موسسه بابان

انتشارات بابان و انتشارات راهیان ارشد

پایگاه داده‌ها

فصل اول: مفاهیم پایه (اسلاید)

(ویژه کنکور ارشد ۱۴۰۵)

ویژه‌ی داوطلبان کنکور کارشناسی ارشد مهندسی کامپیوتر و IT

براساس کتب مرجع

آبراهام سیلبرشاتز، راما کریشنان، رامز المصری، سی جی دیت، توماس کانلی و کارولین بگ

ارسطو خلیلی‌فر

تراکنش

به عملیات مجاز **درج**، **حذف** و **آپدیت** که تحت کنترل DBMS اســت **تراکنش** گفته می‌شــود که باید خواص **ACID** را رعایت کند.

خواص تراکنش (ACID)

۱- تجزیه‌ناپذیری (Atomicity)

اگر یک تراکنش شــروع به **پیشــروی** کرد باید **همه** دســتورات آن به طور کامل انجام شــود، یا **هیچکدام** انجام نشود.

تضمین خاصیت تجزیه‌ناپذیری تراکنش بر عهده DBMS است.

مثال: درج یک رکورد

```
BEGIN TRANSACTION;

INSERT INTO S (S#, Sname, City)  //DML
VALUES (S4, Sn4', C3');

END TRANSACTION;  //TCL


COMMIT; // Save to hard disk
```

توجه: در SQL برای پرس‌وجوهای تک خطی BEGIN و END به طور خودکار درج می‌شود.

مثال: مقادیر حساب‌های A و B به ترتیب 1000 و 2000 واحد پولی باشد.

T_i: read(A);
$A := A - 50$;
write(A);
read(B);
$B := B + 50$;
write(B).

الف) بخش اول (برداشت وجه)

ب) بخش دوم (واریز وجه)

BEGIN TRANSACTION;

① UPDATE accounts SET balance = balance - 50 WHERE user_id = 'A';
② UPDATE accounts SET balance = balance + 50 WHERE user_id = 'B';

END TRANSACTION;

COMMIT; // Save to hard disk

توجه: اگر هر دو دستور **UPDATE** با موفقیت اجرا شود، دستور **COMMIT** فراخوانی شده و تغییرات به صورت **دائمی روی دیسک** ذخیره می‌شود. که در این حالت مقدار A برابر 950 و مقدار B برابر 2050 خواهد بود.

توجه: اگر در حین اجرای تراکنش مشکلی پیش آید (مثلاً به دلیل نقض محدودیت‌های جامعیتی یا خطای سیستمی بخش اول تراکنش انجام شود اما بخش دوم انجام نشود که در این حالت مقدار A برابر 950 و مقدار B برابر همان 2000 خواهد بود)، آنگاه دستور **ROLLBACK** **به طور خودکار** فراخوانی شده و تمام تغییرات **لغو** می‌شود.

توجه: در شروع و پایان یک تراکنش سیستم باید سازگار باشد ولی در اثنای اجرای تراکنش ممکن است **موقتاً** نیاز به ناسازگاری باشد. برای مثال هنگام واریز پول از حساب A به B، پس از برداشت پول از حساب A ، سیستم **به‌طور موقت ناسازگار** است و پس از واریز آن به حساب B دوباره سیستم **سازگار** می‌شود. لذا برنامه برداشت از حساب A یا واریز به حساب B به تنهایی تراکنش نیست.

توجه: دستور شروع تراکنش BEGIN TRANSACTION یا START TRANSACTION معمولاً اولین دستوری است که برای آغاز یک تراکنش صادر می‌شود. در هنگام اجرای این دستور، DBMS به طور خودکار یک شناسه منحصر به فرد Transaction ID یا TID به آن تراکنش اختصاص می‌دهد. DBMS با اختصاص یک شناسه منحصر به فرد به هر تراکنش، امکان مدیریت و کنترل مؤثر بر روی تراکنش‌ها را فراهم می‌کند.

تست – در کدام‌یک از دستورات تراکنش، DBMS یک شناسه منحصر به فرد به آن تراکنش نسبت می‌دهد؟ (مهندسی کامپیوتر- آزاد ۹۳)

۱) start ۲) abort و commit ۳) فقط commit ۴) input و outout

پاسخ – گزینه (۱) صحیح است.

با ایجاد و شروع هر تراکنش، یک شناسه توسط DBMS به آن نسبت داده می‌شود.

۲- سازگاری (Consistency) هماهنگ

مثال: انسان **سازگار (هماهنگ)** با قوانین طبیعت

مثال: داده **سازگار (هماهنگ)** با قوانین پایگاه داده

اگر یک تراکنش شـــروع به **پیشـــروی** کرد، در **حین** و **میانه** مســـیر نباید **محدودیت‌های جامعیتی داخلی و خارجی** (Integrity Constraints) را نقض کند.

محدویت‌های امنیتی (Security constraints) : کاربران مجاز

محدویت‌های جامعیتی (Integrity constraints) : عملیات مجاز

به طور کلی **محدودیت‌های جامعیتی** (Integrity Constraints) در **سیستم‌های بانکی** به دو طبقه‌ی **جامعیت داخلی** و **جامعیت خارجی** **(قواعد کسب و کار یا Business rules)** تقسیم می‌گردد.

قوانین جامعیت داخلی

۱- **قانون جامعیت درون رابطه‌ای** (Intra-Relational Integrity Rule)

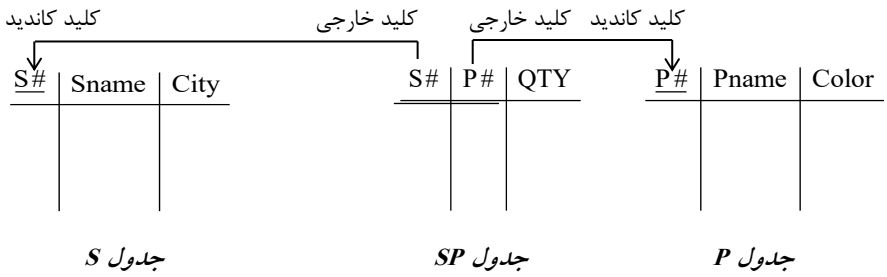
۲- **قانون جامعیت موجودیت** (Entity Integrity Rule)

۳- **قانون جامعیت دامنه‌ای** (Domain Integrity Rule)

۴- **قانون جامعیت ارجاعی** (Referential Integrity Rule)

قوانین جامعیت خارجی

مثال: درج مقدار منفی در حساب بانکی، یا درج نمره 22 خارج از بازه تعریف شده 0 تا 20



کارکرد قطعه

Create Table S (S# char (5), Sname char (20), City char (15), Primary key (S#))	Create Table SP (S# char (5), P# char (5), QTY numeric (10), Primary key (S#, P#), Foreign key (S#) References S(S#) on delete cascade on update cascade, Foreign key (P#) References P(P#) on delete cascade on update cascade, Check (QTY>1 AND QTY<1000))	Create Table P (P# char (5), Pname char (20), Color char (10), Primary key (P#))

مثال تک دستور: درج یک رکورد

```
BEGIN TRANSACTION;

INSERT INTO S (S#, Sname, City)  //DML
VALUES (S4, Sn4', C3');

END TRANSACTION;  //TCL


COMMIT; // Save to hard disk
```

دستورات SQL به چهار دسته زیر تقسیم می‌گردد:

۱ـ دستورات تعریف داده‌ها (DDL: Data Definition Language)

کارهای مربوط به ظرف‌سازی و ساختارسازی، مثل ایجاد و حذف جداول، دیدها و شاخص.

مثال: دستور **Create Table** برای ایجاد جداول و دستور **Drop Table** برای حذف جداول

۲ـ دستورات تغییر داده‌ها (DML: Data Manipulation Language)

کارهای مربوط به تغییرات محتوای جداول، مثل ذخیره و بروزرسانی داده‌ها در جداول و بازیابی و حذف داده‌ها از جداول

۳ـ دستورات کنترل داده‌ها (Data Control Language : DCL)

کارهای مربوط به امنیت جداول، مثل ایجاد سطوح دسترسی برای کاربران نهایی

مثال: دستور **Grant** برای ایجاد حق دسترسی و دستور **Revoke** برای حذف حق دسترسی

۴ـ دستورات کنترل تراکنش‌ها (TCL: Transaction Control Language)

کارهای مربوط به کنترل تراکنش‌ها، مثل **تثبیت** و **لغو** تراکنش‌ها. این دستورات در واقع تغییرات اعمال شده توسط دستورات DML را اداره می کند. SQL هیچ یک از دستورات DML را ماندگار نمی‌کند، مگر اینکه به طور کامل اجرا و **Commit** شود.

مثال: دستور **Commit** برای **تثبیت تراکنش** و دستور **Rollback** برای **لغو تراکنش**

مثال چند دستور: مقادیر حساب‌های A و B به ترتیب 1000 و 2000 واحد پولی باشد.

T_i: read(A);
$A := A - 50$;
write(A);
read(B);
$B := B + 50$;
write(B).

الف) بخش اول (برداشت وجه)

ب) بخش دوم (واریز وجه)

BEGIN TRANSACTION;

① UPDATE accounts SET balance = balance - 50 WHERE user_id = 'A';
② UPDATE accounts SET balance = balance + 50 WHERE user_id = 'B';

END TRANSACTION;

COMMIT; // Save to hard disk

قوانین جامعیت خارجی

مثال: جمع دو حساب در پایگاه داده سازگار باید قبل و بعد از اجرای تراکنش، یکسان و برابر 3000 باشد که هست. و به معنی این است که نه مبلغی گم شده و نه مبلغی پیدا شده.

مثال: یا در سیستم حساب بانکی مقدار مربوط به برداشت و واریز وجه برابر نباشد برای مثال مبلغ 50 واحد پولی از حساب برداشت شده ولی مبلغ 70 واریز شده باشد.

توجه: اگر هر دو دستور **UPDATE** با موفقیت اجرا شود، دستور **COMMIT** فراخوانی شده و تغییرات به صورت **دائمی روی دیسک** ذخیره می‌شود. که در این حالت مقدار A برابر 950 و مقدار B برابر 2050 خواهد بود.

توجه: سیستم پایگاه داده قبل از شروع هر تراکنشی در حالت **سازگار** (Consistent) است، یعنی محتوای داخل جداول پایگاه داده با **محدودیت‌های جامعیت داخلی و خارجی** (**Integrity Constraints**) تطابق دارد.

توجه: اگر یک تراکنش انجام شود، نتیجه اجرای تراکنش باید سیستم پایگاه داده را از حالت **سازگار قدیم** به حالت **سازگار جدید** انتقال دهد. و اگر اینچنین نشود همه چیز باید به عقب بازگردد.

توجه: هدف اصلی، **حفظ سازگاری (حفظ هماهنگی)** بانک اطلاعات است.

توجه: به وضعیت **سازگار** بانک اطلاعات، **وضعیت معتبر** (Valid State) نیز گفته می‌شود.

تست- کدام یک از موارد زیر صحت (Integrity) بانک اطلاعاتی را تضمین می‌کند؟

(مهندسی کامپیوتر-دولتی ۱۳۸۶)

۱) افراد مجاز عملیات مجاز روی بانک اطلاعاتی انجام می‌دهند.

۲) افراد مجاز روی بانک اطلاعاتی عملیات انجام می‌دهند.

۳) بانک اطلاعاتی از حالت سازگار آغاز شده و عملیات مجاز روی آن انجام می‌شود.

۴) فقط عملیات مجاز روی بانک اطلاعات انجام می‌شود.

پاسخ- گزینه (۳) صحیح است.

گزینه اول و چهارم **عملیات مجاز** برای رعایت **محدودیت‌های جامعیتی** را بیان کرده اما حالت سازگاری اولیه بانک اطلاعات را مشخص نکرده است. گزینه دوم **افراد مجاز** مربوط به **محدودیت‌های امنیتی** است. گزینه سوم به رعایت محدودیت‌های جامعیتی و سازگاری اولیه تاکید دارد.

تست- روش Cascade در قاعده تمامیت ارجاعی در پایگاه دادها، چه کارکردی دارد؟

(مهندسی کامپیوتر-دولتی۱۴۰۳)

۱) فقط برای به‌روزرسانی استفاده می‌شود و هیچ تاثیری بر حذف رکوردها ندارد.

۲) هنگامی که یک رکورد در جدول مرجع حذف یا به‌روزرسانی می‌شود، فقط تغییرات حذف در جدول‌های مرتبط اعمال می‌شود.

۳) فقط در صورت تغییر مقدار ستون‌های غیرکلید در یک جدول، تغییرات را در جدول‌های دیگر اعمال می‌کند.

۴) هنگامی‌که یک رکورد در جدول مرجع حذف یا به‌روزرسانی می‌شود، تغییرات به‌صورت خودکار در جدول‌های مرتبط با کلید خارجی اعمال می‌شود.

پاسخ- گزینه (۴) صحیح است.

تست - در یک سیستم مدیریت پایگاه داده‌ها (DBMS) کدام یک از امکانات زیر جزء عناصر اصلی تشکیل‌دهنده DMBS محسوب نمی‌شوند؟

(کارشناسی ارشد- دولتی ۷۲)

۱) امکان پردازش زبان طبیعی برای کار با پایگاه

۲) امکان کار با داده‌ها به کمک یک DSL (Data Sub Language)

۳) امکان تأمین جامعیت و بی‌نقصی (integrity) پایگاه

۴) امکان تأمین ایمنی پایگاه

پاسخ - گزینه (۱) صحیح است.

پردازش زبان‌های طبیعی (رویه‌ای یا میزان) مانند زبان C و پاسکال بر عهده کامپایلر و پردازش زبان‌های بیانی (میهمان) مانند SQL بر عهده DBMS (SQL Server) است.

تست- چند عبارت از عبارات زیر درست است؟

(مهندسی کامپیوتر-دولتی۱۴۰۲)

- زبان سطح پایین دستکاری داده‌ها (Low Level DML) باید در یک زبان برنامه همه منظوره (General purpose Language) نهفته شود.
- مدل داده‌ای فیزیکی (Physical Data model) مفاهیمی را فراهم می‌کند که توسط کاربران نهایی (End Users) به راحتی قابل فهم باشند.
- مدل داده‌ای (Data Model) ابزاری برای حصول تجرید داده‌ها (Data Abstraction) است.
- DBMS مسئولیت کامل اینکه در هر لحظه پایگاه داده‌ها در وضعیت معتبر (Valid State) باشد را برعهده دارد.

لازم به ذکر است وضعیت معتبر، وضعیتی است که تمام محدودیت‌ها و شرایط و ساختارهای تعریف شده در شمای پایگاه داده را ارضا کند.

۱) صفر

۲) 1

۳) 2

۴) 3

پاسخ- گزینه (۳) صحیح است.

گزاره اول درست است. زیرا، DML چه زبان سطح پایین دستکاری داده‌ها (Low Level DML) باشد و چه زبان سطح بالای دستکاری داده‌ها (High Level DML) باید در یک زبان برنامه همه منظوره (General purpose Language) یا زبان‌های روالی متعارف یا سطح بالا نهفته شود.

گزاره دوم نادرست است. زیرا، کاربران نهایی (End Users) در بالاترین سطح انتزاع هستن و از مدل داده‌ای فیزیکی (Physical Data model) و جزییات مربوط به رسانه و چگونگی ذخیره و بازیابی اطلاعات باخبر نیستن. این مورد توسط استقلال فیزیکی داده‌ها محقق شده است.

گزاره سوم درست است. زیرا، مدل ساده شده یک واقعیت است و این ساده شدن یعنی حذف جزئیات و حذف جزئیات یعنی انتزاع (تجرید).

گزاره چهارم نادرست است. زیرا، DBMS مسئولیت حفظ وضعیت معتبر (Valid State) پایگاه

داده‌ها را در «**اکثر**» مواقع برعهده دارد و نه به طور کامل و در هرلحظه، چون برای مثال در DBMS لحظاتی هست که مبلغی از حسابی کسر شده ولی هنوز به حساب مقصد واریز نشده است و در این میانه، سیستم تا لحظاتی نامعتبر است تا سرانجام در پایان تراکنش به وضعیت معتبر برسد.

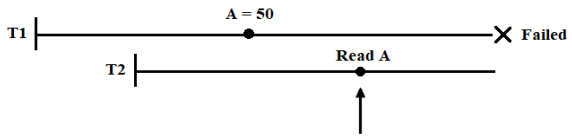
۳- انزوا یا جداسازی (Isolation)

اگر یک تراکنش شروع به **پیشروی** کرد، در حین مسیر فقط باید قادر باشد درصورت لزوم از **نتایج نهایی** تراکنش‌های هم‌روند دیگر استفاده کند و نباید قادر باشد از **نتایج میانی** تراکنش‌های هم‌روند دیگر استفاده کند.

توجه: راه انزوای تراکنش‌ها این اســـت که **هر تراکنش** خودش **عایق شـــود** تا نتایج میانی آن به تراکنش‌های هم‌روند دیگر **درز** نکند. راه عایق‌سازی یک تراکنش این است که داده‌های اشتراکی به عنوان **ناحیه بحرانی** تلقی شود و برای آن **انحصار متقابل** برقرار شود، بدین معنی که در هر لحظه به هر داده اشتراکی فقط و فقط یک تراکنش دسترسی داشته باشد.

توجه: تراکنش‌های هم‌روند، الزاما از نظر زمانی نقطه شروع و پایان یکسان ندارند، اما در مقطعی از زمان به صورت موازی اجرا می‌شوند.

مثال: دو تراکنش T1 و T2 را به صـــورت هم‌روند در نظر بگیرید. فرض کنید خاصـــیت Isolatin برای آنها برقرار نیست و به تبع دو تراکنش می‌توانند از **نتایج میانی** یکدیگر قبل از commit شدن هر **داده مشترک** استفاده کنند.



فرض کنید مقدار اولیه **داده مشـــترک** A برابر 100 اســـت. و تراکنش T1 مقدار آن را به 50 آپدیت می‌کند. سپس تراکنش T2 بدون commit شدن T1 مقدار داده مشترک A را استفاده می‌کند. فرض کنید در ادامه تراکنش T1 به دلیل قطعی برق Failed شـــود، بنابراین چون T1 به طور کامل تمام نشـده اسـت، مطابق خاصـیت Atomicity تغییرات آن Rollback می‌شـود، پس مقدار A به 100 باز می‌گردد. این درحالی اســـت که T2 کارهای خود را با مقدار 50 انجام داده اســـت و نتایج آن قابل قبول نیست. در این مثال خاصیت Isolation نقض شده است، چون T2 توانسته از نتایج میانی T1 استفاده کند.

توجه: اگر در یک تراکنش **منفرد** اصل **Atomicity** برقرار باشد و همچین اصل **Consistency** نیز برقرار باشد، تضمینی نیست که همین تراکنش به طور هم‌روند در مقابل تراکنش دیگر اصل **Isolation** برای آن برقرار باشد. اگر به تنهایی عالی باشید تضمینی نیست که در همکاری با دیگران

هم عالی باشید، تعامل با دیگران نیاز به مهارت‌های بیشتر دارد.

توجه: یک راه برای جلوگیری از مشکل اجرای هم‌روندی تراکنش‌ها، **اجرای سریالی تراکنش‌ها** است، یعنی اجرای یکی پس از دیگری و به ترتیب. انجام سریالی تراکنش‌ها از نظر نتیجه، کار درستی است، اما از نظر بهره‌وری، کار نادرستی است. چون، اجرای هم‌روند تراکنش‌ها مزایای عملکردی قابل توجهی دارد که در صورت اجرای سریالی، سیستم از مزایای هم‌روندی محروم می‌شود:

بهبود توان عملیاتی و بهره‌وری (Improved throughput and resource utilization)

یک تراکنش شامل مراحل زیادی است. برخی شامل فعالیت ورودی وخروجی هستند؛ برخی دیگر شامل فعالیت CPU هستند. CPU و دیسک‌ها در یک سیستم کامپیوتری می‌توانند به‌طور موازی کار کنند. بنابراین، فعالیت ورودی و خروجی می‌تواند به‌طور موازی با پردازش در CPU انجام شود. همه اینها توان عملیاتی (تعداد تراکنش‌های اجرا شده در واحد زمان) را افزایش می‌دهند. متقابلاً، استفاده از CPU و دیسک نیز افزایش می‌یابد؛ به عبارت دیگر، CPU و دیسک زمان کمتری را بیکار می‌مانند.

کاهش زمان انتظار (Reduced waiting time)

ممکن است ترکیبی از تراکنش‌ها در یک سیستم در حال اجرا باشد، برخی کوتاه و برخی طولانی. اگر تراکنش‌ها به‌طور سریال اجرا شوند، یک تراکنش کوتاه ممکن است مجبور باشد تا تکمیل شدن یک تراکنش طولانی قبلی منتظر بماند (convoy effect یا اثر کاروان)، که می‌تواند منجر به تأخیر در اجرای یک تراکنش شود. اگر تراکنش‌ها در بخش‌های مختلف پایگاه داده کار می‌کنند، بهتر است اجازه داد به‌طور هم‌روند اجرا شوند و چرخه‌های CPU و دسترسی‌های دیسک را بین آن‌ها به اشتراک گذاشت. اجرای هم‌روند، انتظار در اجرای تراکنش‌ها را کاهش می‌دهد. علاوه بر این، متوسط زمان پاسخ را کاهش می‌دهد. در یک قاعده کلی هرچه قدر به اجرای تراکنش‌های با زمان اجرای کوتاه‌تر اولویت اجرا دهیم، میانگین زمان پاسخ تراکنش‌ها به کمترین مقدار ممکن می‌رسد، پیش‌بینی زمان اجرای فرآیندها امکان پذیر نیست، بنابراین در عمل با الگوریتم اشتراک زمانی نوبت چرخشی Round Robin خود به خود تراکنش‌های با زمان اجرای کمتر زودتر پردازنده می‌گیرند و از سیستم خارج می‌شود و این باعث بهبود توان عملیاتی (Improved throughput) می‌شود. بنابراین بهتر است راه‌حل‌هایی توسعه یابد که به اجرای هم‌روند چندین تراکنش در کنار هم و بدون اختلال اجازه می‌دهد.

کنترل هم‌روندی (control concurrency)

کنترل هم‌روندی جهت تضمین خاصیت Isolation **ادعا** دارد که هم اجازه می‌دهد تراکنش‌ها به صورت هم‌روند اجرا شوند جهت بهره‌برداری از مزیت‌های هم‌روندی و هم اینکه طوری کنترل می‌کند که هیچ اختلالی به وجود نیاید. در واقع قاطعانه اجازه نمی‌دهد تراکنش‌های همکار از **نتایج میانی** هم استفاده کنند.

توجه: در مباحث پایگاه داده، **کنترل هم‌روندی تراکنش‌های هم‌کار** به دو روش **زمان‌بندی ترتیبی** و **زمان‌بندی موازی** انجام می‌شود.

زمان‌بندی ترتیبی

مجدداً در نظر بگیرید T1 و T2 دو تراکنش هستند که وجه را از یک حساب به حساب دیگر منتقل می کنند. فرض کنید درست قبل از اجرای تراکنش T_i ، مقادیر حساب‌های A و B به ترتیب 1000 و 2000 واحد پولی است. تراکنش T1 ، 50 واحد پولی از حساب A به حساب B منتقل می‌کند:

T_1: read(A);
$A := A - 50$;
write(A);
read(B);
$B := B + 50$;
write(B).

همچنین تراکنش T2 مقدار 10 درصد از حساب A را به B منتقل می‌کند.

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

فرض کنید دو تراکنش به طور هم‌روند و به ترتیب T1 و سپس T2 اجرا شوند. این ترتیب اجرایی در شکل زیر نشان داده شده 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

Schedule 1—a serial schedule in which T_1 is followed by T_2.

مقادیر نهایی حساب‌های A و B، پس از اجرای تراکنش‌ها مطابق شکل فوق، به ترتیب 855 و 2145 هستند. بنابراین، کل مبلغ پول در حساب‌های A و B یعنی مجموع A + B برابر 3000 از اجرای هر دو تراکنش حفظ می‌شود. جمع دو حساب در پایگاه داده سازگار باید قبل و بعد از اجرای تراکنش، یکسان و برابر 3000 باشد که هست. و به معنی این است که نه مبلغی گم شده و نه مبلغی پیدا شده.

زمان‌بندی موازی

با بازگشت به مثال قبلی، فرض کنید دو تراکنش به طور هم‌روند اجرا شوند. یک زمان‌بندی احتمالی می‌تواند مطابق شکل زیر باشد:

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

Schedule 3—a concurrent schedule equivalent to schedule 1.

پس از انجام این اجرا، به همان نتایجی می‌رسیم که در آن تراکنش‌ها به صورت سری و به ترتیب T1 و سپس T2 اجرا می‌شوند. مجموع A + B برابر 3000 در واقع حفظ می‌شود.

تمام اجرای‌های هم‌روند منجر به یک حالت صحیح نمی‌شوند. اما وظیفه DBMS اطمینان از این است که هر زمان‌بندی که اجرا می‌شود، پایگاه داده را در یک حالت سازگار قرار دهد. مؤلفه **کنترل هم‌روندی** DBMS توسط مکانیزم‌هایی که دارد این وظیفه را انجام می‌دهد.

توجه: خاصیت Isolation در تراکنش‌ها تضمین می‌کند که اجرای هم‌روند چند تراکنش، نتیجه‌ای معادل با حالتی داشته باشد که این تراکنش‌ها به‌صورت پشت‌سرهم و ترتیبی اجرا شده باشند. این کار توسط بخشی از DBMS به نام واحد **کنترل هم‌روندی** (Concurrency Control) انجام می‌شود.

تست – کنترل هم‌روندی (**Concurrency Control**) به عنوان واحدی در **DBMS** عهده‌دار رسیدگی به کدام خاصیت است؟ (مهندسی IT- آزاد ۹۳)

۱) Atomicity ۲) Isolation ۳) Consistency ۴) Durability

پاسخ – گزینه (۲) صحیح است.

۴- پایائی یا ماندگاری (Durabiltiy)

اگر یک تراکنش شروع به **پیشروی** کرد و به طور کامل تمام شد، باید در نهایت توسط دستور **commit** ذخیره و **Save** شود تا اطلاعات آن **از حافظه اصلی به دیسک** منتقل شود تا اثر آن **ماندگار** و **دائمی** شود.

توجه: داده‌هایی که روی دیسک نوشته می‌شود **هرگز** از بین نمی‌رود. چون حفاظت در برابر از دست دادن داده‌ها روی دیسک توسط **مکانیزم‌های پشتیبان‌گیری** و **ذخیره‌سازی پایدار** امکان‌پذیر است.

توجه: برای پیاده‌سازی **ذخیره‌سازی پایدار**، اطلاعات در چندین رسانه ذخیره‌سازی **غیرفرار** (معمولاً دیسک) به صورت **RAID** تکثیر می‌شود. همچنین، برخی سیستم‌های پشتیبانی، پشتیبان باتری را نیز ارائه می‌دهند، بنابراین **حافظه اصلی** نیز می‌تواند از قطعی برق جان سالم به در ببرد.

توجه: پس از **commit شدن** یک تراکنش، نمی‌توانیم اثرات آن را با **Undo** کردن **لغو** کنیم. تنها راه لغو اثرات یک تراکنش commit شده، اجرای یک **تراکنش جبران‌کننده** است. برای مثال، اگر یک تراکنش 20 واحد پولی به یک حساب اضافه کرد، تراکنش جبران‌کننده 20 واحد پولی از حساب کم می‌کند.

توجه: دو خاصیت **Isolation** و **Durabiltiy** توسط واحدی از **DBMS** به نام واحد **مدیریت بازیابی** (Recovery management) کنترل می‌گردد.

دلایل ۶ گانه نقض خاصیت Consistency در سیستم‌های بانک اطلاعات

به طور کلی نقض خاصیت Consistency در سیستم‌های بانک اطلاعات به ۶ دلیل زیر است:

(۱) **خاصیت Atomicity نقض شده باشد:** در صورت اجرای ناقص تراکنش. برای مثال مبلغ 50 واحد پولی از حساب مبدا برداشت شده، اما به دلیل خرابی سیستم مبلغ 50 به حساب مقصد واریز نشده است.

(۲) **خاصیت Isolation در تراکنش‌های هم‌روند نقض شده باشد:** استفاده از نتایج میانی

(۳) **محدودیت‌های جامعیتی داخلی نقض شده باشد:** برای مثال درج مقدار تکراری در ستون کلید اصلی نقض **جامعیت درون رابطه‌ای**، درج مقدار NULL در ستون کلید اصلی نقض **جامعیت موجودیت**، درج مقدار عددی در ستون با نوع کاراکتر نقض **جامعیت دامنه‌ای** و درج مقداری که زیر مجموعه کلید کاندید جدول اصلی نیست در ستون کلید خارجی جدول فرعی نقض **جامعیت ارجاعی**.

(۴) **محدودیت‌های جامعیتی خارجی نقض شده باشد:** برای مثال درج مقدار منفی در حساب بانکی، یا درج نمره 22 خارج از بازه تعریف شده 0 تا 20 ، یا در سیستم حساب بانکی مقدار مربوط به برداشت و واریز وجه برابر نباشد برای مثال مبلغ 50 واحد پولی از حساب برداشت شده ولی مبلغ 70 واریز شده باشد. تاریخ برگشت پرواز قبل از تاریخ رفت باشد.

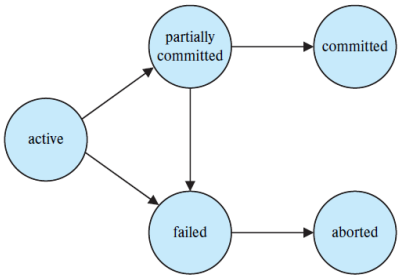
(۵) **خاصیت Durabiltiy در تثبیت تراکنش نقض شده باشد:** برای مثال قطعی برق یا خرابی دیسک.

(۶) **خطای انسانی در ورود داده:** برای مثال درج آدرس غلط برای مشتری که منجر به ورود داده ناصحیح به پایگاه داده شده است. DBMS امکان کنترل این نوع خطا را ندارد چون پدیده خطای انسانی است و نه فنی.

چرخه حیات تراکنش

همانطور که قبلاً اشاره کردیم، یک تراکنش ممکن است همیشه اجرای خود را با موفقیت تکمیل نکند. چنین تراکنشی **لغو شده** (aborted) نامیده می‌شود. اگر بخواهیم خاصیت Atomicity را تضمین کنیم، یک تراکنش لغو شده (aborted) نباید هیچ تأثیری بر حالت پایگاه داده داشته باشد. بنابراین، هر تغییری که تراکنش لغو شده در پایگاه داده ایجاد کرده است باید لغو شود. پس از لغو تغییرات ایجاد شده توسط یک تراکنش لغو شده، می‌گوییم تراکنش rolled back شده است که توسط سیستم مدیریت بازیابی (Recovery management) انجام می‌شود.

نمودار وضعیت یک تراکنش به صورت زیر است:



State diagram of a transaction.

۱- **فعال (Active) :** تراکنش در حال اجرا است و هنوز تکمیل نشده است. تراکنش در ابتدا در حالت فعال قرار دارد و تا زمانی که در حال اجرا است، در این حالت باقی می‌ماند.

۲- **نیمه تمام شده (Partially committed) :** پس از اجرای آخرین دستور، تراکنش در این حالت قرار می‌گیرد.

نیمه تمام شده: به این معنی است که تراکنش کار خود را تمام کرده و آخرین دستور آن اجرا شده است، اما هنوز commit نشده است. یعنی تغییرات ایجاد شده توسط تراکنش هنوز به طور کامل در **دیسک** ذخیره نشده است.

پس از اجرای آخرین دستور: یعنی زمانی که تراکنش تمام دستورات خود را اجرا کرده و به پایان رسیده است.

به عبارت ساده‌تر: یک تراکنش نیمه تمام شده، تراکنشی است که کار خود را انجام داده و نتایج نهایی داخل **حافظه اصلی** است، اما هنوز به طور کامل تضمین نشده است که تغییرات آن به طور **دائمی** در **دیسک** ذخیره شده باشد. مطابق نمودار این حالت واسطه‌ای بین حالت فعال **(Active)** و حالت تمام شده **(Committed)** است. یا واسطه‌ای بین حالت فعال **(Active)** و حالت ناکام **(Faild)** است.

۳- **ناکام (Failed) :** تراکنش به دلیل خطا یا شکست نتوانسته است به طور عادی اجرا شود.
این خطاها می‌توانند از انواع مختلفی باشد، مانند:

- خطاهای منطقی در خود تراکنش (مثلاً تقسیم بر صفر)
- خطاهای سخت‌افزاری در سیستم.
- خطاهای شبکه‌ای که ارتباط بین تراکنش و پایگاه داده را مختل می‌کند.

۴- **لغو شده (Aborted) :** تراکنش به دلیل خطا یا شکست لغو شده است و تغییرات آن نیز لغو شده‌اند. این حالت زمانی رخ می‌دهد که یک تراکنش ناکام (Failed) شده و سیستم تصمیم گرفته است آن را لغو کند. لغو کردن تراکنش به معنای بازگرداندن پایگاه داده به حالت قبلی خود قبل از شروع تراکنش است. **یک تراکنش لغو شده نمی‌تواند به حالت دیگری منتقل شود.**

۵- **انجام شده (Committed) :** تراکنش با موفقیت اجرا شده است و تغییرات آن در دیسک **تثبیت** و **پایدار** شده‌است. این حالت زمانی رخ می‌دهد که یک تراکنش با موفقیت اجرا شده و تمام تغییراتی که ایجاد کرده است در پایگاه داده ذخیره شده‌اند. این به معنای آن است که تراکنش به طور کامل تکمیل شده و پایگاه داده به حالت سازگار جدیدی رسیده است. پس از انجام کامل یک تراکنش، تغییرات آن به طور دائمی در دیسک ذخیره و Save می‌شود و نمی‌توان آن‌ها را لغو کرد. **یک تراکنش commit شده نمی‌تواند به حالت دیگری منتقل شود.**

شرایط لازم برای Commit شدن یک تراکنش:

- تراکنش باید تمام دستورات خود را بدون خطا اجرا کرده باشد.
- تراکنش باید تمام تغییراتی که ایجاد کرده است در دیسک ذخیره کرده باشد.

توجه: زمانی که یک سری تغییرات بر روی پایگاه داده انجام می شود، این تغییرات به صورت موقت در **حافظه اصلی** ذخیره می‌شود. اجرای دستور commit باعث می‌شود که این تغییرات موقت به صورت دائمی در **دیسک** ذخیره و Save شوند و تراکنش به پایان برسد.

تست – در یک نظام بانک اطلاعات کارآمد کدام گزینه ممکن است رخ دهد؟ (در حالت عادی سیستم – بدون رخ دادن هیچ اشکالی)

(مهندسی IT– آزاد ۸۷)

۱) برخی از دستورات تراکنش اجرا گردند و برخی دیگر خیر.
۲) مدتی پس از اتمام تراکنش و commit شدن آن ، DBMS آن را Undo می‌کند.
۳) در اجرای همزمان تراکنش‌ها، تراکنشی باعث غلط شدن نتیجه تراکنش دیگر می‌شود.
۴) در میانه اجرای تراکنش برخی قوانین جامعیتی به طور موقت نقض گردند.

پاسخ – گزینه (۴) صحیح است.

گزینه اول، خاصیت Atomicity مانع می‌شود. گزینه دوم، خاصیت Durability مانع می‌شود. گزینه سوم، Iolation مانع می‌شود. گزاره چهارم درست است. زیرا، DBMS مسئولیت حفظ وضعیت معتبر (Valid State) پایگاه داده‌ها را در «اکثر» مواقع برعهده دارد و نه به طور کامل و در هرلحظه، چون برای مثال در DBMS لحظاتی هست که مبلغی از حسابی کسر شده ولی هنوز به حساب مقصد واریز نشده است و در این میانه، سیستم تا لحظاتی نامعتبر است تا سرانجام در پایان تراکنش به وضعیت معتبر برسد.

تست – کدام یک از گزینه‌های زیر نادرست است؟ **(مهندسی IT – آزاد ۸۸)**

۱) یک تراکنش نمی‌تواند قوانین جامعیتی را نقض کند.
۲) در یک پایگاه داده، تراکنش‌ها هم‌زمان می‌توانند به داده‌های مورد نظر خود آزادانه دسترسی داشته باشند و کنترل خاصی در این زمینه لازم نیست.
۳) یا همه دستورات یک تراکنش با موفقیت اجرا می‌شود یا هیچ یک اجرا نمی‌شود.
۴) نتیجه یک تراکنش پس از تکمیل و اتمام آن نباید در اثر حوادث غیرمترقبه از بین برود.

پاسخ – گزینه (۲) صحیح است.

گزینه اول، خاصیت Consistency است. گزینه دوم، ناقض خاصیت Isolation در اجرای تراکنش‌های هم‌روند است. گزینه سوم، خاصیت Atomicity است. گزینه چهارم، خاصیت Durability است.

تست – کدام یک جزء وظایف DBMS نمی‌باشد؟ (مهندسی کامپیوتر- آزاد ۸۹)

۱) حفظ جامعیت و امنیت داده‌ها
۲) به روز رسانی کاتالوگ سیستم
۳) نظارت بر کارایی و پاسخ به تغییر نیازها
۴) ترمیم و حفظ سازگاری داده‌ها

پاسخ – گزینه (۳) صحیح است.

نظارت بر کارایی و پاسخ به تغییر نیازها از وظایف DA و DBA بانک اطلاعات است نه DMBS. بنابراین گزینه سوم جزء وظایف DBMS نیست. گزینه‌های دیگر از وظایف DBMS هستند.

تست – تراکنش نهایی نشده Uncommit تراکنشی است که: (مهندسی کامپیوتر- آزاد ۹۱)

۱) ساقط (abort) شده است.
۲) یا فعال (active) باشد یا ساقط شده باشد.
۳) فعال نیست.
۴) دستور Start آن صادر نشده است.

پاسخ – گزینه (۲) صحیح است.

تراکنش اگر در هر یک از وضعیت‌های فعال (Active) ، نیمه تمام شده (Partially committed) ، ناکام (Failed) و لغو شده یا ساقط شده (Aborted) منهای Commited باشد، تراکنش Uncommit محسوب می‌شود.

تست – در یک سیستم پایگاه داده‌ای، تراکنش‌ها باید چهار ویژگی اصلی معروف به (Atomicity, Consistency, Isolation, Durability) ACID را رعایت کنند. کدام یک از گزاره‌های زیر، به درستی مفهوم Durability و Consistency را شرح می‌دهد؟

(مهندسی کامپیوتر – دولتی ۱۴۰۴)

I – هر تراکنش باید وضعیت سیستم را از یک وضعیت معتبر به وضعیت معتبر دیگر تغییر دهد، حتی اگر چندین تراکنش به طور همزمان اجرا شوند.

II – هر تراکنش بلیدبه گونه‌ای اجرا شود که تراکنش‌های دیگر نتوانند وضعیت میانی آن را مشاهده کنند و تراکنش‌ها به صورت سریالی عمل کنند.

III – هر تراکنش باید به صورت کامل اجرا شود و در صورت رخ دادن خطا، تمامی تغییرات آن برگشت داده شود.

IV– هر تغییر انجام‌شده توسط یک تراکنش باید پس از اتمام موفقیت‌آمیز تراکنش، به طور دائمی در سیستم ذخیره شود.

۱) I و IV
۲) I و III
۳) II و III
۴) III و IV

پاسخ – **گزینه (۱) صحیح است.**

گزاره اول به خاصیت Consitency اشاره دارد و البته خاصیت مکمل آن یعنی Isolation جهت مدیریت تراکنش‌های هم‌روند. گزاره دوم به خاصیت Isolation اشاره دارد. گزاره سوم به خاصیت Atomicity اشاره دارد. گزاره چهارم به خاصیت Durability اشاره دارد.

زبان کنترل تراکنش‌ها (TCL: Transaction Control Language)

زبان کنترل تراکنش‌ها Transaction control language یا به اختصار TCL زیر مجموعه زبان SQL است که تراکنش‌ها را در پایگاه داده مدیریت می‌کند. دستورات این زبان در واقع تغییرات اعمال شده توسط دستورات DML را اداره می‌کند.

توجه: SQL هیچ یک از دستورات DML را **ماندگار** نمی‌کند، مگر اینکه به طور کامل اجرا و **Commit** شود. اگر تراکنش به طور کامل انجام نشود، برای مثال حین آن برق برود یا سیستم دچار مشکل و از کار افتادگی شود، تمامی تغییرات انجام شده به حالت اولیه بازگشته و لغو می‌شود.

دستورات پرکاربرد TCL در SQL به صورت زیر است:

دستور COMMIT : تغییرات اعمال شده در طی تراکنش را بر روی دیسک دائمی می‌کند.

دستور Commit اغلب به طور خودکار توسط DBMS فراخوانی می‌شود، گاهی هم توسط برنامه‌نویسی داخل برنامه کاربردی فراخوانی می‌شود.

در SQL Server ، هر دستور SQL که به تنهایی اجرا می‌شود، به عنوان یک **تراکنش ضمنی** در نظر گرفته می‌شود. به محض اینکه این دستور با موفقیت اجرا شد، SQL Server به طور خودکار آن را **commit** می‌کند.

مثال: درج یک رکورد

```
BEGIN TRANSACTION;

INSERT INTO S (S#, Sname, City)  //DML
VALUES (S4, Sn4', C3');

END TRANSACTION;  //TCL


COMMIT; // Save to hard disk
```

زمانی که چندین دستور SQL را می‌خواهیم به عنوان یک **واحد کاری منطقی** در نظر بگیریم، باید یک **تراکنش صریح** ایجاد کنیم و با دستور COMMIT آن را پایان دهیم. اگر در حین اجرای این دستورات مشکلی پیش بیاید، دستور ROLLBACK تمام تغییرات را لغو می‌کند.

مثال: انتقال وجه بین دو حساب

```
BEGIN TRANSACTION;

① UPDATE accounts SET balance = balance - 50 WHERE user_id = 'A';
② UPDATE accounts SET balance = balance + 50 WHERE user_id = 'B';

END TRANSACTION;

COMMIT; // Save to hard disk
```

دستور **ROLLBACK** : تمامی تغییرات یک تراکنش را به حالت اول برمی‌گرداند و لغو می‌کند. دستور Rollback اغلب به طور خودکار توسط DBMS فراخوانی می‌شود، مانند زمانی که **محدودیت‌های جامعیتی داخلی و خارجی** نقض شود مثل درج مقدار تکراری در ستون کلید اصلی یا خطای سیستمی رخ دهد یا در نیمه تراکنش برق قطع شود. گاهی هم جهت کنترل خطای بهتر توسط برنامه کاربردی فراخوانی می‌شود.

مثال: در **SQL Server**

```
BEGIN TRANSACTION;

عملیات مختلفی که باید به صورت یک تراکنش انجام شود

IF ERROR_NUMBER() <> 0
BEGIN
   ROLLBACK TRANSACTION;
   PRINT  'تراکنش لغو شد.' ;
END
ELSE
BEGIN
   COMMIT TRANSACTION;
   PRINT  'تراکنش با موفقیت انجام شد.' ;
END
```

توجه: دریک عبارت ساده هرآنچه که Commit نشود در نهایت باید Rollback شود.

معماری سیستم‌های بانک اطلاعات

منظور از معماری بانک اطلاعات، پیکربندی اجزای سیستمی است که در آن حداقل یک بانک اطلاعات، یک سیستم مدیریت بانک اطلاعات (DBMS)، یک سیستم عامل، یک کامپیوتر با دستگاه‌های جانبی و تعدادی کاربر (کاربران نهایی، مدیران و برنامه‌سازان) وجود دارد و خدماتی به کاربران نهایی ارائه می‌کند. نوع معماری به عواملی همچون سخت‌افزار، نرم‌افزار مدیریت بانک اطلاعات، موقعیت جغرافیایی کاربران، نیازهای کاربران ماهیت تراکنش‌ها، حجم داده‌ها و موقعیت مکانی داده‌ها و ارتباطات بین آنها بستگی دارد.

به طور کلی دو نوع معماری برای سیستم‌های بانکی وجود دارد :

۱- معماری متمرکز

۲- معماری نامتمرکز

- معماری مشتری ـ سرویس‌دهنده (خدمتگزار)
- معماری توزیع‌شده

معماری متمرکز

در این معماری یک پایگاه داده روی یک سیستم کامپیوتری و بدون ارتباط با سیستم کامپیوتری دیگر ایجاد می‌شود. و برای کاربردهای کوچک و با امکانات محدود است. سخت‌افزار این سیستم می‌تواند در حد یک کامپیوتر شخصی و یا بالاتر باشد. (مانند برنامه دفترچه تلفن شخصی)

معماری نامتمرکز

معماری مشتری - سرویس‌دهنده

هر معماری که در آن قسمتی از پردازش را یک برنامه، سیستم یا ماشین انجام دهد و انجام قسمت دیگری از پردازش را از برنامه، سیستم یا ماشین دیگری بخواهد، معماری مشتری - سرویس‌دهنده نامیده می‌شود.

در واقع وظایفی که باید سیستم انجام دهد به دو رده تقسیم می‌شوند: رده‌ای که انجام آنها بر عهده سرویس دهنده است و رده‌ای که توسط مشتری انجام می‌شود. بدین ترتیب یک معماری دو سطحی داریم که مشکل توزیع را تسهیل می‌کند. با این تعریف، در این معماری یک ماشین (یا سیستم یا برنامه) سرویسی را به ماشین (یا سیستم یا برنامه) دیگر ارائه می‌کند. بنابراین به این ماشین (یا سیستم یا برنامه) سرویس دهنده می‌گویند. خدماتی که سرویس‌دهنده ارائه می‌کند متنوع است، برای مثال خدمات چاپ، خدمات فایل، خدمات پست الکترونیک، خدمات اینترنت، خدمات بانک اطلاعات و بسته به نوع خدمت، سرویس‌دهنده را با نامی مناسب می‌نامند. برای نمونه سرویس‌دهنده چاپ و سرویس‌دهنده فایل، بنابراین منظور از معماری مشتری - سرویس‌دهنده آن نوع از معماری است که در آن مسئولیت‌ها به طور منطقی تقسیم شده است.

انواع معماری مشتری ـ سرویس‌دهنده از نظر تعداد مشتری و سرویس‌دهنده:

۱ـ چند مشتری ـ یک سرویس‌دهنده

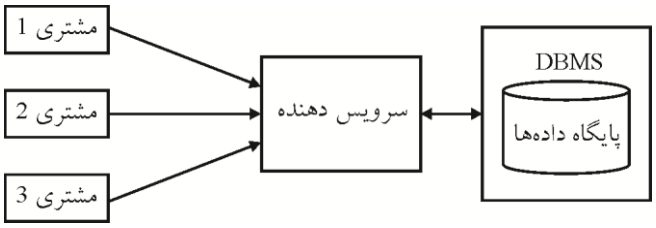
۲ـ یک مشتری ـ چند سرویس‌دهنده

۳ـ چند مشتری ـچند سرویس‌دهنده

انواع معماری مشتری ـ سرویس‌دهنده از نظر پیکربندی سخت‌افزاری

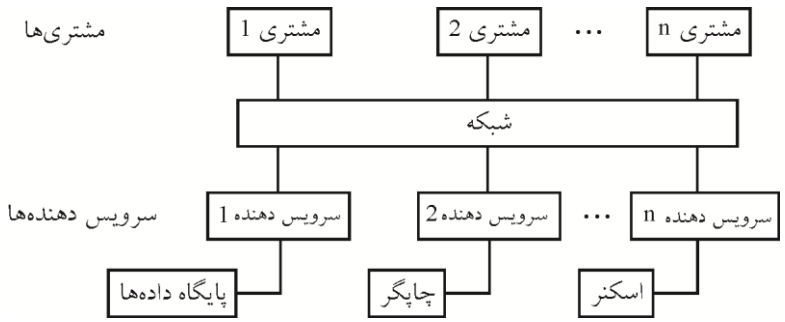
معماری حول سرویس‌دهنده

در این طرح، ماشین سرویس‌دهنده یک ابر کامپیوتر است و پایگاه داده‌ها روی همین کامپیوتر ایجاد و مدیریت می‌شود و تعدادی مشتری، از طریق شبکه از خدمات آن استفاده می‌کنند.



معماری حول شبکه

در این طرح، تعدادی کامپیوتر شخصی به عنوان سرویس‌دهنده و تعدادی دیگر به عنوان مشتری از طریق شبکه بهم مرتبط هستند.



معماری توزیع شده

اصطلاح پردازش توزیعی بدین مفهوم است که ماشین‌های مجزا می‌توانند در یک شبکه ارتباطی به یکدیگر متصل شوند. به گونه‌ای که یک عملکرد پردازش داده منفرد می‌تولند بر روی چندین ماشین شبکه پراکنده شود. در سیستم پایگاه‌های توزیع شده، پایگاه داده‌ها بر روی چند کامپیوتر ذخیره می‌شود. کامپیوترها در یک سیستم توزیع شده از طریق رسانه‌های مختلف ارتباطی نظیر شبکه‌های با سرعت بالا یا خطوط تلفن به یکدیگر مرتبط هستند.

می‌توان گفت که در این معماری تعدادی پایگاه داده‌های ذخیره شـــده روی کامپیوترهای مختلف داریم که از نظر کاربران، پایگاه داده واحدی هستند.

تست – کدام مورد از ویژگی های معماری توزیع شده در پایگاه داده‌ها نیست؟

(مهندسی کامپیوتر- آزاد ۸۹)

۱) داده‌ها به بخش‌هایی تقسیم و در سایت‌ها توزیع شده‌اند.

۲) سیستم چنان عمل می‌کند که از نظر کاربران شبیه به یک پایگاه داده متمرکز به نظر می‌رسد.

۳) داده‌های ذخیره شده در هر سایت تحت کنترل یک سیستم مدیریت پایگاه داده‌ها هستند.

۴) سیستم بانک اطلاعاتی به صورت یک ساختار دو بخشی در نظر گرفته می‌شود.

پاسخ – گزینه (۴) صحیح است.

هر سیستم پایگاه داده توزیع شده شامل مجموعه‌ای از سایت‌ها است که هر یک، یک سیستم پایگاه داده‌ی محلی را در خود نگه می‌دارند. (پس گزینه‌های اول و سوم درست‌اند.)

سیستم پایگاه داده‌ی توزیع شده باید از دید کاربر مشابه یک پایگاه داده‌ی متمرکز عمل کند و اگر برای اجرای یک پرس و جو به داده‌های چند سایت نیاز است، عمل تجزیه پرس و جو و ارسال آن به سایت‌ها و سپس دریافت پاسخ‌ها و ترکیب آن‌ها باید از دید کاربر مخفی بماند. (پس گزینه دوم نیز درست است.)

سیستم پایگاه توزیع شده می‌تواند شامل هر تعداد سایت باشد، پس گزینه‌ی چهارم عبارت نادرستی است.
