ADVANCED

DATA BASE

استاد : سرکارخانم دکترسلطانی

کاری از : مریم حمیدی نسب

فصل 1

“ Security”

(امنیت)

امنیت :

با وجود اینکه معمولا موضوعات امنیت داده ها اغلب با موضوعات جامعیت داده ها همراه است (حداقل در زمینه های غیر رسمی) اما این دومفهوم کاملا با هم متفاوت اند . امنیت به حفاظت پایگاه داده ها در مقابل انتشار ،تغییر یا خرابی غیر مجاز ،اطلاق می گردد اما جامعیت به دقت یا اعتبار داده ها اشاره دارد .

به طور ساده تر این که :

- امنیت یعنی حفاظت از داده ها در مقابل کاربردهای غیر مجاز .
- جامعیت یعنی حصول اطمینان از این مسئله که آیا کاری که کاربران سعی در انجام آن دارند ،درست می باشد یا خیر.

به عبارت ساده تر این که :امنیت یعنی اطمینان از اینکه کاربران کارهایی را انجام می دهند که برای انجام آنها مجوز دارند و جامعیت به معنی حصول اطمنیان از این نکته است که آنچه کاربران انجام میدهند ،درست است .

شباهت هایی هم بین این دو مفهوم وجود دارد :در هردو مورد ،سیستم باید از محدودیت هایی که کاربران نباید آنها را نقص کنند ،آگاه باشند . در هردو مورد ،این محدودیتها باید با یک زبان مناسب مشخص شوند و باید در کاتالوگ سیستم ذخیره شوند و در هر دو مورد ،DBMS باید بر اعمال کاربران نظارت داشته باشد تا مطمئن شود که محدودیت ها اعمال می شوند . در این فصل در مورد امنیت بحث خواهد شد.

ملاحظات کلی :

مساله امنیت ،جنبه های متعددی دارد که بعضی از آنها عبارتند از :

-جنبه های قانونی ،اجتماعی و اخلاقی (برای مثال آیا شخصی که تقاضایی دارد ،مانند مطلع شدن از سقف اعتباری یک مشتری ،دارای حق قانونی برای دریافت این اطلاعات هست ؟)

-کنترلهای فیزیکی :برای مثال آیا اتاق کامپیوتر یا سایت کامپیوتری قفل دارد یا به شکل دیگر ی محافظت می شود ؟)

- پرسشهای سیاست گذاری (برای مثال موسسه صاحب سیستم چگونه تصمیم می گیرد که چه کسانی به چه اطلاعاتی دسترسی داشته باشند .؟)
- مشکلات عملیاتی \(برای مثال اگر ازیک روش کلمه رمز استفاده می شود خود کلامات رمز چگونه سری باقی میمانند ؟و این کلمات رمز هر چند وقت یک بار عوض می شوند ؟)

- کنترلهای سخت افزاری (برای مثال آیا واحد پردازش ،ویژگی های امنیتی خاصی را مانند کلیدهای محافظ حافظه یا یک وضعیت عملیاتی محافظت شده را تدارک میبیند ؟)
- پشتیبانی سیستم عامل (برای مثال آیا سیستم عامل,محتویات حافظه اصلی و فایلهای دیسکی را پس از اینکه کار تمام شد از بین می برد ؟)
- موضوعاتی که مختص خود سیستم بانک اطلاعاتی هستند (برای مثال آیا سیستم بانک اطلاعاتی دارای مفهوم مالکیت داده ها است ؟)

امروزه DBMS جدید از یکی از دو روش کلی امنیت داده ها و یا هردوی آنها پشتیبانی می کنند . این دو روش عبارتند از :

کنترل اختیاری و کنترل اجباری .

. د رهر دو مورد واحد داده یا شی داده که باید از آن محافظت شود ممکن است کل یک بانک اطلاعاتی و یا یک جز مشخص از یک چند گانه باشد . تفاوت این دو روش به صورت زیر خلاصه شده است .

- **کنترل اختیاری** ،یک کاربر دارای حقوق دسترسی گوناگونی به اشیا مختلف است (که به آن حق انحصاری یا امتیاز می گویند) علاوه براین محدودیت هایی وجود دارند که مشخص میکنند چه کاربرانی چه حقوق هایی بر روی کدام اشیا دارند . بنابر این روش های اختیاری بسیار انعطاف پذیر هستند .

- در عوض در کنترل اجباری ،هرشی داده توسط یک سطح طبقه بندی خاص برچسب گذاری می شود و به هر کاربر ،یک سطح مجوز مشخص داده می شود. لذا یک شی داده ها قط در اختیار کاربرانی قرار می گیرد که برای دسترسی به آن شی دارای مجوز مناسب هستند . بنابر این ،روشهای اجباری دارای ماهیت سلسله مراتبی هستند و در نتیجه نسبتا از انعطاف پذیری کمتری برخوردار می باشند .

❖ نکته :در کنترل اختیاری برای کاربران سطوح تعیین میشود . و در کنترل اجباری برای داده ها سطح تعیین میگردد.

کنترل دسترسیابی اختیاری :برای تعریف محدودیتهای امنیتی (اختیاری)باید زبانی وجود داشته باشد. بنا به دلایل بدیهی بهتر است بگوییم چه چیزهایی مجاز هستند و چه چیزهایی مجاز نیستند . بنابراین زبانها نوعا از چنین تعریفی پشتیبانی می کنند ،یعنی زبان ها از مجوزها پشتیبانی میکنند . بنابراین ابتدا زبانی را برای تعریف مجوزها ارائه می دهیم .

مثال :

Aythority sa 3

Grant retrieve (s#'sname'city)'delete

ON s

TO JIM'MARY

که به طو رکلی نشان میدهد که مجوزها دارای چهار جزهستند که عبارتند از :

نام - SA3

یک یا چند امتیاز که به وسیله عبارت GRANT

مشخص میشوند از قبیل DELETE;RETRIEVE

متغیر رابطه ای که مجوز برای ان در نظر گرفته می شود .که به وسیله عبارت ON مشخص می شود .-s

یک یا چند کاربر که با عبارت TOمشخص می گردد. – jim’mary

- لیست اختیارات داده شده به کاربران می تواند RETRIEVE;DELETE; INSERT ; UPDATE باشد .

اگر کاربری بخواهد عملی را برروی شی انجام دهد که فاقد مجوز آن است چه اتفاقی می افتد؟

- MESSAGE
- Terminate
- USER Lock
- ایجاد یک log

توضیح:ساده ترین راه این است که درخواست وی رد شود. اغلب چنین پاسخی در مل موزد نیاز است لذا آن را به عنوان حالت پیش فرض در نظر می گیرم . اما در وضعیت های حساستر ،شاید اعمال دیگری مناسب باشند. برای مثال شاید لازم باشد برنامه خاتمه یابد و یا صفحه کلید قفل شود. همچنین شاید ثبت این تلاشها در یک فایل سابقه نیز مطلوب باشد تا بعدا تلاش های مربوط به نقض کردن امنیت تحلیل شوند و همچنین به عنوان مثال عامل بازدارنده در مقابل نفوذ غیر مجاز محسوب شوند .

برای حذف مجوزها از دستور :

Drop authority{authority name};

drop authority sa3: مثال

بازرسی حسابها –حسابرسی :(Audit trails)

توجه به این نکته مهم است که سیستم امنیتی بی عیب نیست . افراد نفوذی می توانند از سد کنترلهای منیتی عبور کنند مخصوصا اگر مبلغی که برای اینکار به آنها پرداخت می شود ،زیاد باشد . در موردی که داده ها حساس هستند بازرسی حسابها میتواند آنچه راکه اتفاق افتاده است را مشخص کند و موضوع را بررسی نماید که تحت کنترل است یا خیر .

در حقیقت حسابرسی یک فایل یا بانک اطلاعاتی خاص است که سیستم در آن اطلاعات را و تمام اعمال انجام شده توسط کاربران برروی داده های مجاز را نگهداری می کند.

یک درایه حسابرسی نمونه ممکن است حاوی اطلاعات زیر باشد :

- تقاضا (متن اصلی)
- پایانه ای که عملکرد موردنظر از طریق ان فراخوانی شده است
- کاربری که ملکرد را فراخوانی کرده است
- زمان و تاریخ عملکرد
- متغیر رابطه ای ،چندگانه ها ،و صفات تحت تاثیر
- مقادیر قبلی
- مقادیر جدید

کنترل دستیابی اجباری –MC

فرمت تعریف شدن mc:

دیتا آبجکت ها ممکن است یکی از سطوح دستیابی زیر را تعریف کنند :

- فوق سری –top secret
- سری – secret
- محرمانه – confidential
- عادی – unclassified

کاربران درصورتی میتوانند اطلاعات را ببینند که سطح دسترسی آنها بالاتر یا مساوی این طبقه بندی باشد .

- نکته :کاربر اطلاعاتی میتواند insert کند که سطح دسترسی به آن کوچکتر یا مساوی آن باشد .
- Palyinstantiation: به این پدیده که داده های یکسان برای کاربران مختلف متفاوت به نظر میرسند گفته میشود .- Pi

- Inference: استنتاج کردن داده هایی از داده های دیگر .

برای از بین بردن inferenceدو را وجود دارد :

- از Pi استفاده کنیم .
- جدول را به همان حالت رها کرده و به این امید باشیم که کاربران کنکاش نمی کنند . (به کاربر اجازه دیدن فیلدی را ندهیم)
- باید تعاملی بین security و practicality ایجاد شود که با آن trade off گفته میشود.

- **رمزگذاری داده ها (data incryption):**

حالتی را بررسی می کنیم که کاربر میخواهد سیستم را دور بزند (مثلا با حذف فیزیکی قسمتی از بانک اطلاعاتی وارد یک خط ارتباطی شود). یک روش برای جلوگیری از این کار رمزگذاری داده ها است . یعنی داده های حساس ،به صورت رمز شده ذخیره و منتقل شوند .

- چند اصطلاح رمزگذاری :

داده های رمز نشده را متن ساده می گویند .با قراردادن متن ساده در یک الگوریتم رمزگذاری این متن رمزگذاری میشود. ورودی این الگوریتم عبارتند از متن ساده و یک کلید رمزگذاری و خروجی آن متن رمزی میباشد . جزئیات الگوریتم رمزگذاری مخفی نیست ولی کلید رمزگذاری باید سری باشد.

GRANT: در SQL برای تعریف اختیارات از این دستور استفاده می کنیم .

REVOTE: برای صلب اختیارات از یک کاربر از این دستور استفاده میکنیم.

فصل 2

Recovery

:Recovery

دریک سیستم بانک اطلاعاتی اصولا ترمیم به معنای تصحیح خطا توسط خود بانک اطلاعاتی است یعنی در مواقعی که خطایی رخ میدهد و بانک اطلاعاتی در وضعیت ناپایداری قرار میگیرد ویژگی ترمیم باعث میشود تا بانک اطلاعاتی در وضعیت درستی قرار گیرد .

تراکنش (TRANSACTION):

تراکنش یک واحد منطقی از کار است .

```
BEGIN TRANSACTION;
UPDATE ACC 123{BALL=BALL-100}
 IF ERROR ACCURS  GO TO UNDO; END IF
UPDATE ACC 456{BALL=BALL=100}
IF ERROR ACCURS GO TO UNDO ;END IF;
COMMITE
GO TO FINISH;
UNDO
ROLLBACK
FINISH
```

کار این تراکنش انتقال پول از یک حساب به حساب دیگر است . کاری که باید صورت بگیرد شامل رشته ای از دو UPDATE است .

بعد از UPDATE اول دیتا بیس در یک وضعیت ناصحیح قرار میگیرد.

بعد از انجام UPDATE دوم اگر باز خطایی پیش بیاید باید بلافاصله تمام تغییرات UNDOشود .

تراکنش ممکن است شامل رشته ای از عملگرها یا اپراتورها باشد که از دید کاربر خارجی اتمیک به نظر می آید .

برای اینکه گارانتی شود که وقتی یک عملگر تراکنش را انجام داد بقیه عملگرها هم حتما انجام میشوند از

TRANSACATION یا TRASACATION MANAGER یا TP PROCESSING(که به آن ناظر پردازش تراکنش یا ناظر نیز میگویند) استفاده میشود.

عملگرهای COMMITEو ROLLBACK را حل کلیدهای حل این مساله محسوب میشوند:

- عملگر COMMITE انتهای موفقیت آمیز یک تراکنش را معین میکند.
- عملگر ROLLBACK انتهای ناموفق یک تراکنش را اعلام میکند. این عمل به مدیر تراکنش اعلام میکند که در حین این عمل مشکلی به وجود آمده و بانک اطلاعاتی احتمالا در حالتی نا پایدار است . و تمام به هنگام سازی ها باید رد یا لغو شوند .

:MESSAGE HANDLING

در بحث تراکنشها مدیریت پیغامها هر اتفاقی را که در سیستم می افتد را حتما باید به کاربر اعلام کند . اگر موفقیت آمیزبوده اعلام و اگر هم ناموفق بوده پیغام دهد .

نکته : چگونه میتوان یک به هنگام سازی ر ا لغو کرد ؟

پاسخ این است که سیستم یک ثبت وقایع (JOURNAL –LOG) را روی نوار یا معمولا دیسک نگهداری میکند که در آن جزئیات مربوط به عمل به هنگامرسانی (به ویژه مقادیر قبل و بعد شی به هنگام شده) ضبط می شود. بنابراین اگر لازم باشد که یک عمل به هنگام رسانی خاص لغو شدو سیتم می تواند با استفاده از درایه مربوطه به هنگام رسانی در ثبت وقایع شی به هنگام شده را به وضعیت قبلی خود برگرداند..

در عمل ثبت وقایع از دو بخش تشکیل شده است :

بخش فعال یا مستقیم –ACTIVE OR ONLINE

بخش آرشیو یا غیر مستقیم –ARCHIVE OR OFFLINE

طریقه ترمیم تراکنش:

COMMIT POINT:

دستور COMMITE در کنار موارد دیگر چیزی را به نام نقطه قبول (COMMITE POINT) ایجاد میکند. بنابراین نقطه قبول متناظر با انتهای یک واحد منطقی از کار است و نقطه ای است که در آن بانک اطلاعاتی در حالت پایدار قرار میگیرد یا باید قرار بگیرد .

- در COMMIT POINT دو کا رانجام میشود : .

- ✓ کلیه اعمال به هنگام رسانی که توسط برنامه در حال اجرا انجام میشوند از نقطه قبول قبلی پذیرفته پذیرفته می شوند یعنی به حالت پایدار درمیآیند .
- ✓ کلیه اعمال آدرس دهی بانک اطلاعاتی از دست می رود و کلیه قفلها ی چندگانه باز میشوند .

نکته :دو تراکنش در عین حال نمی توانند روی یک آبجکت اعمال شوند .

- این امکان وجود دارد که بعد ازاجرای فرمان COMMIT و قبل از نوشته شدن داده های به هنگام شده به صورت فیزیکی برروی دیسک ،سیستم دچار مشکل شود . یعنی این احتمال وجود دارد که حین خرابی سیستم داده هایی که در بافر حافظه قرار دارند از دست بروند . حتی در صورت بروز چنین اتفاقی روال شروع مجدد سیستم بازهم اعمال به هنگام رسانی لازم را روی بانک اطلاعاتی انجام می دهد. در اینجا سیستم می تواند بار بررسی مقادیر مربوطه در ثبت وقایع ،مقادیری که باید در بانک اطلاعاتی نوشته شوند را مشخص کند . (نتیجه می شود که قبل از اینکه پردازش با دستورCOMMIT کامل شود اطلاعات ثبت وقایع باید به صورت فیزیکی نوشته شوند . این قانون مهم را قانون پیش ثبت وقایع (**WRITE AHEAD LOG RULE**) می گویند.

❖ **تسهیلاتی که به ما در پیاده سازی کمک می کند:**

- UPDATEهای پایگاه داده در بافر نگهداری میشوند و اگر تراکنش خراب شود نیازی نیست در سطح فیزیکی UNDO کنیم .
- UPDATE های پایگاه داده به طور فیزیکی روی دیسک نوشته می شوند .تمام تغییرات همزمان در LOG به صورت فیزیکی درج می شوند در این حالت نیازی نیست REDOاز فضای فیزیکی انجام شود.

خواص ACID :

- می توان گفت تراکنشها دارای 4 ویژگی هستند :
- غیر قابل تفکیک بودن (ATOMICITY) :تراکنشها اتمیک هستند (همه یا هیچکدام)
- پایداری (CONSISTENCY):یعنی تراکنش بانک اطلاعاتی را از یک حالت پایدار به حالت پایدار دیگری می برد،بدون این که نیازی باشد در نقاط میانی پایداری بانک اطلاعاتی حفظ شود .
- جداسازی (ISOLATION) : تراکنشها از هم جدا هستند یعنی هرچند به طور کلی ممکن است تراکنشها به صورت همزمان اجرا شوند ولی تازمانی که آن تراکنش قبول میشود به هنگام رسانی هر تراکنش از سایر به هنگام رسانی ها پنهان است .
- تداوم (DURABILITY): پس از اینکه یک تراکنش قبول شد،به هنگام رسانی ها ی آن در بانک اطلاعاتی پایدار خواهند بود حتی اگر سیستم در ادامه کار دچار اختلال شود .

انواع خرابی ها :

LOCAL FAILURES: خرابی محلی تنها بر روی تراکنشی که در آن روی داده است تاثیر می گذارد .

Global FAILURES: خرای سراسری روی تمام تراکنشهایی که از زمان بروز خرابی مورد پردازش قرار گرفته اند تاثیر گسترده نامطلوبی روی کل سیستم می گذارد .

چنین خرابی هایی را می توان به دو دسته کلی تقسیم کرد:

خرابی سیستمی (مانند خرابی منبع تغذیه): این خرابی ها روی کلیه تراکنشهایی که در حال حاضر پردازش می وشوند تاثیر میگذارند اما بانک اطلاعاتی را از لحاظ فیزیکی خراب نمی کنند. این خرابی را گاهی اوقات خرابی نرم (soft crash) نیز میگویند .

خرابی های رسانه ای (مانند خرابی هد دیسک): باعث تخریب بانک اطلاعاتی یا بخشی از آن می شوند و حداقل روی تراکنشهایی که در حال حاضر از آن بخش استفاده می کنند تاثیر می گذارند . این خرابی را گاهی اوقات خرابی سخت(hard crash) نیز می نامند .

- **ترمیم رسانه :**

گفتیم خرابی رسانه ای خرابی است مانند خرابی هد دیسک که اصولا ترمیم این خرابی شامل بارگیری مجدد بانک اطلاعاتی از یک نسخه پشتیبان و سپس استفاده از ثبت وقایع برای تکرار کلیه اعمال انجام شده از زمانی که نسخه کپی گرفته شده است لازم نیست که تراکنشهایی که حین بروز خرابی در حال انجام بوده اند لغو شوند ،زیرا که به هنگام رسانی ها ی این تراکنش ها به هر حال نادیده گرفته می شوند . لزوم اجرای یک ترمیم رسانه ای نیاز به داشتن یک نرم افزار سودمند روگرفت /بازیابی restore/dump را نشان می دهد . بخش dump برای تهیه نسخ پشتیبان از بانک اطلاعاتی و بخش بازیابی بانک اطلاعاتی را از یک نسخه پشتیبان مجد ایجاد میکند.

قبول دو مرحله ای :

زمانی قبول دو مرحله ای اهمیت دارد که یک تراکنش بتواند با چندین "مدیر منبع" مستقل که هر کدام مجموعه منابع ترمیم پذیر خود را اداره و ثبت وقایع مخصوص به خود را نگهداری میکنند در تعامل باشد .

اگر commit دریافت شد دو مرحله هماهنگ کننده انجام میدهد :

- کلیه مدیران را برای انجام تراکنش آماده میکند .و سپس آنها پیغام" ok" یا "notok" را به هماهنگ کننده میدهند.
- هماهنگ کننده با توجه به پیغام های مدیر منبع اگر همه ok بودند تصمیم مورد "قبول" وگرنه "لغو" می شود.

فصل 3

همزمانی

دراین فصل به مدیریت کنترل همزمانی میپردازیم .

همزمانی :concurrency

سیستم های مدیریت بانک اطلاعاتی (dbms) اجازه میدهند چندین تراکنش در آن واحد به داده های مشابهی دسترسی داشته باشند و مسلما در چنین سیستمی نیاز به مکانیزم کنترل همزمانی می باشد تا از عدم تراکنشها ی همزمانی بایکدیگر اطمینان حاصل شود .

سه مشکل همزمانی عبارتند از :

- مشکل به هنگام رسانی مفقود شده (lup)
- مشکل وابستگی پذیرفته نشده (uDP)
- مشکل تحلیل ناسازگار (IAP)

TRANX B	TIME	TRANX A
	T1	RETRIVE t
RETRIVE t	T2	
	T3	Update t
Update t	T4	

LUP:

B مفقود می شود زیرا تراکنش T4به هنگام رسانی انجام شده در زمان حتی بدون نگاه کردن به این به هنگام رسانی ها تغییرات خودرا برروی آن اعمال میکند .

UDP:

TRANX B	TIME	TRANX A
Update t	T1	
	T2	RETRIVE t/UPDATE
ROLLBACK/COMMIT	T3	
	T4	

در اینجا تراکنش A نه تنها به تغییر پذیرفته نشده ای در زمان T2 وابسته است بلکه یک به هنگام رسانی را نیز در زمان T3 از دست میدهد .

زیرا برگشت در زمان T3 موجب می شود مقدار قبل از زمان T1 در چندگانه T قرار گیرد .

IAP:

ACC3=30	ACC2=50	ACC1=40
TRANX B	TIME	TRANX A
	T1	RETRIVE ACC1 SUM=40
	T2	RETRIVE ACC2 SUM=90
RETRIVE ACC3	T3	
UPDATE ACC3 30→20	T4	
RETRIVE ACC1	T5	
UPDATE ACC1 40→50	T6	
COMMIT	T7	
	T8	RETRIVE ACC3 SUM=110
	T9	

- تراکنش a موجودی حساب ها را باهم جمع میکند و تراکنش b در حال انتقال مقدار 10 از حساب 3به 1 است . نتیجه تولید شده به وسیله a یعنی 110 مشخصا نادرست است . اگر A بخواهد این نتیجه را در بانک اطلاعاتی بنویسد مسلما بانک را بر وضیعیت ناسازگاری قرار می دهد . در این حالت میگوییم که A یک وضعیت ناسازگار از بانک را دیده لذا یک تحلیل ناسازگار را انجام داده است .

زمانی که یک تاپل ،فیلد یا داده ای مشترک بخواهد مورد تغییر قرار گیرد 4 حالت پیش می اید :

TUPLE T

TRANX A

TRANX B

حالت1 (RR):

Tranx A read tuple t

Tranx B read tuple t

حالت 2(rw): IAP

Tranx A read tuple t

TranxB is allowed write tuple t

حالت 3(Wr):Udp

به این حالت dirty read نیز گفته میشود.

Tranx A write tuple t

TranxB is allowed read tuple t

حالت 4(ww):lup

به این حالت dirty write نیز گفته میشود.

Tranx A write tuple t

TranxB is allowed write tuple t

قفل گذاری :

مکانیزمی که برای جلوگیری از تداخلات تراکنشها استفاده میشودlocking گفته میشود .

ایده اصلی این روش ساده این است :

وقتی یک تراکنش نیاز دارد از عدم تغییر شی مورد نظر خود تا زمان برگرداندن آن مطمئن شود آن شی را قفل میکند .

دو نوع قفل در سیستم موجود است :

- Exclusive lock(x lock)یا write lock
- Shared lock(s lock) یا read lock

اگر تراکنش A چندگانه t را به وسیله یک قفل انحصاری X قفل کند آنگاه تراکنش دیگری ماند B نمیتواند بر آن قفل دیگری ایجاد کند .

اگر تراکنش A یک قفل اشتراکی S را بر روی چندگانه tاعمال کند آنگاه :

تراکنش دیگری مانند B نمی تواند یک قفل X را بر روی این چندگانه اعمال کند .

تراکنش دیگری مانند B می تواند قفل S را بر روی چندگانه t اعمال کند .

این قواعد را میتوان به وسیله یک ماتریس سازگاری نشان داد .

	x	s	
Y	N	y	s
Y	N	N	x
Y	Y	Y	

Compatiability matrix”"

تعریف data access process or locking protocol:

- اگر تراکنش هدفش بازیابی است نیاز است که قفل s را روی آن آبجت ایجاد کند .

S lock for retrieving

- اگر تراکنش قصد update دارد باید قفل x را ایجاد کند .

X lock for update

- اگر درخواست جواب داده نمیشود در آن صورت تراکنش به حالت انتظار میروند تا زمانی که منابع درخواست آزاد شوند در اولین فرصت تراکنش B انجام میشود . اما این انتظار نباید ابدی یا طولانی باشد در این صورت به آن LIVELOCK یا قحطی زدگی میگویند .

Tranx A holds lock

Tranx B goes “wait”

- زمانی که تراکنش به commit یا rollback برسد همه قفلهایش آزاد می شود .

به این پروتکل **“strict two phase locking”** نیز گفته می شود.

مثال:

TRANX B	TIME	TRANX A
	T1	(s)RETRIVE t
RETRIVE t(s)	T2	
	T3	Update t (Requiest x lock) wait
Update t (Requiest x lock) wait	T4	

در اینجا مشکل **قحطی زدگی** داریم .

مثال 2:

TRANX B	TIME	TRANX A
Update t(x)	T1	
	T2	RETRIVE t/UPDATE (Requiest x lock) wait
ROLLBACK/COMMIT Release all lock	T3	
	T4	Resume:tranx A aquire s lock t

مثال3

ACC3=30	ACC2=50	ACC1=40
TRANX B	TIME	TRANX A
	T1	RETRIVE ACC1(s) SUM=40
	T2	RETRIVE ACC2(s) SUM=90
RETRIVE ACC3(s)	T3	
UPDATE ACC3(x) 30→20	T4	
RETRIVE ACC1(s)	T5	
UPDATE ACC1 40→50 (Requiest x lock) wait	T6	
	T7	Read acc3 (Requiest s lock) wait
waite	T8	

اینجا نیز مشکل **قحطی زدگی** به وجود می آید .

تشخیص DEADLOCK:

اگر بن بستی روی دهد بهتر است که سیستم ا=آن را تشخیص داده و رفع کند . د رحقیقت ،تشخیص بن بست شامل تشخیص یک چرخه در گراف انتظار (wait-for-graph) است –یعنی گرافی که تشخیص میدهد چه کسی منتظر چه کس دیگری است ؟- . رفع یک شامل انتخاب یکی از تراکنشهای بن بست شده به عنوان قربالی و برگشت آن است که در این صورت ،قفل های مربوطه به آن باز می شوند و سایر تراکنشها می توانند به کار خود ادامه دهند.

توجه :تراکنش قربانی بدون این که خطایی داشته باشد با شکست مواجه می شود .

پیشگیری از بن بست (deadlock avoidance):

دو مکانیزمی که روی قفل گذاری دو فازی برای جلوگیری از بن بست استفاده میشود:

- Wait-die
- Wound-wait

در ابتدا برای هر تراکنش یک مهر زمانی (time-stamp) در نظر میگیریم این زمان یکتاست و زمان شروع تراکنشها میباشد.

Tranx A request resourses which are hold by tranx B

- در این صورت داریم :

Wait-die

اگر A قدیمی تر باشد به حال انتظار می رود وگرنه به حالت ROLLBACK

WOUND

اگر A جوان ترباشد منتظر می ماند وگرنه تراکنش B به حالت WOUND میرود.

❖ قابلیت سریال سازی:(serializability)

گفته شد اگر تراکنشی قفل x روی یک شی داشته باشد هیچ تراکنش دیگری نمی تواند برروی آن یک قفل ایجاد کند . سپس قراردادی رابری به کارگیری این قفلها مطرح کردیم که ما را از عدم بروز مشکلات به هنگام رسانی مفقود شده و سایر مشکلات مطمئن می سازد "برروی انچه که قابل بازیابی است یک قفل s و برروی آنچه که قابل به هنگام رسانی است یک قفل x اعمال و تمام این قفلها را تاپایان تراکنش حفظ کنید" این قرارداد،قابلیت سریالسازی را فراهم میکند.

این قرارداد شکل کامل از قرارداد قفل گذاری دومرحله ای است .

❖ قفل گذاری دومرحله ای :(2plp)

اگر تمام تراکنشها از این قرارداد پیروی کنند،آنگاه تمام برنامه های زمانی قابل سریالسازی هستند.

یک برنامه زمانی قابل سریال سازی بیان می کند که اگرAوB دو تراکنش د رآن برنامه زمانی باشند ،آنگاه A می تواند خروجی B را ببیند و باالعکس . متاسفانه قرارداد قفل گذاری دو مرحله ای ممکن است منجر به بن بست شود. برای برطرف کردن بن بست ها باید یکی از تراکنشهای بن بست شده را به عنوان قربانی انتخاب کرد و آن را برگشت داد (در نتیجه تمام قفلهای آن باز میشوند.)

به فاز اول (growing) و به فاز دوم (shrinking) گفته می شود.

Cascade rollback:

در آپدیت های ثبت نشده باید به اجبار تراکنشهای اجرا شده روی آن منبع را هم کنسل کرد. به این کنسل کردن کنسل کردن آبشاری گفته می شود.
برای رفع آن باید حتما از قانون قفل گذاری استفاده کرد.

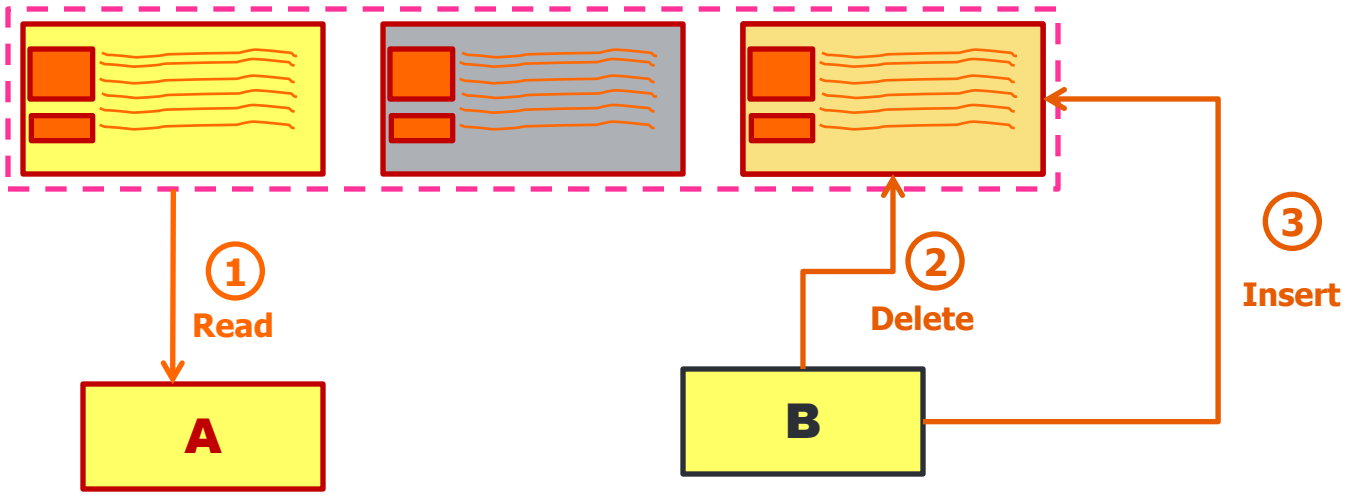
پدیده "phantom":

فرض کنید تراکنش T1 مجموعه ای از تمام سطرهایی که در شرط خاصی صدق میکنند را بازیابی می کند (برای مثال سطرهای تمام عرضه کنندگانی که شهر آنها پاریس است) . فرض کنید تراکنش T2 سطر جدیدی که دران شرط صدق میکند را درج میکند. اکنون تراکنش T1 تقاضای بازیابی خودرا تکرار میکند سطری را مشاهده میکند که قبلا وجود نداشته است این سطر را "شبح" گویند .

سوال :سیستم چگونه میتواند از بروز "شبح" جلوگیری کند؟

باید مسیر دستیابی استفاده شده برای رسیدن به داده مورد نظر را قفل کند . برای نمونه ،در مثال ارائه شده در بالا که مربوط به عرضه کننده شهر پاریس بود ،ار آن مسیر دستیابی ،یک شاخص بر روی شهرهای عرضه کنندگان باشد آنگاه سیستم باید درایه پاریس را در آن شاخص قفل گذاری کند. چنین قفلی از ایجاد اشباح جلوگیری میکند زیرا ایجاد فوق نیازمند این است که مسیر دستیابی به هنگام رسانی شود .

- راه حل



سطوح جداسازی :

اصطلاح سطح جداسازی به معنی درجه تداخلی است که تراکنش خاصی باید برروی بخشی از تراکنشهای همزمان داشته باشد . اکنون اگر لازم باشد قابلیت سریال سازی تضمین شود اصلا هیچ میزای از تداخل قابل قبول نیست . به عبارت دیگر سطح جداسازی باید به میزان حداکثر باشد. سطح جداسازی را به عنوان یک ویژگی عمده از تراکنش د رنظر میگیریم . حداقل میتوان 5 سطح در نظر گرفت . به طور کلی سطح جداسازی بالاتر منجر به تداخل کمتر (و همزمانی پایین تر)سطح جداسازی پایین تر منجر به تداخل بیشتر (و همزمانی بالاتر) می شود .

DB2 از دو سطح بیشتر پشتیبانی نمیکند که عبارتند از :

1-پایداری مکان نما 2- خواندن قابل تکرار

- خواندن قابل تکرار (RR) سطح ماکزیمم است . اگر تمام تراکنشها دراین سطح عمل کنند آنگاه تمام برنامه های زمانی قابل سریال سازی هستند .
- درمقابل پایداری مکان نما (CS) اگر تراکنش T1
- قابلیت آدرس دهی به چندگانه t را به دست آورد و در نتیجه
- بر روی t قفل گذاری کند و سپس
- قابلیت آدرس دهی به چند گانه t را بدون به هنگام رسانی آن را از دست بدهد بنابراین
- قفل آن را به سطح X ارتقا نداده انگاه
- آن قفل را میتوان بازکرد بدون انتظار رسیدن به پایان تراکنش .

INTENT lockig:

تا کنون فرض می کردیم که واحد کاری برای اهداف قفلگذاری فقط یک چندگانه است اما در اصل دلیلی وجود ندار که قفلها نباید برروی واحدهای بزرگتر یا کوچکتر داده ها مانند یک متغیر رابطه ای یا حتی کل بانک اطلاعاتی و یا یک جز مشخص از یک چندگانه اعمال شوند .

. از درجه بندی قفل گذاری (Locking granularity) یک مصالحه وجود دارد :مسلما درجه بندی بهتر موجب همزمانی بیشتر میشود و هرچه درجه درشتر باشد به تنظیم و امتحان قفلهای کمتری نیاز است و لذا سربار کاهش می یابد .

انواع قفلها عبارتند از:

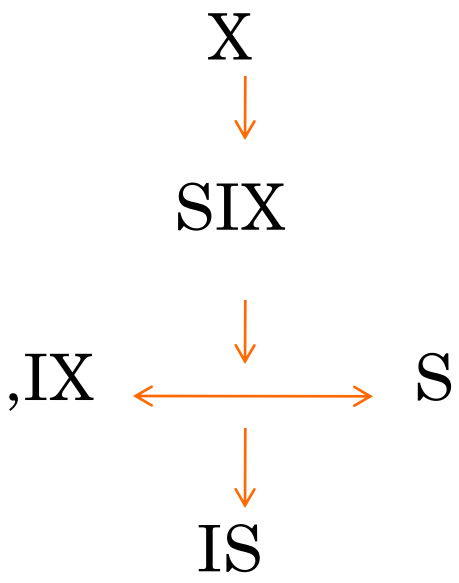
- اشتراکی دقیق (IS)
- انحصاری دقیق (IX)
- انحصاری دقیق اشتراکی (SIX)
- IS: T تمایل دارد قفلهای S را بر روی تک تک چندگانه های موجود R اعمال کند تا پایداری چندگانه ها را تازمانی که پردازش می شوند تضمین کند .
- IX: همانند IS است . به علاوه این که T ممکن است تک تک چندگانه های موجود در R را به هنگام رسانی کند و لذا قفلهای X برروی آن چندگانه ها اعمال کند .
- SIX : ترکیبی از S و IX است . یعنی T می تواند چندین فرآیند خواندن همزمان را انجام دهد ولی نمیتواند چندین به هنگام سازی همزمان را درR انجام دهد به علاوه این که T ممکن اس تک تک چندگانهای موجود در R را به هنگام رسانی کند و لذا قفلهای X را بر روی آن چند گانه ها اعمال کند .

قرارداد قفل گذاری دقیق:

- قبل از اینکه تراکنش بتواند یک قفل S را بر روی یک چندگانه اعمال کند ابتدا باید متغیر رابطه ای در برگیرنده آن چند گانه را به وسیله یک قفل IS یا قوی تر از آن قفل کند .

- قبل از اینکه یک تراکنش بتواند یک قفل X را بر روی یک چندگانه اعمال کند ابتدا باید متغیر رابطه ای در برگیرنده آن چندگانه را به وسیله قفل IX یا قویتر قفل کند .

نکته :ترتیب قفلها از لحاظ قدرت



-	IS	S	IX	SIX	X	
Y	N	N	N	N	N	X
Y	Y	N	N	N	N	SIX
Y	Y	N	Y	N	N	IX
Y	Y	Y	N	N	N	S
Y	Y	Y	Y	Y	N	IS
	Y	Y	Y	Y	Y	-

پایگاه داده توزیع شده :(distributed database)

یک سیستم از پایگاه داده توزیع شده است که داده ها به طور مجزا در سایتهای مجزا نگهداری میشوند .

- هرکدام از آنها یک DBMS و سخت افزار وسیستم عامل و تحت شبکه های مختلفی اجرا میشوند.
- یک نمونه سیستم توزیع شده مانند client/server می باشد.

- یک پایگاه داده توزیع شده مجموعه ای از سایت هاست که از طریق شبکه های ارتباطی به هم متصل شده اند به نحوی که هر سایت یک دیتابیس کامل درنوع خود است . کاربری که با پایگاه داده توزیع شده کار میکند تمام عملیات و پخش اطلاعات ،چگونگی مدیریت .... از دید کاربر خارجی مخفی است .

- بخشی از DBMS است که این مدیریت توزیع شدگی را حمایت میکند.
- پایگاه داده توزیع شده از نظر پراکندگی جغرافیایی متنوع هستند .

مزایا :

- به دلیل اینکه ماهیت داده ها متفاوت بود به سمت توزیع شدگی سوق داده شد.
- اطلاعات به صورت محلی نگهداری می شود و امکان دسترسی به اطلاعات محلی به سادگی انجام می شود .

معایب :

- مدیریت آنها از لحاظ تکنیکی ممکن است پیچیده باشد که این معایب در سطح طراحان مشهود است .
- عملیات دستکاری در دونوع پایگاه داده یکسان است و عملیات تعریف بسط یافته تعریف در سیستمهای متمرکز است .

اصول طراحی پایگاه داده توزیع شده :

- Local outonomy
- No reliance on central site
- Continuos operations
- Location independence(transparency)
- Fragmention independence
- Replication independence
- Distributed query processing
- Distributed transacation management
- Hardware independence
- Os independence
- Network independence
- DBMS independence

Local outonomy

- قانون 1: استقلال محلی سایتها باید تا حد امکان(بیشترین حد ممکن) مستقل باشند. داده های محلی باید در محل ذخیره و مدیریت شوند(با توجه به در نظر گرفتن یکپارچگی و امنیت) عملیات محلی باید حتما در خود محل اجرا شوند. تمام عملیات در یک سایت باید توسط همان سایت کنترل شود. این بدین معناست که سایت X نباید برای انجام موفقیت آمیز عملیات خود وابسته به سایت Y باشد. در برخی موارد، از دست دادن مقدار کمی از استقلال، اجتناب ناپذیر است: § مشکل قطعه قطعه شدن(قانون 5) § مشکل Replication(قانون 6) § به روز رسانی رابطه Replicateشده(قانون 6) § مشکل محدودیت یکپارچگی بین چند سایت(قانون 7 2 a in participation problem A § (phase commit process(قانون 8)

No reliance on central site

- قانون 2: عدم وابسته بودن به سایت مرکزی به هیچ عنوان نباید برای یک سرویس مرکزی به یک سایت وابسته بود. بعنوان مثال نباید دارای یک پردازشگر مرکزی(متمرکز) جستجوها یا مدیریت مرکزی(متمرکز) Transactionبود، چرا که کل سیستم به یک سایت خاصی وابسته می شوند. وابسته بودن به یک سایت خاص، حداقل به دو دلیل زیر غیر مطلوب می باشد: § سایت مرکزی ممکن است یک گلوگاه (Bottleneck)باشد. § سیستم ممکن است آسیب پذیر باشد. در یک سیستم توزیع شده، عملیات زیر (در میان سایر عملیات) حتما باید توزیع شده باشند § :مدیریت دیکشنری § پردازش جستجو § کنترل همزمان § کنترل بازیابی

- وابسته بودن به یک سایت خاص، حداقل به دو دلیل زیر غیر مطلوب می باشد:

§ سایت مرکزی ممکن است یک گلوگاه (Bottleneck)باشد.

§ سیستم ممکن است آسیب پذیر باشد.

در یک سیستم توزیع شده، عملیات زیر (در میان سایر عملیات) حتما باید توزیع شده باشند :

مدیریت دیکشنری § پردازش جستجو § کنترل همزمان § کنترل بازیابی

Continuos operations

- قانون 3: عملیات پیوسته هیچگاه نباید نیاز به خاموش کردن (از قبل پیش بینی شد)ه کل سیستم برای اعمال تغییرات داشته باشیم. اضافه کردن سایت جدید X به سیستم توزیع شدهD ، نباید باعث توقف کل سیستم شود. اضافه کردن سایت جدید X به سیستم توزیع شدهD ، نباید نیازمند تغییری در برنامه های کاربر یا فعالیتهای ترمینال باشد. حذف سایت X از سیستم توزیع شده، نباید ئقفه های غیر ضروری در سرویس ایجاد کند. ایجاد و حذف و تکثیر قطعات به صورت پویا باید در یک سیستم توزیع شده امکان پذیر باشد. باید بتوان بدون نیاز به خاموش کردن کل سیستم، DBMSیک سایت را به روز کرد.

LOCATION INDEPENDENCE(TRANSPARENCY)

قانون 4: استقلال Location نه تنها کاربران نباید از محلی فیزیکی ذخیره داده ها مطلع باشند، بلکه از لحاظ منطقی باید به تصور کنند که داده ها در سایتهای محلی خودشان قرار دارد. ساده کردن برنامه های کاربر و فعالیتهای ترمینال اجازه تغییر سکو فراهم کردن استقلال Location برای عملیات ساده بازیابی ساده تر از عملیات به روز رسانی می باشد. داشتن طرحی برای نام گذاری داده توزیع شده (Distributed Data Naming Scheme)و ایجاد پشتیبانی مناسب از طریق زیر سیستم دیکشنری مواردی که باید در مورد کاربران پیاده سازی شود: § کاربر Uباید شناسه معتبری برای ورود در سایتهای مختلف داشته باشد. § پروفایل هر کاربر برای هر شناسه مجاز باید در دیکشنری باشد. دسترسی های هر کاربر در هر سایت به وی اختصاص داده شود.

FRAGMENTION INDEPENDENCE

- قانون 5: استقلال قطعات (Fragmentation)سیستمهای توزیع شده از قطعه قطعه شدن داده ها پشتیبانی می کنند، منوط به اینکه یک رابطه خاص قابلیت تقسیم به قسمتهای مختلف برای ذخیره در محلهای فیزیکی گوناگون را داشته باشد .سیستمی که این قابلیت را داشته باشد، از استقلال قطعات نیز پشتیبانی می کند. کاربران باید از لحاظ منطقی به گونه ای تصور کنند که گویا اصلا داده ها در قسمتهای مختلف ذخیره نشده اند. از دلایل قطعه قطعه شدن داده ها، می توان به افزایش کارآیی اشاره کرد. قطعه قطعه شدن افقی (Select)قطعه قطعه شدن عمودی (Project) قطعه قطعه شدن باید در متن یک پایگاه داده توزیع شده تعریف شود. استقلال قطعات همانند استقلال Location باعث ساده تر شدن برنامه های کاربر و فعالیتهای ترمینال می شود. داده هایی که به کاربران نمایش داده می شود، از ترکیب منطقی قطعات مختلف (به واسطه الحاقها (Joins)و اجتماعات (Unions)مناسب)به دست می آید.

REPLICATION INDEPENDENCE

- قانون 6: استقلال Replication کاربران باید از لحاظ منطقی به گونه ای تصور کنند که گویا اصلا داده ها تکرار (replicated)نشده اند. سیستم توزیع شده از کپی برداری دادها پشتیبانی می کند، به شرط آن که یک رابطه(یا بطور کلی تر یک قطعه از رابطه) بتواند از لحاظ فیزیکی در کپی های مجزا و در سایتهای مجزا ذخیره شود. کپی برداری داده ها باید همانند قطعه قطعه شدن برای کاربران شفاف(غیر قابل تشخیص) باشد. دلایل عمده کپی برداری داده ها § کارآیی § در دسترس بودن(دسترسی) مشکل انتشار به روز رسانی استقلال Replication همانند استقلال قطعات و استقلال Location باعث ساده تر شدن برنامه های کاربر و فعالیتهای ترمینال می شود. رو نوشت از داده ها (Snapshots)

DISTRIBUTED QUERY PROCESSING

قانون 7: پردازش توزیع شده جستجوها یکی از مهمترین و حیاتی ترین نکات در مرود سیستمهای پایگاه داده توزیع شده، انتخاب استراتژی مناسب برای پردازش توزیع شده جستجو (Query)می باشد. پردازش جستجو در سیستم های توزیع شده شامل موارد زیر می باشد: عملیات محلی ورودی و خروجی (I/O)و CPU در سایتهای مجزا تبادل اطلاعات میان سایتهای فوق الذکر Query Compilation Ahead Of Time Views That Span Multiple Sites integrity constraints that within DDBS that span multiple sites

DISTRIBUTED TRANSACATION MANAGEMENT

- قانون 8: مدیریت توزیع شده Transaction دو نکته مهم برای مدیریت Transaction، کنترل بازیابی (Recovery Control)و کنترل سازگاری (Consistency Control)می باشد که نیاز به اعمال و دقت بیشتری در محیط های توزیع شده دارند. در یک سیستم توزیع شده، یک Transaction می تواند باعث اجرای کد در چندین سایت شده که همین امر خود می تواند باعث عملیات به روز رسانی در سایتهای مختلف شود. هر Transaction را می توان شامل چندید Agentدر نظر گرفت که هرAgent ، فرآیندی است که از طرف Transactionدر سایت به خصوصی اجرا می شود.

بن بست عمومی:

هیچ سایتی نمی تواند با استفاده از اطلاعات داخلی خود، آن را تشخیص دهد.

Hardware independence

قانون 9 :استقلال سخت افزاری • صرفه نظر از اینکه چه Platform سخت افزاری استفاده می شود، کاربران باید تصویر واحدی از سیستم داشته باشند. • بهتر است بتوان یک DBMS را بر روی سیستمهای سخت افزاری مختلف اجرا کرد. • بهتر است سیستم های مختلف سخت افزاری سهم یکسانی در یک سیستم توزیع شده داشته باشند. • نمی توان به راحتی فرض کرد که همواره می توان از سیستمهای همگن استفاده کرد، به همین دلیل هنوز باید یک DBMS بر روی سیستمهای مختلف سخت افزاری قابل اجرا باشد.

OS INDEPENDENCE

قانون 10: استقلال سیستم عامل • بهتر است که علاوه بر استقلال سخت افزاری، قادر به راه اندازی DBMS بر روی سیستم عاملهای مختلف (حتی سیستم عاملهای مختلف بر روی یک سخت افزار) باشیم. • حداقل سیستم عاملهای مهمی که باید DBMS پشتیبانی کند(با توجه به معیارهای تجاری)، عبارتند ازMVS/XA :؛ MVS/ESA، VM/CMS، VAX/VMS،)UNIXمحصولات مختلف(، OS/2، MS/DOSوWINDOWS

NETWORK INDEPENDENCE

قانون 11: استقلال شبکه • مطلوب آن است که بتوانیم شبکه های نامتجانس مختلف را پشتیبانی نماییم. • از دید یک DBMS توزیع شده، شبکه یک سرویس مطمئن انتقال پیغام می باشد. • مفهموم مطمئن در عبارت فوق را می توان بدین صورت توصیف نمود که به طور مثال اگر شبکه پیغامی را از سایت X برای تحویل به سایت Y دریافت کرد، سرانجام آن پیغام را به سایت Y تحویل دهد. • نباید در محتوای پیغامها خللی ایجاد شده و پیغامها باید به ترتیب فرستاده شدن ارسال شده و بیش از یکبار نیز تحویل مقصد نشوند. • شبکه مسئول تایید سایت (Site Authentication)نیز می باشد. • یک سیستم ایده آل باید هم از شبکه های محلی (LAN)و هم از شبکه های گسترده (WAN)پشتیبانی نماید • .سیتمهای توزیع شده باید معماریهای مختلف شبکه را پشتیبانی نمایند.

DBMS INDEPENDENCE

قانون 12:استقلال DBMS سیستم توزیع شده ایده آل باید استقلال DVBMSرا مهیا سازد.


8BeheshtGroup.com
آدم‌های موفق به اندیشه‌هایشان عمل می‌کنند
اما سایرین تنها به سختی انجام آن می‌اندیشند!
موفق و موید باشید
8BeheshtGroup
Prosperous people put their thoughts and plans into action,
but others only think about how laborious it's going to be!