همزمان تراکنش ها هزینه عملیاتی چشمگیری دارد که در فصل ۵-۱۴ مشاهده خواهیم کرد. بنابراین راه حل های دیگر توسعه می یابند که اجرای همزمان تراکنش های چندگانه را امکان پذیر می‌کنند.

در مورد مشکلاتی که علت آنها اجرای همزمان تراکنش ها است در فصل ۵-۱۴ بحث می‌شود. ویژگی اتمیک تراکنش ، اجرای هم زمان نتایج تراکنش ها در حالتی تضمین می‌کند که اجرای تراکنش ها در یک زمان به چندین روش انجام می‌شود. درمورد ایزولاسیون بیشتر در بخش ۶-۱۴ بحث خواهیم کرد. تضمین ویژگی ایزولاسیون توسط بخشی از سیستم پایگاه داده است که سیستم کنترل همزمانی نامیده می‌شود و در فصل ۱۵ بیشتر در مورد آن بحث خواهیم کرد.

۳-۱۴ ساختار ذخیره سازی

به منظور فهم این موضوع که چگونه ویژگی های پایایی و اتمیک یک تراکنش را تضمین می‌کنند، چگونگی ذخیره سازی داده ها در پایگاه داده بطوری که قابل دسترس باشند را مورد بررسی قرار می‌دهیم.

در فصل ۱۰ رسانه ذخیره سازی را مشاهده کردیم که با سرعت نسبی،ظرفیت و انعطاف پذیری متمایز شدند و به عنوان ذخیره سازی فرار یا غیر فرار طبقه بندی می‌شوند. ما این اصطلاح ها را مرور کردیم و طبقه دیگری از ذخیره سازی، که ذخیره سازی پایدار نامیده می شود را معرفی می‌کنیم.

• **ذخیره سازی فرار**:اطلاعات موجود در ذخیره سازی فرار معمولا با خرابی سیستم از بین می‌روند .مثال های اینگونه ذخیره سازی هایی عمدتا حافظه اصلی و حافظه پنهانی است. دسترسی به ذخیره سازی فرار به دلیل سرعت و دسترسی حافظه و همچنین به این دلیل دسترسی مستقیم به داده ها در ذخیره سازی فرار به طور بی نهایت سریع است.

• **ذخیره سازی غیرفرار**: اطلاعات موجود در ذخیره سازی غیرفرار با خرابی سیستم از بین نمی‌روند. مثال هایی از ذخیره سازی های غیر فرار عبارت است از وسایل ذخیره سازی ثانویه مانند دیسک مغناطیسی و فلش که برای ذخیره سازی آنلاین استفاده می-شود و سومین ابزار ذخیره سازی مانند رسانه بصری و نوارهای مغناطیسی که برای ذخیره سازی بایگانی استفاده می‌شود. در وضعیت کنونی فناوری،ذخیره سازی غیرفرار خصوصا برای دسترسی تصادفی کندتر از ذخیره سازی فرار است. اما دومین و سومین ابزار ذخیره سازی برای خطایی که ممکن است باعث پاک شدن داده شود حساس است.

• **ذخیره سازی پایدار**: اطلاعات موجود در ذخیره سازی پایدار هرگز پاک نمی‌شود. به عنوان مثال در صورت بروز بلایای طبیعی (اگرچه خیلی بعید است اما احتمال دارد که یک سوراخ سیاه دور زمین را فرا گیرد و به طور موقت همه داده ها را نابود کند.)اگرچه به دست آوردن ذخیره پایدار از لحاظ نظری غیرممکن است اما می توان آن را تا حدودی با تکنیک هایی که بعید به نظر می‌رسد باعث از دست رفتن داده شوند تخمین زد. به منظور اجرای ذخیره سازی پایدار،اطلاعاتی را در چندین رسانه ذخیره سازی غیر فرار (معمولا دیسک) با حالت های خطای مستقل تکرار کردیم. بروز رسانی ها باید با احتیاط انجام شود تا مطمئن شویم خطایی که در طول یک بروز رسانی رخ می‌دهد باعث از دست رفتن داده نمی‌شود. بخش ۱۶-۲-۱ در مورد ثبات اجرای ذخیره سازی بحث می‌کند.تمایزهای میان انواع ذخیره سازی های مختلف نسبت به اجرای عملی وضوح کمتری دارد. به عنوان مثال برخی سیستم ها مانند کنترل کننده های RAID نسخه ای پشتیبان از باطری ارائه می‌کنند تا حافظه اصلی بتواند مانع از خطاهای نیرو یا خرابی های سیستم شود.

برای اینکه یک تراکنش پایدار باشد،تغییرات آن باید با ذخیره سازی پایدار نوشتن شود. به همین نحو برای اینکه یک تراکنش تجزیه ناپذیر باشد سوابق در لاگ باید ثبت شود تا ذخیره سازی پایدار پیش از هر تغییری روی دیسک در پایگاه داده انجام شود. به طور واضح، عواملی که پایایی واتمیک بودن داده های یک سیستم را تضمین می کند بستگی به این موضوع دارد که چگونه اجرای آن در ذخیره سازی پایدار واقعا ثابت است. در برخی موارد تنها یک کپی روی دیسک کافی به نظر می‌رسد،اما کدام نرم افزارها بسیار ارزشمند هستند و کدام تراکنش ها بسیار مهم هستند که چندین کپی یا به عبارت دیگر تخمین دقیق تر مفهوم ایده آل از ذخیره سازی پایدار را نیاز دارند.

۴-۱۴ پایایی و ظرفیت تراکنش

همانگونه که پیش از این گفته شد ، یک تراکنش ممکن نیست همیشه با موفقیت اجرا شود. چنین تراکنشی بی‌نتیجه نامیده می شود. وقتی ویژگی اتمیک را تضمین می کنیم، یک تراکنش بی نتیجه نباید هیچ تاثیری بر وضعیت پایگاه داده داشته باشد. بنابراین تمامی تغییراتی که تراکنش متوقف شده انجام داده در پایگاه داده ناتمام می‌ماند. نخستین باری که تغییرات به علت تراکنش متوقف شده و ناتمام می‌ماند گفته می شود که تراکنش Rollback کرده است. بخش ترمیم سیستم پایگاه داده برای مدیریت تراکنش های بی نتیجه مانده معمولا با ثبت در یک لاگ انجام می شود. اصلاح هرپایگاه داده که با یک تراکنش انجام می‌شود که برای مرتبه اول در لاگ ثبت می‌شود. ما شناسه اصلاح اجرای تراکنش،شناسه داده اصلاح شده و هر دو مقدار قدیمی (پیش از اصلاح) و مقدار جدید(پس از اصلاح) از داده را ثبت می‌کنیم و تنها در این صورت پایگاه داده اصلاح می‌شود. حفظ یک لاگ امکان اجرای مجدد و تغییر را فراهم می‌کند تا ویژگی اتمیک و پایانی را تضمین کند و همچنین اگر در حین اجرا خطایی رخ دهد ویژگی اتمیک را تضمین می‌کند. در فصل ۱۶ در مورد جزئیات اصلاح لاگ بحث شده است.

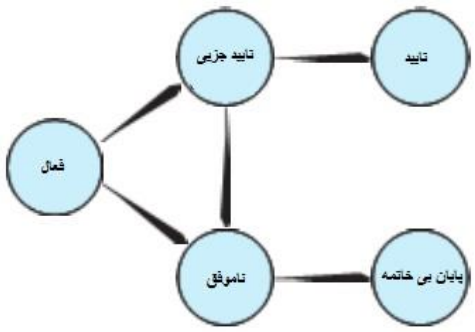
تراکنشی که اجرایش را با موفقیت به انجام رسانده است commit نامیده می‌شود. یک تراکنش commit شده که بروزرسانی ها را انجام داده و پایگاه داده را به وضعیت منسجم جدیدی تبدیل می‌کند حتی زمانی که در سیستم خطایی رخ دهد ،تاثیری روی این تراکنش نخواهد داشت.

وقتی یک تراکنش انجام شد نمی‌توانیم اثرات آن را نادیده بگیریم. تنها راه نادیده گرفتن تاثیرات یک تراکنش commit شده اجرای تراکنشی جبرانی است. به عنوان مثال اگر تراکنشی ۲۰ دلار به حساب اضافه کند تراکنش جبرانی ۲۰ دلار از حساب کم می‌کند. اما همچنین ایجاد چنین تراکنش جبرانی همیشه ممکن نیست. بنابراین مسئولیت نوشتن و اجرای یک تراکنش جبرانی با

کاربر است و سیستم پایگاه داده کنترل نمی‌شود. فصل ۲۶ شامل مبحث تراکنش های جبرانی است. ما باید در مورد اجرای با موفقیت یک تراکنش دقیق تر باشیم. بنابراین یک مدل تراکنش انتزاعی ساده ایجاد می‌کنیم. یک تراکنش باید مانند یکی از حالت های زیر باشد:

- فعال: حالت اولیه،زمانی که تراکنش اجرا می‌شود در این حالت باقی می ماند.
- تایید جزیی:اجرای دستورات کامل شده ولی نتیجه آن نوشتن نشده.
- ناموفق: درصورت بروز خطا در وضعیت های قبلی به این حالت میرود.
- پایان بی خاتمه: پس از اینکه تراکنش rollback شد ،پایگاه داده مجدد به حالت اول ذخیره سازی می شود تا یک تراکنش را آغاز کند.

نمودار وضعیت به همراه تراکنش در شکل ۱-۱۴ آمده است. زمانی که تراکنش وارد حالت تایید جزیی می‌شود می گویند تراکنش انجام شده است.



شکل ۱ – ۱۴ نمودار وضعیت تراکنش

به همین نحو زمانی یک تراکنش دچار پایان بی خاتمه می‌شود که وارد وضعیت بی نتیجه ای شده باشد.زمانی که یک تراکنش در وضعیت پایان بی نتیجه نباشد منقطع نامیده می‌شود.

تراکنش در حالت فعال آغاز می‌شود. زمانی که حالت نهایی خود را تمام می‌کند تا حدودی وارد مرحله تبادل می‌شود.در این نقطه اجرای آن کامل شده است اما همچنان این احتمال وجود دارد که بی نتیجه بماند زیرا ممکن است خروجی واقعی همچنان به طور موقت در حافظه اصلی باقی مانده باشد بنابراین ممکن است خطای سخت افزاری مانع از اتمام موفقیت آمیز آن شود.

سپس سیستم پایگاه داده اطلاعات کافی را حتی در زمان وقوع یک خطا روی دیسک نوشتن می‌کند بروزرسانی ها با تراکنش هایی انجام می‌شود که می توانند پس از راه اندازی دوباره سیستم یا پس از خطا تولید شوند و وقتی آخرین اطلاعات ثبت می‌شود تراکنش وارد حالت شروع بکار می‌شود.همان طور که پیش از این گفته شد،فرض بر این است که خطاها باعث پاک شدن اطلاعات روی دیسک نمی‌شوند. فصل ۱۶ در مورد تکنیک‌های مقابله با از دست دادن داده ها روی دیسک بحث می‌کند.

یک تراکنش پس از اینکه سیستم تعیین کرد که نمی تواند به اجرای عادی خود ادامه دهد وارد حالت خطا می‌شود(به عنوان مثال به دلیل خطاهای سخت افزاری یا منطقی). چنین تراکنشی باید rollbackشود. سپس وارد حالت abort شود. این نقطه سیستم دو گزینه دارد:

- می تواند مجددا تراکنش را اجرا کند اما تنها زمانی که تراکنش به عنوان نتیجه ای از خطاهای سخت افزاری یا نرم افزاری بی نتیجه مانده است .تراکنشی با راه اندازی مجدد به عنوان یک تراکنش جدید در نظر گرفته میشود.
- می تواند تراکنش را از بین ببرد. معمولا به این دلیل انجام می‌شود که بعضی خطاهای منطقی داخلی تنها با بازنویسی برنامه نرم افزار اصلاح می‌شوند یا به این دلیل که ورودی مناسب نبوده است و یا اینکه داده مطلوبی در پایگاه داده یافت نشده است. زمانی که با نوشته های خارجی قابل مشاهده ای مانند نوشتن روی صفحه کاربری یا ارسال ایمیل کار می-کنیم باید احتیاط کنیم. زمانی که نوشتن می‌کنیم میتوانیم آن را پاک کنیم زیرا ممکن است برای سیستم پایگاه داده خارجی به نظر برسد.

اکثر سیستم ها چنین نوشتن هایی را تنها پس از اینکه تراکنش ها وارد حالت شروع به انجام کار می شوند فعال می‌کنند. روشی برای اجرای چنین طرحی در سیستم پایگاه داده برای ذخیره هر مقدار مرتبط با چنین نوشتن های خارجی به طور موقت در رابطه ویژه ای در پایگاه داده هستند و انجام و نوشتن واقعی را تنها زمانی که تراکنش وارد مرحله شروع به انجام کار می‌شود انجام می-دهند. اگر سیستم پس از اینکه وارد مرحله شروع به کار شد خطا رخ دهد ولی پیش از آن بتواند نوشتن خارجی را تکمیل کند، درصورتی که سیستم دوباره شروع به کار می‌کند سیستم پایگاه داده نوشتن خارجی را انجام می‌دهد.(با استفاده از داده های ذخیره سازی غیرفرار).

کنترل نوشتن های خارجی می تواند در بعضی شرایط پیچیده تر باشد. به عنوان مثال فرض کنید عمل خارجی گرفتن پول نقد از خودپرداز است و پیش از اینکه پول نقد توسط دستگاه به مشتری داده شود سیستم خطا می‌دهد(فرض کنیم که پول نقد می تواند به طور خودکار داده شود.)

این باعث می‌شود وقتی سیستم دوباره راه اندازی شد هیچ واکنشی به دادن پول نقد نداشته باشد زیرا ممکن است کاربر دستگاه خودپرداز را ترک کرده باشد. در چنین مواردی تراکنش جبرانی مانند واریز پول نقد به حساب کاربر بازمی گرددو زمانی که سیستم مجددا راه اندازی می شود اجرا می‌شود.

مثال دیگر،فرض کنید یک کاربر درسایتی رزرو انجام می‌دهد ممکن است که سیستم پایگاه داده یا سرور نرم افزار پس از اینکه تراکنش رزرو انجام شد با خطا مواجه شود. همچنین ممکن است اتصال شبکه کاربر پس از اینکه تراکنش رزرو انجام شد قطع شود. در مورد دیگر اگرچه حتی تراکنش انجام شده است،نوشتن خارجی انجام نشده است. برای کنترل چنین موقعیت هایی نرم افزاری باید طراحی شود تا زمانی که کاربر مجددا به نرم افزار وب متصل می‌شود بتواند مشاهده کند که آیا تراکنش او موفق بوده است یا با خطا مواجه شده است. نرم افزارهایی خاص می‌توانند به کاربران اجازه دهند داده ها را مشاهده کنند، خصوصا برای مدت زمان طولانی که ساعت ها یا دقایقی تراکنش انجام می‌شود. متاسفانه،نمی توانیم چنین خروجی هایی را داده قابل مشاهده ای کنیم مگر اینکه بخواهیم ظرفیت تراکنش را هم تراز کنیم. در فصل ۲۶ در مورد مدل های تراکنش جایگزین صحبت کرده ایم که تراکنش های غیرفعال طولانی مدت را پشتیبانی می‌کند.

۵-۱۴ ایزولاسیون تراکنش

سیستم های پردازشی تراکنش معمولا اجرای صحیح تراکنش های چندگانه را امکان پذیر می‌کنند و باعث می‌شود تراکنش های چندگانه به طور همزمان داده ها را بروزرسانی کند و همان طور که پیش از این دیدیم باعث تضمین ثبات داده می‌شود. تضمین ثبات بر خلاف اجرای هم زمان تراکنش ها نیازمند کار زیادی است، اجرای تراکنش ها به صورت سری به مراتب ساده تر است، به این معنا که یکی یکی، هرکدام از تراکنش ها پس از اینکه تراکنش قبلی کامل شد آغاز می‌شود. اما دو دلیل خوب برای همزمانی زیر وجود دارد:

• **بهبود توان و بهره وری منابع**:یک تراکنش از چندین مرحله تشکیل شده است. بعضی مراحل شامل عملI/Q میشود،مراحل دیگر شامل CPU می‌شود. CPUو دیسک ها در سیستم کامپیوتر می توانند به موازات هم عمل کنند.بنابراین عمل I/Q می تواند به موازات با فرآیند CPUانجام شود. از تطابق CPU و سیستم I/Q بنابراین می توان تراکنش های متعدد را به صورت موازی اجرا کرد. در حالیکه خواندن یا نوشتن در نیمی از یک تراکنش روی دیسک در حال پردازش است تراکنش دیگر می‌تواند در CPU اجرا شود در حالیکه دیسک دیگر ممکن است نوشتن یا خواندن را در نیمه سوم تراکنش انجام دهد.تمام این افزایش توان سیستم که تعداد تراکنش های اجرا شده در بازه زمانی مشخصی است متقابلا پردازشگر و استفاده از دیسک نیز افزایش می‌یابد،به عبارت دیگر پردازشگر و دیسک زمان بیکاری کمتری صرف می‌کنند یا هیچ کار مفیدی انجام نمی‌دهند.

• **کاهش زمان نوشتن**: ممکن است ترکیبی از تراکنش های در حال اجرا کوتاه یا بلند روی سیستم وجود داشته باشد. اگر تراکنش ها به طور سری اجرا شوند،یک تراکنش کوتاه باید صبر کند تا پردازش تراکنش طولانی کامل شود که می‌تواند به تاخیرهای غیرقابل پیش بینی در اجرای یک تراکنش منجر شود. چنانچه تراکنش ها در حال انجام در قسمت های مختلف پایگاه داده هستند ،بهتر است اجازه دهیم آنها به طور همزمان اجرا شوند،فرایند های CPU و دسترسی دیسک را میان آنها تقسیم می-کنیم. اجراهای همزمان تاخیرهای غیرقابل پیش بینی در تراکنش های در حال اجرا را کاهش می‌دهد علاوه بر این همچنین میانگین زمان پاسخگویی را هم کاهش می‌دهد: زمان میانگین برای یک تراکنش برای کامل شدن پس از اجرای آن است.

انگیزه برای استفاده از اجرای همزمان در یک پایگاه داده الزاما همانند انگیزه استفاده از چند برنامگی در سیستم عامل است.

وقتی چندین تراکنش به طور همزمان اجرا می‌شود ویژگی ایزولاسیون ممکن است ناقص شود،در نتیجه، ثبات پایگاه داده برخلاف صحت هر تراکنش فردی از بین می‌رود. در این بخش ما مفهوم زمانبندی را ارائه می‌دهیم تا به اجراهایی که ویژگی ایزولاسیون را تضمین و به ثبات پایگاه داده کمک می‌کنند را شناسایی کنیم. سیستم پایگاه داده باید فعل و انفعال میان تراکنش های هم زمان را کنترل کند تا مانع از نابودی ثبات پایگاه داده شود. این کار از طریق انواعی از مکانیسم ها انجام می‌شود که روش های کنترل همزمانی نامیده می‌شود. روش های کنترل همزمانی را در فصل ۱۵ مطالعه می‌کنیم،اکنون بر مفهوم اجرای همزمان صحیح تمرکز می‌کنیم. مجددا سیستم بانکداری ساده شده بخش ۱-۱۴ را در نظر بگیرید که چندین صورتحساب دارد و مجموعه ای از تراکنش ها که به این حساب ها دسترسی دارند و آنها را به روز می‌کنند T1وT2 دو تراکنشی هستند که سرمایه را از یک حساب به حساب دیگر انتقال می‌دهند. تراکنش T1 ۵۰ دلار از حساب A به حساب B انتقال می‌دهد و اینگونه تعریف می‌شود:

```
T1 :read(A);
    A:=A−50;
    write(A);
    read(B);
    B:=B+ 50;
    write(B).
```

تراکنش T2 ۱۰ درصد از تراز را از حساب A به حساب B انتقال می‌دهد و اینگونه تعریف می‌شود:

```
T2 :read(A);
    temp:=A* 0.1;
    A:=A−temp;
    write(A);
    read(B);
    B:=B+temp;
    write(B).
```

روش های موازی در همزمانی

چندین روش موازی همزمان در زمینه محاسبه افزایش می یابند تا مقدار همزمانی های ممکن را افزایش دهند. همچنان که سیستم ها ی پایگاه داده این همزمانی را کاوش می کنند تا عملکرد کل سیستم را افزایش دهند ،شمار چشمگیری از تراکنش ها به طور همزمان در حال اجرا خواهد بود:

کامپیوترهای اولیه تنها یک پردازشگر داشتند. بنابراین هیچ همزمانی واقعی در این کامپیوتر وجود نداشت. تنها همزمانی موجود همزمانی ظاهری ایجاد شده توسط سیستم عامل بود،همچنان که پردازشگر در میان چندین کار یا فرآیند های سخت تقسیم شده بود. کامپیوترهای پیشرفته احتمالا چندین پردازشگر دارند.این ها ممکن است به درستی همه بخش های پردازشگر یک کامپیوتر را جدا کنند. اما حتی یک پردازشگر مجزا هم می تواند در یک زمان با داشتن چندین هسته فرآیندهای بیشتری را اجرا کند. پردازنده INTEL CORE DUO مثال معروفی از یک پردازشگر چند هسته ای است. برای سیستم های پایگاه داده از چندین پردازشگر و هسته انتخاب شدند. یک روش کشف وجه تشابه در یک تراکنش واحد یا جستجو بود روش دیگر پشتیبانی از تعداد زیادی تراکنش های همزمان است.

بسیاری از ارائه دهندگان خدمات اکنون از کلکسیون های عظیم کامپیوتری به جای رایانه های بزرگ استفاده می کنند تا خدمات خود را ارائه دهند. آنها این انتخاب را بر مبنای هزینه کم این روش انجام می‌دهند. یکی از نتایج این انتخاب افزایش خواندنه همزمانی است که می‌تواند پشتیبانی شود.

این نکات به متن هایی اشاره می‌کند که این پیشرفت ها ساختار کامپیوتر و همسان سازی کامپیوتری را شرح می‌دهد. فصل ۱۸ الگوریتمی برای ساخت سیستم های پایگاه داده مشابه شرح می‌دهد که پردازشگرهای چندگانه و هسته های چندگانه را پیاده سازی می‌کند.

T_2 :read(A);
$temp := A * 0.1$;
$A := A - temp$;
write(A);
read(B);
$B := B + temp$;
write(B).

فرض کنید مقادیر جریان حساب Aو B به ترتیب ۱۰۰۰دلار و ۲۰۰۰ دلار همچنین دو تراکنش T1و T2 به ترتیب انجام می‌شوند. این ترتیب اجرا در شکل ۲-۱۴ آمده است. در شکل توالی مراحل دستورالعمل ترتیب زمانی از بالا به پایین است. دستورالعمل T1 در ستون چپ و دستورالعمل T2 در ستون راست موجود می باشد. مقادیر نهایی حساب Aو B پس از اجرا در شکل ۲-۱۴ آمده است که به ترتیب ۸۵۵ دلار و ۲۱۴۵ دلار هستند بنابراین مقدار نهایی پول در حساب AوB پس از اجرای هر دو تراکنش جمع A+B است.

T_1	T_2
read(A)	
$A := A - 50$	
write(A)	
read(B)	
$B := B + 50$	
write(B)	
commit	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
	read(B)
	$B := B + temp$
	write(B)
	commit

شکل ۲-۱۴ زمانبندی-زمانبندی که T2پس از T1انجام می‌شود.

به همین نحو اگر تراکنش ها در زمانی انجام شوند که T2 پس از T1باشد سپس ترتیب اجرا مشابه شکل ۳-۱۴ است. مجددا انتظار می رود جمع A+B حفظ شود و مقدار نهایی حساب AوB به ترتیب ۸۵۰ و۲۱۵۰ دلار است.

T_1	T_2
	read(A)
	$temp$:=A∗0.1
	A:=A−$temp$
	write(A)
	read(B)
	B:=B+$temp$
	write(B)
	commit
read(A)	
A:=A−50	
write(A)	
read(B)	
B:=B+50	
write(B)	
commit	

زمانبندی ۲-زمانبندی که T1پس از T2 انجام می‌شود.

ترتیب اجرا درست همانگونه که شرح داده شد برنامه زمان بندی نامیده می‌شود. آنها براساس زمانبندی دستورها در سیستم اجرا و ارائه می‌شوند. به طور واضح یک برنامه زمان بندی برای گروهی از تراکنش ها باید تمام تراکنش ها را دربرگیرد و باید ترتیب که دستورها در هر تراکنش فردی ظاهر می‌شوند را حفظ کند. به عنوان مثال در تراکنش T1،در هر تراکنش معتبری دستور نوشتن A باید پیش از دستور خواندن B ظاهر شود. توجه داشته باشید که در جدول زمان بندی ما عملیاتی وارد کار می‌شوند تا نشان دهند تراکنش وارد حالت شروع بکار شده است. در ادامه بحث به اجرای اول برنامه زمان بندی ۱ و دومین اجرای جدول زمان بندی ۲ اشاره می‌کنیم.

این جدول های زمان بندی سری هستند: هر برنامه زمان بندی سری شامل ترتیبی از دستورهای تراکنش های مختلف می‌شود جایی که دستورها به یک تراکنش واحد تعلق دارند که باهم در جدول زمان بندی نمایان می‌شوند. فرمول ترکیب را به خاطر آورید،یادآوری می‌کنیم که برای مجموعه ای از تراکنش ها تعداد N فاکتوریل در جدول زمان بندی معتبر مختلف سری وجود دارد.

زمانی که سیستم پایگاه داده چندین تراکنش را همزمان اجرا می‌کند نیازی نیست تا برنامه زمان بندی مربوطه سری باشد. اگر دو تراکنش همزمان اجرا شوند سیستم عامل ممکن است یک تراکنش را کمی انجام دهد و سپس زمینه تغییر کند و برای مدتی تراکنش دوم اجرا شود و سپس اولین تراکنش انجام شودوبه همین ترتیب تراکنش ادامه یابد. با تراکنش های متعدد زمان CPU در میان همه تراکنش ها تقسیم می‌شود.

اجرای چندین تراکنش متوالی امکان پذیر است زیرا دستورهای مختلف هردو تراکنش ممکن است اکنون در میان یکدیگر قرار گیرند.در مجموع ممکن نیست دقیقا پیش بینی کنید چند دستور از تراکنش پیش از اینکه CPU به تراکنش دیگر تغییر کند اجرا خواهد شد.

به مثال قبل بازمی‌گردیم،فرض کنید دو تراکنش به طور همزمان اجرا می‌شوند. یکی از زمانبندی های ممکن در شکل ۴-۱۴ آمده است. پس از این که اجرا انجام شد به همان حالتی می‌رسیم که یک تراکنش سری اجرا می‌شود و T2پس از T1اجرا می‌شود. مجموع A+B باید حفظ شود.۰۰۰

نتیجه اجرای همه تراکنش ها صحیح نیست. با مثال نشان می‌دهیم،زمانبندی شکل ۵-۱۴ را در نظر بگیرید. پس از اجرای این برنامه زمان بندی به حالتی می‌رسیم که مقادیر نهایی حساب های AوB به ترتیب ۹۵۰ و۲۱۰۰ دلار می باشد. حالت نهایی حالت متناقضی است زیرا ما ۵۰دلار در فرآیند اجرای همزمان به دست آورده ایم. در مقابل جمع A+B با اجرای دو تراکنش حفظ نشده است.

چنانچه کنترل همزمان به طور کلی در سیستم عامل بماند چندین زمانبندی ممکن از جمله تراکنشی که در یک حالت متناقض در پایگاه داده باقی مانده است مانند نمونه ای که شرح داده شد امکان پذیر شود. این کار سیستم پایگاه داده است تا تضمین کند هر برنامه زمان بندی که درحال اجرا باشد پایگاه داده در یک حالت متناقض باقی خواهد ماند. بخش کنترل همزمانی سیستم پایگاه داده این کار را انجام می‌دهد.

T_1	T_2
read(A)	
A:=A−50	
write(A)	
	read(A)
	$temp$:=A∗0.1
	A:=A−$temp$
	write(A)
read(B)	
B:=B+50	
write(B)	
commit	
	read(B)
	B:=B+$temp$
	write(B)
	commit

شکل ۴-۱۴ برنامه زمانبندی ۳-معادل برنامه زمان بندی همزمان با برنامه زمان بندی۱

می‌توانیم ثبات پایگاه داده را تحت اجرای همزمان تضمین کنیم از این طریق که اطمینان حاصل کنیم هر برنامه زمان بندی که اجرا می‌شود همان تاثیری را دارد که می تواند بدون یک اجرای همزمان انجام شود. به این ترتیب از برخی جهات باید مشابه با برنامه زمان بندی سری باشد. چنین برنامه های زمان بندی سریال پذیر نامیده می‌شوند.

T_1	T_2
read(A)	
A:=A−50	
	read(A)
	$temp$:=A∗0.1
	A:=A−$temp$
	write(A)
	read(B)
write(A)	
read(B)	
B:=B+50	
write(B)	
commit	
	B:=B+$temp$
	write(B)
	commit

شکل ۵-۱۴ برنامه زمانبندی ۴- یک زمانبندی همزمان ناشی از یک حالت نامتناقض

۶-۱۴ قابلیت سریال پذیری

پیش از اینکه بتوانیم بررسی کنیم چگونه عنصر کنترل همزمانی سیستم پایگاه داده می‌تواند قابلیت سریال پذیری را تضمین کند، سریال پذیری جدول زمانبندی را بررسی می‌کنیم که چگونه که یک جدول زمانبندی سریال پذیر است.قطعا برنامه های زمانی سری، سریال پذیر هستند، اما اگر مراحل تراکنش های چندگانه دچار تداخل شوند تعیین اینکه یک زمانبندی سریال پذیر می-باشد یا نه، تعیین دقیق اینکه یک تراکنش چه عملکردهایی انجام می‌دهد و چگونه عملکردهای تراکنش های مختلف با یکدیگر فعل و انفعال داخلی دارند دشوار است.به این دلیل ما انواع مختلفی از عملکردهایی که یک تراکنش می‌تواند بروی یک داده انجام دهد را بررسی نمی‌کنیم اما در مقابل دو تراکنش را بررسی می‌کنیم:خواندن و نوشتن. تصور می‌کنیم که میان دستور خواندن و نوشتن بروی داده یک تراکنش ممکن است ترتیب اختیاری از عملکردها بر کپی Q انجام دهد که در حافظه کمکی محلی تراکنش مستقر است. در این مدل،تنها عملیات تراکنش از نقطه نظر زمانبندی دستور خواندن و نوشتن است. اگرچه عملیات وارد عمل شده مرتبط هستند اما تا بخش ۷-۱۴ مورد بررسی قرار نگرفته اند. ما بنابراین دستورهای خواندن و نوشتن را در زمانبندی نشان می-دهیم،همان طور که در زمانبند ۳ در شکل ۶-۱۴ مشاهده کردیم.

در این بخش ما حالت های مختلفی از تعادل زمانبندی را مورد بحث قرار می‌دهیم اما بر نوع خاصی تمرکز می‌کنیم که سریال ناپذیر نامیده می‌شود.

باهم زمانبندی S را بررسی می‌کنیم که دو دستور متوالی دارد،اول از تراکنش های TiوTj به ترتیب (i=j). چنانچه i وj به آیتم های داده مختلف اشاره می‌کند،سپس می توانیم iو j را بدون تاثیر نتایج هر دستوری در جدول زمانبندی معاوضه کنیم. اما اگر i وj به همان داده Q اشاره کند سپس ترتیب دو مرحله ممکن است اهمیت داشته باشد. به دلیل اینکه تنها با دستورهای خواندن و نوشتن سروکار داریم چهار گزینه موجود است که باید بررسی کنیم:

۱- i=خواندن،j=خواندن،ترتیب iوj اهمیتی ندارد زیرا همان مقدار از Q توسط TiوTj صرف نظر از توالی آن را خواندن می کند.

۲- i=خواندن ،j=نوشتن : اگر i پیش از j بیاید سپس Ti مقدار Q که با Tj دردستور نوشته شده است را نوشتن نمی‌کند. چنانچه j پیش از i بیاید سپسTi مقدار Q که توسط Tj نوشته شده است را خواندن می‌کند. بنابراین ترتیب i وj اهمیت دارد.

T_1	T_2
read(A)	
write(A)	
	read(A)
	write(A)
read(B)	
write(B)	
	read(B)
	write(B)

شکل ۶-۱۴ زمانبندی ۳ دستورات خواندن و نوشتن را نشان می‌دهد.

T_1	T_2
read(A)	
write(A)	
	read(A)
read(B)	
	write(A)
write(B)	
	read(B)
	write(B)

شکل ۷-۱۴ زمانبندی ۵- زمانبندی ۳ پس از مبادله دو تراکنش

۳- i=نوشتن (Q)،j= خواندن (Q). ترتیب i وj به دلیل شباهت به مورد قبلی اهمیت دارد.

۴- i=نوشتن(Q)، j= نوشتن(Q). به دلیل اینکه هردو دستور نوشتن هستند ترتیب این دستورها بر TiیاTj تاثیر گذار نمی باشد. اما مقدار به دست آمده از دستور خواندن بعدی (Q) از S تحت تاثیر قرار می‌گیرد زیرا نتیجه دو دستور نوشتن بعدی در پایگاه داده حفظ می‌شود. اگر هیچ دستور نوشتن دیگری پس از اولJ در S وجود نداشته باشد سپس ترتیب i وj به طور مستقیم بر مقدار نهایی Q در حالت پایگاه داده که نتیجه زمانبندی S است تاثیر می‌گذارد.

بنابراین،تنها درموردی که هردو دستورi وjخواندن می‌باشد ترتیب نسبی اجرای آنها اهمیتی ندارد.

چنانچه i وj عملیاتی از تراکنش های مختلف در همان اقلام داده باشند و حداقل یکی از این عملیات خواندن باشد i وj متضاد هستند.

برای نشان دادن مفهوم دستورهای متضاد زمانبندی ۳ در شکل ۶-۱۴ را بررسی می‌کنیم. دستور نوشتن (A) از T1 با دستور خواندن(A) از T2 متضاد است. اما دستور خواندن (A) از T2 با دستور دخواندن(B) از T1 متضاد نیست زیرا دو دستور به آیتم های داده مختلفی دسترسی دارند.

T_1	T_2
read(A)	
write(A)	
read(B)	
write(B)	
	read(A)
	write(A)
	read(B)
	write(B)

شکل ۸-۱۴ زمانبندی ۶-زمانبندی سریالی که معادل با زمانبندی ۳ است.

T_3	T_4
read(Q)	
	write(Q)
write(Q)	

شکل ۹-۱۴ زمانبندی ۷

i و j دستورالعمل های متوالی زمانبندی S می‌باشند.اگر اوJ دستورالعمل های تراکنش های مختلف باشند و متضاد نباشند بنابراین می توانیم ترتیب i و j را تغییر دهیم تا زمانبندی S جدیدی ایجاد کنیم.S معادل با همان S است،زیرا همه دستورالعمل ها به همان ترتیب در هردو زمانبندی به جز i وj موجود است که ترتیب اهمیت ندارد.

به دلیل اینکه دستور نوشتن (A) از T2 در زمانبندی ۳ شکل ۶-۱۴ با دستور خواندن(B) از T1 متضاد نمی‌باشد می توانیم این دستورالعمل ها را تغییر دهیم تا زمانبندی معادل با زمانبندی ۵ که در شکل ۷-۱۴ نشان داده شد را تولید کنیم . صرف نظر از حالت اولیه سیستم زمانبندی ۳ و ۵ هردو همان حالت نهایی سیستم را تولید می‌کنند.

ما همچنان به مبادله دستورالعمل های بدون conflict ادامه می‌دهیم:

- تعویض دستور خواندن(B) از T1 با دستور خواندن(A) از T2
- تعویض دستور نوشتن(B) از T1 با دستور نوشتن (A) از T2
- تعویض دستور نوشتن(B) از T1 با دستور نوشتن (A) از T2

نتیجه نهایی این تعویض ها،زمانبندی۶ از شکل ۸-۱۴ یک زمانبندی سریالی است. توجه داشته باشید که زمانبندی ۶ دقیقا همانند زمانبندی ۱ است،اما تنها دستور العمل های خواندن و نوشتن را نشان می‌دهد. بنابراین نشان می‌دهیم که زمانبندی ۳ معادل با زمانبندی سریالی است. این تشابه به این معنا است که،صرف نظر از حالت اولیه سیستم زمانبندی ۳ همان حالت نهایی را تولید می‌کند که بعضی برنامه های زمانی سریالی تولید خواهند کرد.

چنانچه زمانبندی S بتواند توسط یک سری از تغییرات دستورالعمل های نامتناقض به زمانبندی S تبدیل شود گفته میشود SوS معادله conflict هستند.

توجه داشته باشد که همه زمانبندی های سریالی با یکدیگر دارای برخورد هستند. به عنوان مثال زمانبندی ۱ و ۲ برخوردندارند.

مفهوم برخورد مفهوم قابلیت ناپذیری را به وجود می آورد. چنانچه برخورد همراه با زمانبندی سریالی باشد گفته می‌شود که زمانبندی S سریال ناپذیر است. بنابراین زمانبندی S سریال ناپذیر است زیرا با زمانبندی سریالی۱ ، سریال ناپذیر است.

در نهایت زمانبندی ۷ برای شکل ۹-۱۴ را در نظر بگیرید که شامل عملیات چشمگیری از تراکنش های T3وT4 است. این زمانبندی سریال ناپذیر است زیرا معادل با هیچ زمانبندی سریالی T3,T4 یا برنامه سریالی T4,T3نمی‌باشد.

شکل ۱۰-۱۴ گراف اولویت برای زمانبندی۱ و زمانبندی ۲

اکنون یک روش موثر و ساده برای تشخیص قابلیت سریال پذیری یک زمانبندی را ارائه می‌دهیم. زمانبندی s را در نظر بگیرید. ما یک نمودار دستورالعمل می‌سازیم که گراف اولویت از Sنامیده می‌شود. این گراف شامل یک جفت G=(V,E) است که V مجموعه ای از رئوس و E مجموعه ای از یال ها می باشد. مجموعه رئوس شامل همه تراکنش های موجود در زمانبندی است. مجموعه یال ها شامل همه یال از T i→T j است که یکی از سه حالت زیر اجرا میشود:

۱- Ti نوشتن (Q) را پیش از اینکه Tj خواندن را اجرا کند انجام می‌دهد.

۲- Ti خواندن را پیش از اینکه Tj نوشتن را اجرا کند انجام می‌دهد.

۳- Ti نوشتن را پیش از اینکه Tj نوشتن را اجرا کند انجام می‌دهد.

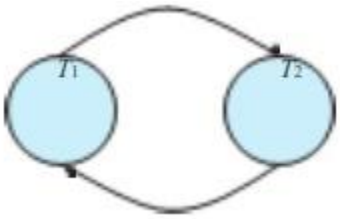
چنانچه یال T i→T در گراف اولویت وجود داشته باشد سپس در هر زمانبندی S سریالی معادل با S Ti باید پیش از Tj اجرا شود.

به عنوان مثال گراف اولویت برای زمانبندی ۱ در شکل ۱۰-۱۴ A شامل یال مجزای *T1→T2* است،زیرا همه دستورالعمل های T1 پیش ازاجرای اولین دستورالعمل T2 اجرا می‌شوند. به همین نحو شکل ۱۰-۱۴B گراف اولویت زمانبندی ۲ را با یال مجزا *T2→T1* نشان می‌دهد زیرا همه تراکنش های T2 پیش از اولین اجرای دستورالعمل T1 اجرا می‌شوند.

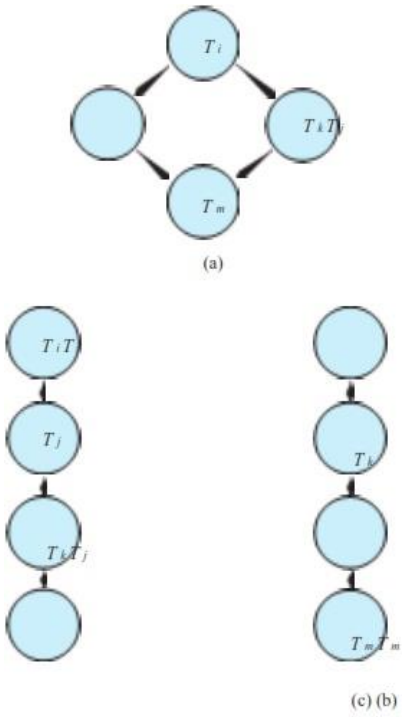
گراف اولویت برای زمانبندی ۴ در شکل ۱۱-۱۴ نشان داده شده است. نمودار شامل یال *T1→T2* زیرا T1 خواندن را پیش از نوشتن T2 انجام می‌دهد. همچنین شامل یال *T2→T1* است ، زیرا T2 خواندن را پیش از اینکه T1 نوشتن را جرا کند انجام می‌دهد.

اگر گراف اولویت برای S یک حلقه داشته باشد بنابراین زمانبندی S قابل سریال پذیری متضاد نمی‌باشد. چنانچه نمودار شامل هیچ حلقه ای نباشد بنابراین زمانبندی S قابل سریال پذیری متضاد است.

ترتیب قابلیت سریال پذیری تراکنش ها را می توان از طریق کشف یک ترتیب خطی نامتناقض با ترتیب جزئی گراف اولویت بدست آورد. این فرآیند مرتب سازی توپولوژیکی نامیده میشود. در کل،چندین ترتیب خطی وجود دارد که میتوان از طریق مرتب سازی توپولوژیکی به دست آورد.



شکل ۱۱-۱۴ گراف اولیت برای زمانبندی ۴



شکل ۱۲-۱۴ مثالی از مرتب سازی توپولوژیکی

به عنوان مثال نمودار شکل ۱۲-۱۴a دو ترتیب خطی قابل قبول دارد که در شکل ۱۲-۱۴bو ۱۲-۱۴c نشان داده شده است.

بنابراین برای آزمایش قابلیت سریال پذیری متضاد ما باید گراف اولویت طراحی کنیم و و یک الگوریتم تشخیص حلقه ایجاد کنیم. الگوریتم های تشخیص حلقه را میتوان در کتاب های استاندارد الگوریتم پیدا کرد.الگوریتم های شناسایی حلقه همانند آنهایی که بر مبنای جستجوی اول عمق هستند نیازمند ترتیب عملیات های n2 می‌باشند که n تعداد رئوس در نمودار است.(که تعداد تراکنش ها می باشد.)

به مثال های قبل باز می‌گردیم،توجه داشته باشد که گراف اولویت برای زمانبندی ۱ و ۲ (شکل ۱۰-۱۴) در واقع شامل هیچ حلقه ای نمی‌شوند. از طرف دیگر گراف اولویت برای شکل ۴ (شکل ۱۱-۱۴) شامل حلقه ای می‌شود که نشان می‌دهد این زمانبندی قابل سریال پذیری متضاد نیست.

ممکن است دو زمانبندی داشته باشد که همان خروجی ها را تولید می‌کند اما معادل های متناقض نمی‌باشند.به عنوان مثال تراکنش T5 را در نظر بگیرید که دلار را از حساب b به حساب a انتقال می‌دهد.زمانبندی ۸ را همانگونه که در شکل ۱۳-۱۴ تعریف شده است نگه دارید. ما استنباط می‌کنیم که زمانبندی ۸ معادل متضاد با زمانبندی سریالی<*T5* ,*T1*> نمی‌باشد زیرا در زمانبندی ۸ دستورالعمل نوشتن (b) از T5 با خواندن (b) در دستورالعمل T1 متناقض است. این یال *T5*→*T1* در گراف اولویت را ایجاد می‌کند. به همین نحو مشاهده می‌کنیم که دستورالعمل نوشتن (a) از T1 با دستورالعمل خواندن از T5 متناقض است و یال *T1*→*T5* را ایجاد می‌کند.این نشان می‌دهد که گراف اولویت یک حلقه دارد و زمانبندی ۸ قابلیت سریال پذیری ندارد. اما مقادیر نهایی حساب های A و B پس از اجرای هرزمانبندی ۸ یا زمانبندی سریالی <*T5* ,*T1*> به ترتیب همان ۹۶۰و۲۰۴۰ دلار می‌باشد.

از این مثال می‌توانیم مشاهده کنیم که تعاریفی با دقت کمتر از معادل زمانبندی نسبت به تعادل متناقض وجود دارد. برای این سیستم که تولید ۸ زمانبندی با همان خروجی ها را به عنوان زمانبندی سریالی<*T5* ,*T1*>تعیین می‌کند،باید به جای عملیات نوشتن و خواندن محاسبات انجام شده توسط T1و T5 را تجزیه و تحلیل کرد. در مجموع، انجام چنین تجزیه و تحلیلی دشوار و از نظر محاسباتی پرهزینه است. در مثال ما، نتیجه نهایی همانند زمانبندی سریالی است به دلیل این واقعیت ریاضی که جمع و تفریق جابجایی پذیر می باشند. در حالیکه مشاهده مثال ساده ما ممکن است آسان به نظر برسد اما مورد کلی خیلی ساده نیست زیرا یک تراکنش ممکن است به عنوان حالت SQL پیچیده ،یک برنامه جاوا با JDBC شرح داده شود.

اما تعاریف دیگری از معادل زمانبندی تنها بر مبنای عملیات نوشتن و خواندن موجود است. یکی از این تعاریف تعادل نظری است،تعریفی که به مفهوم سریال پذیری نظری منجر می‌شود. سریال پذیری نظری در اقداماتی به درجه بالایی از پیچیدگی محاسباتی منجر می‌شود استفاده نمی‌شود. بنابراین بحث سریال پذیری نظری را به فصل ۱۵ موکول می‌کنیم اما برای تکمیل بحث دقت داشته باشید که در این مبحث زمانبندی ۸ یک قابلیت سریال پذیری نظری نمی‌باشد.

۱۴-۷ ایزولاسیون تراکنش و اتمیک بودن

تا کنون برنامه های زمانی را بررسی کردیم که به طور ضمنی فرض می‌کردند که هیچ خطای تراکنشی وجود ندارد. اکنون تاثیر خطاهای تراکنش در حین اجرای همزمان را مشخص می‌کنیم.

T_1	T_5
read(A)	
A:=A−50	
write(A)	
	read(B)
	B:=B−10
	write(B)
read(B)	
B:=B+50	
write(B)	
	read(A)
	A:=A+10
	write(A)

شکل ۱۴-۱۴ زمانبندی ۹- یک زمانبندی غیرقابل بازیابی

چنانچه تراکنش Ti به هر دلیلی با خطا مواجه شود،ما باید اثر این تراکنش را خنثی کنیم تا ویژگی ظرفیت اتمی تراکنش را تضمین کنیم.در یک سیستم که اجرای همزمان مجاز است، ویژگی اتمیک بودن نیازمند هر اقدامی از تراکنش Tj مستقل از Ti است(Tjداده خواندن شده ای دارد که توسط Ti نوشتن شده است) و همچنین بی نتیجه مانده است. به منظور دستیابی به این موضوع باید محدودیت هایی از نوع زمانبندی که در سیستم مجاز است اعمال کنیم.

در دو بخش زیر،مسئله ای که چگونه برنامه های زمانی از نقطه نظر بازیابی از خطای تراکنش قابل دسترسی هستند را مشخص میکنیم. در فصل ۱۵ شرح می‌دهیم که چگونه تضمین کنیم که تنها چنین برنامه های زمانی قابل دسترسی تولید می‌شوند.

۱-۷-۱۴ برنامه های زمانی قابل بازیابی

برنامه جزئی ۹ در شکل ۱۴-۱۴ را در نظر بگیرید، که T7 تراکنشی است که تنها یک دستورالعمل را اجرا می‌کند:خواندن(A). این برنامه را جزئی می‌نامیم زیرا شامل یک عملیات بی نتیجه یا انجام برای T6 نمی‌شود. بنابراین T7 زمانی که T6هنوز در حال فعالیت است شروع بکار می‌کند.اکنون فرض کنید که T6پیش از اینکه شروع بکار کند با خطا مواجه می‌شود. T7 مقدار آیتم داده A نوشته شده توسط T6 را می‌خواند. بنابراین می‌گوییم که T7 وابسته به T6 است. به همین دلیل،T7 را بی نتیجه رها می‌کنیم تا ظرفیت اتمی را تضمین کنیم. اما T7 در حال حاضر شروع بکار کرده است و نمی‌تواند بی نتیجه رها شود. بنابراین موقعیتی داریم که غیرممکن است تا به درستی آن را از خطای T6 بازیابی کنیم.

برنامه۹ مثالی از یک زمانبندی غیرقابل بازیابی است. یک زمانبندی قابل بازیابی است که برای هر دو تراکنش های TiوTj است به طوریکه Tj یک داده را پیش از اینکه توسط Ti نوشته شود می‌خواند ،عملیات شروع بکار Ti پیش از عملیات شروع بکار Tj نمایان می‌شود. به عنوان مثال از فرانمای۹ می‌تواند قابل بازیابی می‌باشد،شروع بکار T7تا زمانی که T6 شروع بکار کند تاخیر دارد.

۲-۷-۱۴ زمان بندی بدون سقوط آبشاری

حتی چنانچه یک زمانبندی قابل بازیابی باشد برای پوشاندن صحیح از خطای تراکنش Ti باید به چندین تراکنش قبل بازگردیم. اگر تراکنش ها داده نوشتن شده Ti را خوانده باشند چنین موقعیت هایی اتفاق می‌افتد. زمانبندی جزئی شکل ۱۵-۱۴ را به عنوان یک مثال در نظر بگیرید.تراکنش T8 مقداری از A را نوشته که تراکنش T10آن را می‌خواند.در این نقطه فرض کنید که، T8 با خطا مواجه می‌شود. T8 باید به rollback شود. زیرا T9 وابسته به T8 است،T9 باید بازگردانده شود. به دلیل اینکه T10وابسته

به T9 است،T10 باید به rollback شود. این عارضه که یک تراکنش واحد با خطا مواجه می‌شود و باعث می‌شود مجموعه ای از تراکنش ها به عقب بازگردند عقبگرد آبشاری نامیده می‌شود.

عقبگرد آبشاری نامطلوب است زیرا باعث لغو میزان چشمگیری از فعالیت ها می‌شود اما محدود کردن فرانما به قسمت هایی که عقب گرد آبشاری نمی‌تواند انجام شود مطلوب است. چنین زمانبندی غیر عقبگرد آبشاری نامیده می‌شود. معمولا سبک زمانبندی برای دو تراکنش TIوTJ به طوریکه TJ یک داده را پیش از نوشتن توسط TI می‌خواند، عملیات شروع بکار TI پیش از عملیات خواندن TJ آغاز می شود. بررسی اینکه هر زمانبندی سقوط آبشاری قابل بازیابی باشد آسان است.

۸-۱۴ مراحل ایزولاسیون تراکنش

قابلیت سریال پذیری مفهوم مهمی است زیرا به برنامه نویس ها این امکان را می‌دهد تا مسائل مربوطه به همزمانی را زمانی که به تراکنش ها کد می دهد نادیده بگیرند. چنانچه هر تراکنش این ویژگی را داشته باشد که چنانچه به تنهایی اجرا شود ثبات پایگاه داده را حفظ می‌کند بنابراین قابلیت سریال پذیری تضمین می‌کند که اجراهای همزمان پایایی را حفظ می‌کنند اما پروتوکول ها به تضمین قابلیت سریال پذیری نیاز دارند تا زمان بسیار کمی برای نرم افزارهای مشخص ایجاد کنند. در این موارد مراحل ضعیف تر پیگیری استفاده می‌شود. استفاده از مراحل ضعیف تر پایایی مسئولیت بیشتری برای برنامه نویس ایجاد می‌کند تا صحت پایگاه داده را تضمین کند. استاندارد SQL همچنین باعث می‌شود تا یک تراکنش مشخص کند که آیا ممکن است به روشی اجرا شود که با تراکنش های دیگر غیر قابل سریال پذیر شود. به عنوان مثال،یک تراکنش ممکن است در مرحله ایزولاسیون فعالیت خواندن داده را آغاز کند و باعث شود تراکنش یک داده را حتی توسط تراکنش که شروع بکار نکرده است خواندن کند. SQL همچنین ویژگی هایی را به نفع تراکنش های طولانی ارائه می‌دهد که به صحت نتایج آن نیازی نیست. اگر این تراکنش ها به روشی قابل سریال پذیری اجرا شوند،می‌توانند با تراکنش‌های دیگر مداخله کنند و باعث شوند اجرای تراکنش های دیگر به تاخیر افتد.

مراحل ایزولاسیون که توسط استاندارد SQL مشخص می‌شوند به شرح زیر می باشد:

• قابلیت سریال پذیری: قابلیت سریال پذیری معمولا اجرای متوالی تراکنش ها را تضمین می‌کند. اما همانگونه که به طور متخصر شرح داده شد، بعضی سیستم های پایگاه داده این مرحله از ایزولاسیون را تنها به روشی انجام می‌دهند که ممکن است در شرایط خاص اجراهای قابل سریال پذیری را امکان پذیر کنند.

• خواندن قابل تکرار: خواندن قابل تکرار به داده ای که شروع بکار کرده است این امکان را می‌دهد تا خواندن شود و بیشتر مستلزم آن است که میان دو خواندن یک آیتم داده توسط یک تراکنش هیچ تراکنش دیگری ممکن نیست آن را بروزکند. اما تراکنش با توجه به دیگر تراکنش ها ممکن است سریال پذیر نباشد. به عنوان مثال زمانی که در حال جستجوی داده ای برای رضایت بخشی بعضی شرایط می باشد یک تراکنش ممکن است داده هایی را پیدا کند که توسط تراکنش شروع بکار کرده تعبیه شده اند اما داده دیگری پیدا نمی‌کند که با همان تراکنش تعبیه شده باشد.

• شروع بکار خواندن تنها باعث می‌شود داده شروع بکار کرده خوانده شود اما به خواندن های قابل تکرار نیازی ندارد. به عنوان مثال،میان خواندن دو داده با تراکنش،تراکنش دیگر ممکن است داده را بروزرسانی کرده و شروع بکار کند.

• خواندن شروع بکار نکرده باعث می‌شود داده شروع بکار نکرده خوانده نشود. پایین ترین مرحله ایزولاسیون را SQL انجام می‌دهد.

تمامی مراحل ایزولاسیون بالا علاوه براین نوشتن نامنظم را نمی‌پذیرند،عدم پذیرش نوشتن یک داده که تاکنون توسط تراکنش دیگر نوشته شده است،هنوز شروع بکار نکرده یا بی نتیجه مانده است.

بعضی سیستم های پایگاه داده به صورت پیش فرض در مرحله ایزولاسیون انجام خواندن اجرا میشود.در SQL ممکن است به صراحت مرحله ایزولاسیون را به جای اینکه تنظیمات پیش فرض را بپذیرد خودتان تنظیم کنید. به عنوان مثال"جمله قابلیت سریال پذیری مرحله ایزولاسیون تراکنش را تنظیم کنید" مرحله ایزولاسیون را برای سریال پذیری تنظیم می‌کند،در مقابل هرکدام از مراحل دیگر ایزولاسیون باید مشخص شوند و ترکیب بالا را به طور کلی پشتیبانی کنند،سرورSQL،SQL،POSTGRE،DB2 از این ترکیب استفاده می‌کنند تا مرحله ایزولاسیون را با اختصارهای خود برای مدل ایزولاسیون تغییر دهند.

تغییر مرحله ایزولاسیون باید به عنوان اولین حالت یک تراکنش انجام شود. علاوه براین شروع بکار خودکار حالت های فردی چنانچه تنظیمات پیش فرض سیستم نباشد باید خاموش شود. عملکردهای API مانند روش JDBC، ارتباطات تنظیم شروع بکار خودکار (غلط) که در بخش ۷-۱-۱-۵ مشاهده کردیم می توانند برای اجرا مورد استفاده قرار گیرند. علاوه براین در JDBC روش اتصالات،تنظیم ایزولاسیون تراکنش می‌تواند برای تنظیم مرحله ایزولاسیون استفاده شود،برای جزئیات دستورالعمل ها ی JDBC را مشاهده کنید.یک طراح نرم افزار ممکن است تصمیم بگیرد مرحله ایزولاسیون ضعیف تری را انتخاب کند تا عملکرد سیستم را ارتقاء دهد. همانگونه که در فصل ۹-۱۴ و۹-۱۵ مشاهده خواهیم کرد تضمین قابلیت سریال پذیری ممکن است باعث شود یک تراکنش منتظر تراکنش دیگر بماند یا در بعضی موارد به طور غیرعادی خاتمه می‌یابد زیرا تراکنش به عنوان بخشی از اجرای متوالی اجرا نمی‌شود. در حالیکه ممکن است کوته بینی به نظر برسد که پایگاه داده برای ثبات عملکرد ریسک کند این تجارت خاموش باعث می شود مطمئن شویم که عدم ثبات انجام شده به نرم افزار مرتبط نیست.

چندین وسیله برای اجرای مرحله ایزولاسیون وجود دارد. تا زمانی که انجام تراکنش سریال پذیری را تضمین می کند،طراح یک نرم افزار پایگاه داده یا کاربر یک نرم افزار نیازی به جزئیات چنین اجراهایی ندارد جز زمانی که با مسائل عملکردی مواجه شود. متاسفانه حتی اگر مرحله ایزولاسیون قابل سریال پذیری تنظیم شود،بعضی سیستم های پایگاه داده به طور حقیقی ایزولاسیون خفیف تری اجرا می کنند که از هر اجرای قابل سریال پذیری ممکن جلوگیری نمی کند. این مسئله را در بخش ۹-۱۴ دوباره مرور می کنیم. چنانچه مراحل ضعیف تر ایزولاسیون به طور آشکار یا ضمنی مورد استفاده قرار گیرند طراح نرم افزار بایداز جزئیات اجرا آگاه باشد تا احتمال بی ثباتی ناشی از فقدان قابلیت سریال پذیری را کاهش دهد یا جلوگیری کند.

قابلیت سریال پذیری در دنیای واقعی

زمانبندی سریال پذیر روش ایده آلی برای تضمین پایایی می باشد اما ما در زندگی روزمره چنین الزامات سختی را مطرح نمی-کنیم. یک وب سایت فروش کالا را در نظر بگیرید که اجناسی را برای فروش پیشنهاد می‌کند ممکن است یک آیتم رابه عنوان موجودی انبار لیست کند،همچنین با گذشت زمان یک کاربر آیتمی انتخاب می‌کند تا فرآیند بررسی پیش می‌رود که ممکن است یک آیتم دیگر در دسترس باشد. از نظر پایگاه داده این یک خواندن غیرقابل تکرار است.

به عنوان مثال دیگر،انتخاب صندلی را برای یک مسافرت هوایی در نظر بگیرید،فرض کنید که یک مسافر یک خط سیر رزرو کرده است و اکنون برای هر پرواز صندلی انتخاب می‌کند. بسیاری از وب سایت های هوایی به کاربران این امکان را می‌دهند از پروازهای مختلف یک صندلی انتخاب کند،پس از این از کاربر درخواست می‌شود که انتخاب خود را تأئید کند. در همین حین مسافران دیگر می‌توانند صندلی خود را انتخاب کنند یا انتخاب صندلی خود را برای همان گروه تغییر دهند. بنابراین قابلیت انتخاب صندلی که به مسافر نشان داده شده است به طور واقعی تغییر می‌کند اما زمانی که مسافر فرآیند انتخاب صندلی را آغاز می‌کند تصویری از صندلی در دسترس به مسافر نشان داده شده است . حتی اگر دو مسافر در همان زمان صندلی انتخاب کنند احتمالاآنها صندلی های متفاوتی انتخاب می‌کنند و اگر این چینین باشد هیچ رقابت واقعی وجود نخواهد داشت. اما تراکنش ها به وسیله مسافران دیگر،بروز رسانی میشوند و منجر به یک حلقه در گراف اولویت می شوند. اگر دو مسافر همزمان دو صندلی انتخاب کنند و به طور واقعی همان صندلی را انتخاب کنند یکی از

آنها نمی تواند در صندلی که انتخاب کرده است بنشیند،اما وضعیت به سادگی با درخواست از انتخاب مجدد مسافر با به روز رسانی اطلاعات صندلی های در دسترس حل می شود.

ممکن است قابلیت سریال پذیری با اجازه اینکه تنها یک مسافر انتخاب صندلی پرواز را در یک زمان انجام دهد بزور انجام شود. اما انجام این کار باعث بروز تاخیرهای چشمگیری برای مسافرانی می‌شودکه منتظر انتخاب صندلی پرواز هستند. به طور کلی یک مسافر که زمان زیادی را صرف انتخاب صندلی می کند باعث مشکلات جدی برای مسافران دیگر می‌شود. در مقابل هر تراکنشی معمولا به چندین بخش تقسیم می‌شود که نیازمند وساطت کاربر است و بخشی که تنها در پایگاه داده اجرا می‌شود. در مثال بالا تراکنش پایگاه داده بررسی می‌شود اگر صندلی های انتخاب شده توسط کاربر هنوز در دسترس باشند و همچنین انتخاب صندلی در پایگاه داده بروز رسانی شود. قابلیت سریال پذیری تنها برای تراکنش هایی که در پایگاه داده اجرا می شوند بدون دخالت پایگاه داده تضمین می شوند.

۹-۱۴ اجرای مرحله ایزولاسیون

تاکنون شاهد این بودیم که یک زمانبندی چه ویژگی هایی باید داشته باشد اگر پایگاه داده را در یک حالت نامتناقض قرار می‌دهد و باعث می شود تراکنش با خطا مواجه شود تا با روشی ایمن آن را کنترل کند.

چندین سیاست کنترل همزمانی وجود دارد که ما می توانیم برای تضمین از آنها استفاده کنیم حتی وقتی تراکنش های متعدد همزمان اجرا می شوند،صرف نظر از اینکه چطور سیستم عامل منابع زمانی را میان تراکنش های دیگر توزیع می کند یک زمانبندی قابل دستری تولید می‌شود.

این را به عنوان یک مثال کم مایه از روش کنترل همزمانی در نظر بگیرید:یک تراکنش پیش از اینکه آغاز شود در کل پایگاه داده به یک قفل نیاز دارد و پس از اینکه شروع بکار کرد قفل را باز می‌کند. در حالیکه یک تراکنش قفل می‌ماند تا هیچ تراکنش دیگری به قفل دسترسی پیدا نکند بنابراین همه تراکنش ها صبر می‌کنند تا قفل باز شود.به عنوان نتیجه ای از روش قفل کردن تنها یک تراکنش در یک زمان انجام می‌شود. بنابراین تنها زمانبندی سریالی تولید می‌شود. این ها قابلیت سریال پذیری کم مایه ای هستند و تأئید اینکه قابل بازیابی و عقبگرد آبشاری هستند ساده است. یک روش کنترل همزمانی مانند این به یک عملکرد ضعیف منجر می شود زیرا تراکنش را مجبور می‌کند تا برای تراکنش های جلوتر صبر کند تا آنها بتوانند تراکنش که شروع کرده بودند را تمام

کنند.به عبارت دیگر،خواندنه ضعیفی از همزمانی ارائه می‌شود(به جای اینکه هیچ همزمانی نشان ندهد). همانطور که در بخش ۵-۱۴ مشاهده کردیم اجرای همزمان مزایای عملکردی قابل توجهی دارد.

هدف سیاست های کنترل همزمان ارائه خواندنه بالایی از همزمانی است در حالیکه تضمین می کند تمام زمان بندی هایی که متناقض یا قابلیت سریال پذیری دیدگاهی دارند قابل بازیابی و بدون عقبگرد آبشاری هستند می توانند تولید شوند.

در اینجا چگونگی عملکرد مهمترین مکانیسم های همزمانی را بررسی می کنیم و جزئیات را به بخش ۱۵ موکول می کنیم.

۱-۹-۱۴ قفل گذاری

به جای قفل کردن کل پایگاه داده یک تراکنش می تواند تنها داده هایی که به آن دستیابی دارد را قفل کند. تحت چنین سیاستی تراکنش باید تا زمان طولانی قفل شود که قابلیت سریال پذیری را تضمین کند اما برای دوره ای کوتاه نباید بیش از اندازه به عملکرد آسیب بزند. اطلاعات مکمل حالت هایی از SQL هستند که در بخش ۱۰-۱۴ مشاهده کردیم جایی که دسترسی به آیتم های داده یک قضیه بستگی دارند.در فصل ۱۵ پروتوکل قفل دو مرحله ای ساده ای را ارائه دادیم که به طور وسیعی استفاده می-شود تا قابلیت سریال پذیری را تضمین کند. به سادگی بیان می‌شود که قفل دو مرحله ای نیازمند تراکنشی است که دو مرحله داشته باشد،مرحله ای که به قفل نیاز دارد اما هر قفلی را باز نمی‌کند و دومین مرحله که تراکنش قفل را باز می‌کند به هیچ چیز احتیاج ندارد.(معمولا قفل ها در عمل تنها زمانی باز می‌شوند که تراکنش کاملا اجرا شده است و یا شروع بکار نکرده است و یا بی خاتمه پایان یافته است.)

اگر دونوع قفل داشته باشیم برای نتیجه قفل کردن به توسعه بیشتری نیاز داریم. قفل های مشترک برای داده هایی استفاده می-شوند که خواندن تراکنش و قفل های آشکار برای نوشتن آنها استفاده می‌شوند. بعضی تراکنش ها می‌توانند قفل مشترک برهمان داده ها را در همان زمان داشته باشند اما یک تراکنش می تواند یک قفل آشکار بر داده داشته باشد تنها اگر هیچ تراکنش دیگری بر هیچ داده ای قفل نداشته باشد. استفاده از این دو حالت با دو مرحله قفل کردن خواندن همزمان داده را امکان پذیر می‌کند در حالیکه هنوز قابلیت سریال پذیری را تضمین کند.

۲-۹-۱۴ مهرهای زمانی

نوع دیگری از تکنیک ها برای اجرای وظایف ایزولاسیون هر تراکنش مهرهای زمانی است. برای هر داده سیستم دو مهر زمانی حفظ می‌شود. خواندن مهر زمانی آیتم داده بزرگترین مهر زمانی تراکنش هایی که آیتم داده را خواندن می کند اجرا می کند.

مهر زمانی نوشتن داده تراکنش ها را اجرا می‌کند،مقدار جریان داده را نوشتن می‌کند. مهرهای زمانی برای تضمین تراکنش هایی استفاده می‌شود که به هر آیتم داده دسترسی دارند. ترتیبی از مهرهای زمانی اگر دسترسی آنها متناقض شود . وقتی چنین چیزی ممکن است تراکنش های متخلفانه بی نتیجه می‌مانند و با یک مهر زمانی جدید دوباره اغاز می‌شود.

۳-۹-۱۴ ورژن های چندگانه و ایزولاسیون نمایش لحظه ای

حفظ بیش از یک نسخه از آیتم داده ممکن است باعث شود یک تراکنش نسخه قدیمی داده را به جای یک نسخه جدید نوشتن شده توسط یک تراکنش ناوابسطه یا توسط تراکنشی که پس از آن به ترتیب سریال پذیری اجرا می‌شود را بخواند.انواعی از تکنیک

های کنترل چندین نسخه ای وجود دارد. یک نوع ویژه آن ایزولاسیون نمایش لحظه ای نامیده می‌شود که به طور گسترده ای در عمل استفاده می‌شود.

در ایزولاسیون نمایش لحظه ای می‌توانیم تصور کنیم به هر تراکنش ورژن خود تراکنش یا نمایش لحظه ای از پایگاه داده زمانی که آغاز می‌شود داده شده است. داده از این ورژن شخصی خواندن می‌شود و بنابراین از بروزرسانی های انجام شده توسط تراکنش های دیگر جدا می‌شود. اگر تراکنش پایگاه داده را بروزرسانی کند که بروزرسانی تنها در ورژن خود نه در خود پایگاه داده واقعی نمایان شود. اطلاعات این بروز رسانی ها ذخیره می‌شود تا به روزرسانی بتواند برای پایگاه داده"واقعی" چنانچه تراکنش شروع بکار کند استفاده شود.

زمانی که تراکنش T تا حدودی وارد حالت شروع بکار می‌شود،اگر هیچ تراکنش همزمان دیگری داده ای که میل بروزرسانی دارد را تائید نکند سپس وارد حالت شروع بکار می‌شود. بعضی تراکنش ها نمی توانند بی نتیجه بمانند.

ایزولاسیون نمایش لحظه ای تضمین می‌کند که هرگز نباید منتظر خواندن داده ماند(برخلاف قفل کردن). تراکنش های فقط خواندنی نمی‌توانند بی نتیجه بمانند،تنها آنهایی که داده را تائید می‌کنند ریسک خفیفی از خاتمه غیرعادی را اجرا می‌کنند. زیرا هر تراکنش ورژن خود یا نمایش لحظه ای پایگاه داده را خواندن می‌کند،داده در حال خواندن باعث نمی‌شود بروزرسانی های متوالی توسط تراکنش های دیگر آزمایش شود.(برخلاف قفل کردن) زیرا بیشتر تراکنش ها فقط خواندنی هستند(و برخی دیگر بیشتر داده ها را زمانی که آنها را بروزرسانی می کند می خوانند) زمانی که این عمل با قفل کردن مقایسه می شود منبع عمده ارتقاء عملکرد است.

مشکل دیگر با ایزولاسیون نمایش لحظه ای این است که به طرز ابهام برانگیزی ایزولاسیون های بسیاری ارائه می‌دهد. دو تراکنش TوT را در نظر بگیرید. مشاهده می‌کنید که همه بروزرسانی ها توسط Tو T انجام شده است،زیرا یکی باید دیگری را به ترتیب سلسله وار دنبال کنند. در ایزولاسیون نمایش لحظه ای دیگر،مواردی وجود دارد که هیچ تراکنشی به روز رسانی تراکنش دیگر را مشاهده نمی‌کند.این شرایطی است که نمی‌تواند در یک اجرای قابل سریال پذیر انجام شود. در برخی موارد (به جای اکثر موارد)،دسترسی داده با دو تراکنش تناقض ندارد و هیچ مشکلی وجود ندارد. اما اگر T برخی داده هایی را خواندن کند که T بروز رسانی می‌کند و T بخواند بعضی داده هایی که T بروزرسانی می کند ممکن است هردو تراکنش نتوانند بروزرسانی انجام شده را توسط دیگری بخوانند. در نتیجه در فصل ۱۵ مشاهده خواهیم کرد که ممکن است یک پایگاه داده با وضعیت متناقض را نتوان از هر اجرای سریال پذیری به دست آورد.ORACLE،POSTGER SQL،و سرور SQL گزینه ای از ایزولاسیون نمایش لحظه ای ارائه می‌کنند و ORACLEو POSTGER SQL مرحله ایزولاسیون قابل سریال پذیری را با استفاده از ایزولاسیون نمایش لحظه ای اجرا می‌کنند. در نتیجه اجرای قابل سریال پذیر می‌تواند در شرایط استثنایی باعث اجرای غیرقابل سریال پذیر شود. سرور SQL در مقابل دربرگیرنده مرحله ایزولاسیون اضافی فراتر از نوع استاندارد آن است که نمایش لحظه ای نامیده می‌شود تا گزینه ای از ایزولاسیون نمایش لحظه ای را پیشنهاد کند.

۱۴,۱۰ تراکنش ها به بیانSQL

در بخش ۳-۴ برای شناسایی شروع و پایان تراکنش ها ترکیب SQL را ارائه کردیم. اکنون که بعضی مسائل در تضمین ویژگی های ACID برای تراکنش ها را مشاهده کردیم حاضریم تا بررسی کنیم چگونه آن ویژگی ها تضمین می‌کنند چه زمانی تراکنش ها به عنوان توالی از احکام SQL به جای مدل محدود شده خواندن و نوشتن ساده مشخص می‌شوند که ما تا این نقطه بررسی کردیم.

در مدل ساده فرض می‌کنیم مجموعه ای از آیتم های داده وجود دارد در حالیکه مدل ساده باعث می‌شود مقدار آیتم داده تغییر کند،باعث نمی‌شود آیتم داده حذف یا تولید شود. اما در SQL احکام های اضافه ای که داده جدیدی ایجاد می‌کند و احکام حذف داده را پاک می‌کند این دو احکام تحت تاثیر عملیات نوشتن هستند زیرا آنها پایگاه داده را تغییر می‌دهند، اما تعامل آنها با عملیات دیگر تراکنش ها از آنچه که در مدل ساده ما دیده شده است متفاوت است. به عنوان یک مثال،پرسش نامه SQL زیر در پایگاه داده دانشگاه ما را در نظر بگیرید که تمام اساتیدی که بیش از ۹۰۰۰۰ دلار درآمد دارند را پیدا کردیم.

```
select ID, name
from instructor
where salary > 90000;
```

با استفاده از رابطه ای ساده برآورد می‌کنیم که فقط انشتین و برندت شرایط را رضایت بخش می‌کنند. اکنون فرض کنید در همان بازه زمانی در حال اجرای پرسش نامه خود هستیم . کاربر دیگر معلم جدید به نام جیمز که حقوق او ۱۰۰۰۰۰ دلار است را اضافه می‌کند. نتیجه پرسش نامه ها بسته به آنچه قبلا اضافه شده است یا پس از آن پرسش نامه اجرا می‌کند متفاوت است. در اجرای همزمان این تراکنش ها بطور حسی واضح است که آنها متضاد هستند اما این تضادی نیست که از مدل ساده ما به دست آید.

این موقعیت به عنوان رویداد فانتوم اشاره می‌شود زیرا یک تناقض در داده فانتوم وجود دارد.

مدل ساده ما از تراکنش ها نیازمند این است که عملیات بر آیتم داده خاصی فعالیت کند که به عنوان استدلالی برای عملیات داده شده است. در مدل ساده ما می توانیم به این مراحل خواندن و نوشتن نگاه کنیم تا ببینیم چه آیتم های داده ای مرجع هستند. اما در یکی از احکام SQL آیتم داده خاص مرجع شده ممکن است توسط یک عبارت مکانی پیش بینی شود. بنابراین همان تراکنش اگر بیش از یک بار انجام شود ممکن است داده های مختلف دیگری را ارجاع دهد که هربار اگر مقادیر در پایگاه داده بین اجراها تغییر کند اجرا می‌شوند. یکی از راه های مقابله با مشکل بالا تشخیص این است که برای کنترل همزمانی بررسی چندین تراکنش کافی نیست. اطلاعاتی که برای کشف چندین متغیر استفاده می‌شود باید توسط تراکنش قابل دسترسی باشند و همچنین به منظور کنترل همزمانی بررسی شوند. اطلاعاتی که برای کشف چندین متغیر استفاده می‌شوند می‌تواند با خواندن یا حذف یا در مورد یک شاخص حتی به وسیله بروزرسانی با ویژگی عمال کلیدی بروزرسانی شود. به عنوان مثال اگر قفل کردن برای کنترل همزمانی استفاده شود،ساختار داده ای که در یک رابطه چندین متغیر را مانند ساختار شاخص پیگیری می کنندباید بطرز مناسب قفل شوند. اگر چنین قفل هایی بتواند باعث همزمانی ضعیف در برخی شرایط شود،پروتوکل های قفل شاخص همزمانی را افزایش می-دهند اما برخلاف خواندن و حذف و پیش بینی های پرسش نامه قابلیت سریال پذیری را افزایش می دهند،در بخش ۳-۸-۱۵ در این مورد بحث شده است.

دوباره به پرسش نامه نگاهی می کنیم:

```
select ID, name
from instructor
where salary > 90000;
```

```
update instructor
set salary=salary* 0.9
where name='Wu';
```

اکنون با موقعیت جالبی روبه رو می‌شویم که آیا پرسش نامه با حالت های بروزرسانی متناقض است. اگر پرسش نامه تمام روابط اساتید را بخواند سپس چندین متغیر را داده WU می‌خواندن و با بروزرسانی تناقض پیدا می‌کند. اما اگر شاخصی قابل دسترس بود باعث می‌شود پرسش نامه ما دسترسی مستقیم به چندین متغیر با حقوق بیش از ۹۰۰۰۰ داشته باشد،بنابراین پرسش نامه ما به کل داده WU دسترسی ندارد در مثال ما رابطه در حقوق اساتید WU از ابتدا ۹۰۰۰۰ بوده است و پس از بروزرسانی به ۸۱۰۰۰ دلار کاهش یافته است.

اما با استفاده از روش بالا تناقض هایی ایجاد می‌شود که به تصمیم پردازش پرسش نامه سطح پایین توسط سیستم بستگی دارد که با مرور سطح کاربری از معنای دو حالت SQL ارتباطی ندارد. اگر بتوان مجموعه ای از متغیرهای انتخاب شده را توسط پیش بینی تحت تاثیر قرار داد یک روش جایگزین برای کنترل همزمانی،خواندن،حذف یا بروزرسانی ها را به عنوان تناقض به عنوان تناقض با پیش بینی یک رابطه مورد عمل قرار می‌دهد. در مثال پرسش نامه بالا پیش بینی حقوق <۹۰۰۰۰ است و به روزرسانی حقوق WU از ۹۰۰۰۰به مقداری بیشتر از ۹۰۰۰۰ است و بروز رسانی حقوق انشتین از مقدار بیش از ۹۰۰۰۰به مبلغ کمتر از ۹۰۰۰۰ یا برابر آن است که با این پیش بینی متناقض است. قفل کردن برمبنای این نظریه قفل پیش بینی نامیده می‌شود اما قفل پیش بینی پرهزینه است و عملا از آن استفاده نمی‌شود.

خلاصه:

یک تراکنش واحدی از اجرای برنامه است که به آیتم های داده مختلف دسترسی پیدا می‌کند و احتمالا آنها را بروز رسانی می کند. درک مفهوم تراکنش برای فهم و اجرای بروزرسانی داده در پایگاه داده ای مهم است که با چنین روشی که اجراها و خطاها ی همزمان به شکل های مختلف وجود دارد باعث نامتناقض شدن پایگاه داده می شوند.

تراکنش ها به ویژگی ACID نیاز دارند،ظرفیت اتمی،ثبات،ایزولاسیون و پایایی

- **ظرفیت اتمی** تضمین می‌کند یا همه یا هیچ کدام از تاثیرات تراکنش در پایگاه داده بازتابی ندارد یک خطا نمی‌تواند پایگاه داده را در حالتی که تراکنش تا حدودی اجرا شده است رها کند.
- **ثبات** تضمین می‌کند که اگر پایگاه داده در ابتدا ثابت باشد اجرای تراکنش پایگاه داده را در حالت پایدار رها می‌کند.
- **ایزولاسیون** تضمین می‌کند که اجرای همزمان تراکنش ها از یکدیگر جدا می‌شود تا هرکدام بر تراکنش دیگری که به طور همزمان اجرا می شود اثری نداشته باشد.
- **پایایی** تضمین می‌کند که وقتی یک تراکنش انجامش شد بروز رسانی ها حتی اگر سیستم با خطا مواجه شود از دست نمی‌رود.

اجرای هم زمان تراکنش ها از طریق استفاده سیستم و تراکنش ها بهبود می‌یابد و همچنین زمان انتظار تراکنش ها را کاهش می-دهد.

انواع مختلفی از ذخیره سازی در کامپیوتر وجود دارد،ذخیره سازی فرار و غیر فرار. داده در ذخیره سازی فرار مانند رم زمانی که کامپیوتر خراب شود پاک می‌شود. داده در ذخیره سازی غیرفرار مانند دیسک وقتی کامپیوتر خراب شود پاک نمی‌شود اما ممکن است بعضی اوقات به دلیل خطاهایی مانند خرابی دیسک پاک شود. داده در ذخیره سازی پایدار هرگز پاک نمی‌شود.

ذخیره سازی پایدار باید قابلیت دسترسی آنلاین داشته باشد و تقریبا با دیسک های منعکس کننده یا شکل های دیگر RAID ذخیره سازی داده اضافی را فراهم کند. ذخیره سازی پایدار،آرشیوی یا آفلاین شامل چندین کپی نواری از داده ذخیره شده در مکان های امن از نظر فیزیکی است.

- زمانی که چندین تراکنش همزمان در پایگاه داده اجرا می‌شوند،ثبات داده ممکن است حفظ شود. بنابراین لازم است سیستم تعامل میان تراکنش های همزمان را کنترل کند.
- زیرا یک تراکنش واحدی است که ثبات را حفظ می‌کند،اجرای سریالی تراکنش ها حفاظت ثبات را ضمانت می‌کند.
- زمانبندی اقدامات اصلی تراکنش هایی هستند که بر اجرای همزمان تاثیر گذار است،مانند عملیات خواندن و نوشتن. در حالیکه جزئیات داخلی اجرای تراکنش را جذب می‌کنند.
- ما نیاز داریم که هر زمانبندی تولید شده توسط فرآیندهای همزمان گروهی از تراکنش ها اثر مشابهی بر یک زمانبندی تولیدشده داشته باشد زمانی که این تراکنش ها به طور سریالی اجرا می‌شوند.
- سیستمی که این ویژگی را ضمانت می کند قابل سریال پذیری نامیده میشود.
- چندین نظریه مختلف از هم ارزی وجود دارد که مفاهیم قابلیت سریال پذیری متضاد و قابلیت سریال پذیری نمایشی را ایجاد می‌کنند.
- قابلیت سریال پذیری زمانبندی با اجرای همزمان تراکنش هایی ایجاد می‌شود که می تواند از طریق یکی از انواع مکانیسم هایی که سیاست کنترل همزمانی نامیده می‌شود تضمین شود.
- ما می‌توانیم یکی از زمان بندی های ارائه شده برای قابلیت سریال پذیری متضاد را با ساخت گراف اولویت برای زمانبندی و با جستجوی عدم حضور حلقه ها در نمودار آزمایش کنیم. اما سیاست های کنترل همزمانی موثرتری برای تضمین قابلیت سریال پذیری وجود دارد.
- زمانبندی باید قابل بازیابی باشد تا اطمینان حاصل کند که اگر تراکنش a اثر تراکنش b را مشاهده می‌کند و سپس b بی خاتمه پایان می یابد بنابراین a هم بی خاتمه پایان یابد.
- زمان بندی باید ترجیحا قابل عقبگرد باشد تا پایان بی خاتمه یک تراکنش باعث پایان بی خاتمه تراکنش های دیگر نشود. نبود سقوط آبشاری با امکان اینکه تراکنش ها تنها داده خواندن شدنی می شوند تضمین می شود.
- عنصر مدیریت کنترل همزمانی پایگاه داده مسئول سیاست های کنترل همزمانی است. فصل ۱۵ سیاست های کنترل همزمانی را شرح می دهد.

مرور اصطلاحات

- تراکنش
- ویژگی های acid
- ظرفیت اتمی
- ثبات
- ایزولاسیون
- پایایی
- حالت نامتناقض
- انواع ذخیره سازی
- ذخیره سازی فرار
- ذخیره سازی غیر فرار
- سیستم کنترل همزمانی

سیستم قابل بازیابی

حالت تراکنش

فعال

تا حدودی انجام شده

خطا داده

بی نتیجه مانده

انجام شده

منقطع

تراکنش

راه اندازی دوباره-کشتن

ترتیب قابل سریال پذیری نوشتن های خارجی قابل مشاهده

زمانبندی قابل بازیابی اجراهای همزمان

عقبگرد آبشاری اجرای سریالی

زمان بندی های تنگی آبشار زمانبندی

تناقض عملکرد کنترل همزمان

قفل تعادل متناقض

ورژن های چندگانه قابلیت سریال پذیری متناقض

قابلیت سریال پذیری آزمایش ایزولاسیون نمایش لحظه ای

گراف اولویت

تمرین های عملی

فرض کنید که یک سیستم پایگاه داده وجود دارد که هرگز خطا نمی‌کند،آیا یک مدیر بازیابی برای این سیستم مورد نیاز است؟

سیستم یک فایل مانند سیستم عامل مورد علاقه خودتان را بررسی کنید.

a. مراحلی که در حذف و ایجاد فایل مشارکت می کنند چه مراحلی هستند و مراحل خواندن داده در یک فایل چه مراحلی هستند؟
b. چگونگی ارتباط ظرفیت اتمی و پایایی در حذف و تولید فایل ها و خواندن داده روی فایل شرح دهید.

عملکردهای سیستم پایگاه داده توجه بیشتری به ویژگی های ACID نسبت به عملکردهای سیستم فایل دارد. علت این موضوع چیست؟

جملات زیرا را تائید کنید: اجرای همزمان تراکنش وقتی داده روی دیسک با سرعت کم اجرا میشود یا وقتی تراکنش ها طولانی هستند مهمتر است و وقتی داده در حافظه است و تراکنش ها کوتاه تر هستند کم اهمیت تر است.

به دلیل اینکه هرزمانبندی سریال پذیری پذیر متناقض قابل سریال پذیری دیده می شود، چرا ما بر قابلیت سریال پذیری به جای دید سریال پذیری پذیر تاکید می کنیم؟

گراف اولویت شکل ۱۶-۱۴ را در نظر بگیرید.آیا زمانبندی همراه قابلیت سریال پذیری متناقض است؟پاسخ خود را شرح دهید.

یک زمانبندی بدون سقوط آبشاری چیست؟ چرا یک زمانبندی بدون سقوط آبشاری زمانبندی معقولی است؟ آیا شرایطی وجود دارد که زمانبندی بدون سقوط سقوط ابشاری امکان پذیر شود،پاسخ خود را شرح دهید؟

گفته می شود که به روزرسانی از دست رفته به طرز غیرعادی انجام میشود اگر تراکنش TJ یک داده را نوشتن کند،سپس تراکنش TK داده را نوشتن میکند(احتمالا بر مبنای خواندن قبلی) پس از اینکه TJ داده را نوشتن میکند،به روز رسانی انجام شده توسط TK از دست رفته است،زیرا به روز رسانی انجام شده توسط TJ مقدار نوشتن شده توسط TK را نادیده گرفته است.

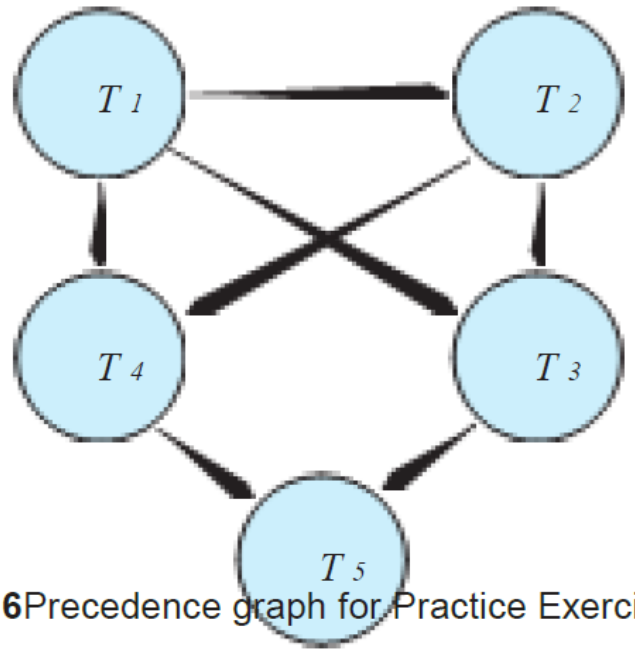


Figure 14.16 Precedence graph for Practice Exercise 14.6.

شکل ۱۶-۱۴ گراف اولویت برای تمرین ۶-۱۴

a- یک مثال برای زمانبندی از دست رفتن غیرطبیعی به روزرسانی ارائه دهید.
b- مثالی از زمانبندی ارائه دهید که نشان میدهد به روز رسانی از دست رفته به طرز غیرطبیعی با سطح ایزولاسیون خواندن انجام شده امکان پذیر است.
c- توضیح دهید که چرا به روزرسانی از دست رفته ی به طور غیرطبیعی مرحله ایزولاسیون با خواندن قابل تکرار امکان پذیر نیست.

پایگاه داده ای برای یک بانک را در نظر بگیرید که سیستم پایگاه داده از ایزولاسیون نمایش لحظه ای استفاده میکند. سناریو خاصی را شرح دهید که اجراهای غیرقابل سریال پذیری انجام میشوند که برای بانک مشکلاتی ایجاد می کنند.

پایگاه داده ای برای خط هوایی در نظر بگیرید که سیستم پایگاه داده از ایزولاسیون نمایش لحظه ای استفاده می کند،سناریوی خاصی را شرح دهید که اجراهای غیر قابل سریال پذیری اجرا میشوند اما خط هوایی ممکن است تمایل به پذیرش آن داشته باشد تا عملکرد بهتری به دست آورد.

تعریفی از زمانبندی بیانگر این است که عملیات ها می توانند به طور کلی بازمان مرتب شوند. سیستم پایگاه داده ای را در نظر بگیرید که یک سیستم را با فرآیندهای چندگانه اجرا میکند،که همیشه ممکن نیست ترتیب دقیقی میان عملیات هایی که در فرآیندهای مختلف اجرا شده است ایجاد کند.

اما عملیات به طور کلی می تواند بر آیتم داده مرتب شود.

آیا موقعیت بالا مشکلی برای تعریف قابلیت سریال پذیری متناقض ایجاد میکند؟ پاسخ خود را شرح دهید.

تمرین ها:

ویژگی های acid را شرح دهید. کاربرد هرکدام را توضیح دهید.

در طول اجرای آن یک تراکنش از چندین حالت عبور می کند تا در نهایت آن را انجام دهد یا تراکنش ها بی خاتمه می ماند. تمام ترتیب های ممکن حالت هایی که از تراکنش ممکن است عبورکند را شرح دهید. توضیح دهید که چرا انتقال هر حالت ممکن است انجام شود.

تفاوت میان زمانبندی سریالی و زمانبندی قابل سریال پذیری را شرح دهید.

دو تراکنش زیر را در نظر بگیرید

T_{13} :read(A);
read(B);
if$A=0$then$B:=B+1$;
write(B).

T_{14} :read(B);
read(A);
if$B=0$then$A:=A+1$;
write(A).

سازگاری مورد نیاز $A=0 \vee B=0$، با $A=B=0$ مقادیر اولیه در نظر بگیرید:

a-نشان دهید که هر اجرای سریالی شامل این دو تراکنش ها است که از ثبات پایگاه داده حفاظت میکند.

b-نشان دهید اجرای همزمان t13وt14 یک زمانبندی غیرقابل سریال پذیری ایجاد میکند.

c-آیا یک تراکنش همزمان از t13وt14 وجود دارد که زمانبندی قابل سریال پذیری تولید کند؟

مثالی از زمانبندی قابل سریال پذیری با دو تراکنش ارائه دهید که ترتیب تراکنش ها از نظر ترتیب متوالی متفاوت است.

یک زمانبندی قابل بازیابی چیست؟ چرا قابلیت بازیابی زمانبندی مطلوب است؟ آیا شرایطی وجود دارد که تحت آن امکان زمانبندی غیرقابل بازیابی مطلوب باشد. پاسخ خود را شرح دهید.

چرا سیستم پایگاه داده از اجرای همزمان تراکنش ها پشتیبانی می کند،بر خلاف زمانبندی اضافی که سعی میکند تضمین کند که اجرای همزمان هیچ مشکلی ایجاد نمیکند؟

شرح دهید که چرا سطح ایزولاسیون خواندن شده زمانبندی بدون سقوط آبشاری را تضمین میکند؟

برای هریک از مراحل ایزولاسیون زیر مثالی از زمانبندی با توجه به سطح مشخصی از ایزولاسیون ارائه دهید. اما قابل سریال پذیری نباشد:

a- خواندن انجام نشده
b- خواندن انجام شده
c- خواندن قابل تکرار

فرض کنید که علاوه بر عملیات نوشتن و خواندن یک عملیات پیش فرض را فعال می کنیم که همه متغیرها در ارتباط با r که p را پیش بینی میکنند خواندن می شوند.

a- مثالی از زمانبندی با استفاده از عملیات پیش خواندن ارائه دهید که رویداد فانتوم را نشان میدهد و این رویداد نتیجه یک تراکنش غیرقابل سریال پذیری نیست.
b- مثالی از زمانبندی ارائه دهید که یک تراکنش از عملیات پیش خواندن در رابطه با تراکنش های همزمان دیگر استفاده میکند تا یک متغیر را از r حذف کند اما زمانبندی یک تناقض فانتومی را نشان نمیدهد.(برای انجام این کار شما باید طرحی از رابطه r ارائه دهید و مقادیر نسبی متغیرهای حذف شده را نشان دهید.)

نکات کتاب شناسی

گری و ریوتر(۱۹۹۳) کتاب های مفصلی ارائه دادند که شامل مفاهیم فرآیندهای تراکنش،جزئیات اجراها و تکنیک ها از جمله کنترل مسائل همزمانی و بازیابی می شد. برنشتین و نیوکامر(۱۹۹۷) کتاب هایی ارائه دادند که دربرگیرنده ابعاد مختلفی از فرآیندهای تراکنش می شود.

اسواران و همکاران(۱۹۷۶) مفهوم قابلیت سریال پذیری را در رابطه با کار بر کنترل همزمانی بر سیستمR رسمی کردند.

منابع ابعاد خاصی از فرآیند تراکنش از جمله کنترل همزمانی و بازیابی را شامل میشود که در فصل ۱۵-۱۶ و۲۶ قرار دارند.