پایگاه‌داده پیشرفته

فصل اول

تراکنش

علی اصغر پورحاجی‌کاظم

مهر ۱۳۹۳

تعریف تراکنش

- واحد منطقی کار در سیستم که دارای خواص ACID باشد
- تراکنش عبارتند از واحد منطقی پردازش پایگاه داده‌ها که شامل یک یا چند عمل دستیابی به داده‌ها به منظور درج، حذف، تغییر یا بازیابی است
- دنباله‌ای از عمل‌های بهم مرتبط که اجرای آنها، تغییری در پایگاه‌داده‌ها ایجاد می‌کند و دارای خواص ACID می‌باشد

خواص تراکنش

- هر تراکنش باید دارای خواص ACID باشد:
  - Atomicity (یکپارچگی یا تجزیه‌ناپذیری)
  - Consistency (سازگاری)
  - Isolation (انزوا)
  - Durability (پایایی یا مانایی)

ATOMICITY

- این ویژگی به این معناست که یک تراکنش با تمامی دستوراتش یا به طور کامل اجرا شده و تثبیت می‌گردد و یا هیچ کدام از دستوراتش اجرا نمی‌شود

- این ویژگی به قانون «**همه یا هیچ**» نیز معروف است

- در صورتی که اجرای تراکنش با شکست مواجه شود، اثر تمامی دستورات اجرا شده از آن باید خنثی گردد

CONSISTENCY

- وقتی یک تراکنش شروع به اجرا می‌کند، فرض بر این است که پایگاه داده در وضعیت سازگار است

- این ویژگی بیانگر این موضوع است که اجرای موفق یک تراکنش پایگاه داده را از یک وضعیت سازگار به یک وضعیت سازگار دیگر می‌برد

ISOLATION

- اجرای موازی و همروند تراکنش‌ها

موازی

T_1	T_2
R(D)	R(D)
D:=D – N	D: =D+M
W(D)	W(D)
R(E)	
E:=E+N	
W(E)	

همروند

T_i	T_2
R(D)	
D:=D – N	
	R(D)
	D:=D+M
W(D)	
R(E)	
	W(D)
E:=E+N	
W(E)	

ISOLATION (ادامه)

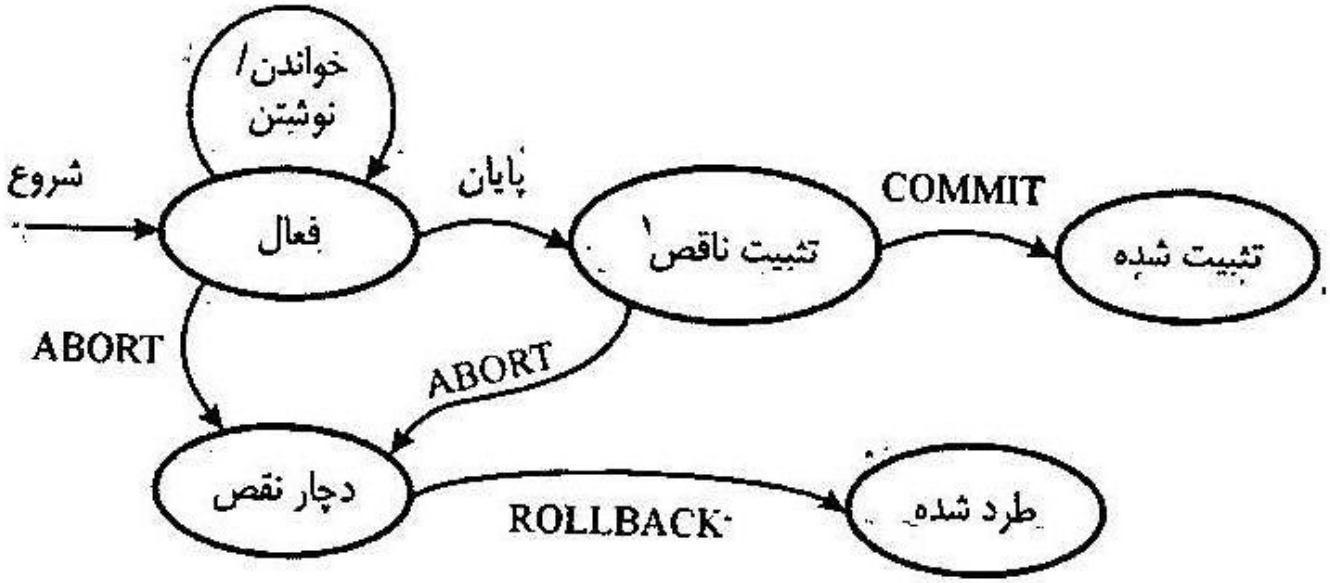
- اثر اجرای همروند چند تراکنش بر روی پایگاه‌داده باید همانند حالتی باشد که آن تراکنش‌ها در انزوا و به صورت تکی و تنها اجرا می‌شوند

- اجرای همروند تراکنش‌ها در صورتی که کنترل نشود، ممکن است پایگاه‌داده را در وضعیت ناسازگار قرار دهد

DURABILITY

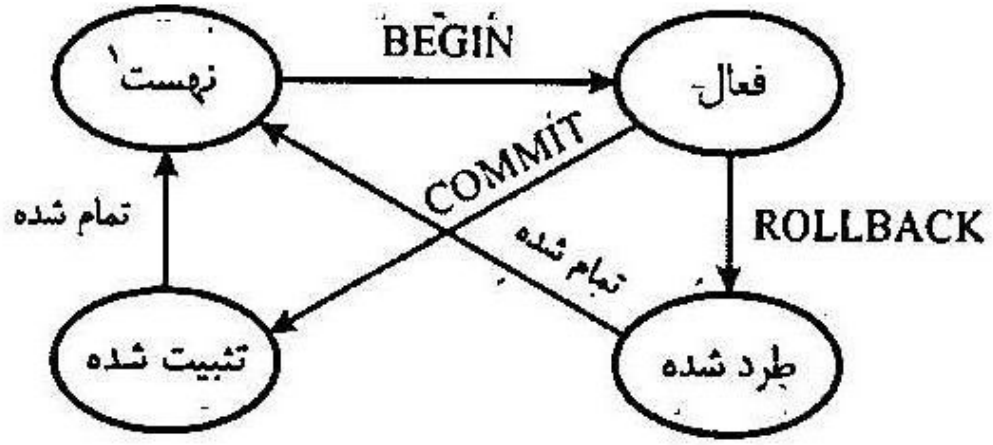
- نتیجه اجرای موفق و درست یک تراکنش پس از اجرای آن ماندگار است و نباید از بین برود

حالات تراکنش



شکل ۲-I : حالات تراکنش

حالات تراکنش (ادامه)



شکل I-۳ : نمایش دیگری از حالات تراکنش

بیان صوری تراکنش

- $O_{ij}(D)$: دستور O_j از تراکنش T_i روی داده D $O_{ij} \in \{R, W\}$
- هر یک از این دستورات تجزیه‌ناپذیر هستند
- OS_i : مجموعه تمام دستورات T_i $OS_i = \{O_{ij} \mid j \geq 0\}$
- N_i : نشان دهنده وضعیت پایان T_i $N_{ij} \in \{A, C\}$
- با نمادهای مذکور می‌توان تراکنش را به صورت یک ترتیب جزئی روی مجموعه دستورات و مجموعه وضعیت پایانی آن تعریف کرد

بیان صوری تراکنش (ادامه)

- تراکنش T_i یک ترتیب جزئی $T_i = \{\sum_i, r\}$ است که در آن:

$$\sum_i = OS_i \cup \{N_i\}$$

- اگر برای هر دو دستور O_{ij} و O_{ik} متعلق به OS_i، $O_{ij} \in \{R(D), W(D)\}$ و $O_{ik} = W(D)$ باشد، در این صورت:

$$O_{ij}\ r_i\ O_{ik} \quad \text{یا} \quad O_{ik}\ r_i\ O_{ij}$$

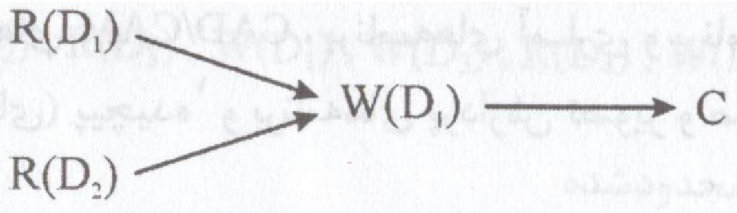
- $\forall O_{ij} \in OS_i\ ;\ O_{ij}\ r_i\ N_i$

بیان صوری تراکنش (ادامه)

```
BEGIN TRANSACTION
    R(D1)
    R(D2)
    D1 : = D1+D2
    W(D1)
COMMIT
```

$$\Sigma = \{R(D_1) , R(D_2) , W(D_1) , C\}$$

$$r = \{(R(D_1) , W(D_1)) , (R(D_2) , W(D_1)) , (W(D_1) , C) , (R(D_1) , C) , (R(D_2) , C)\}$$



ضابطه اول: مدت زمان اجرا

- تراکنش کوتاه
  - زمان اجرا و زمان پاسخدهی بسیار کوتاه دارد (در حدود ثانیه یا دقیقه) و معمولا به بخش کوچکی از پایگاه دادهها دسترسی دارد
- تراکنش طولانی
  - زمان اجرا و زمان پاسخدهی طولانی دارد
  - طول مدت بستگی به کاربرد دارد و ممکن است چند ثانیه، دقیقه و یا چند ساعت باشد

ضابطه دوم: ترتیب خواندن / نوشتن

- تراکنش عمومی
  - دستورهای خواندن و نوشتن از ترتیب خاصی پیروی نمی‌کنند
- تراکنش دو مرحله‌ای
  - تمامی دستورات خواندن پیش از دستورات نوشتن اجرا می‌شوند

ضابطه سوم: تعداد سطوح

- تراکنش مسطح
  - یک نقطه آغاز و یک نقطه پایان دارد و می‌تواند شامل هر تعداد دستور ساده باشد

```
BEGIN  TRANSACTION
    action 1
    action 2
        .
        .
        .
    action i
END  TRANSACTION
```

ضابطه سوم: تعداد سطوح (ادامه)

- تراکنش تو در تو
  - نوعی تراکنش که از تعدادی زیر تراکنش مرتبط به هم تشکیل شده است

ضابطه سوم: تعداد سطوح (ادامه)

```
BEGIN TRANSACTION T1
    .
    .
  BEGIN TRANSACTION T2
      .
      .
    BEGIN TRANSACTION T3
        .
        .
    END TRANSACTION T3
    BEGIN TRANSACTION T4
        .
        .
    END TRANSACTION T4
      .
      .
  END TRANSACTION T2
    .
    .
END TRANSACTION T1 ;
```

انواع تراکنش تو در تو

- تراکنش تو در تو بسته
  - اجرای زیر تراکنش پس از تراکنش بیرونی آغاز می‌شود و تثبیت شدن آن منوط به تثبیت شدن تراکنش بیرونی است

- تراکنش تو در تو باز
  - در این تراکنش نتیجه اجرای یک تراکنش درونی، حتی اگر تراکنش بیرونی‌اش تثبیت نشده باشد، می‌تواند در اختیار تراکنش‌های دیگر (نه تنها پدرانش) قرار بگیرد

تکنیک نقطه نگهداشت

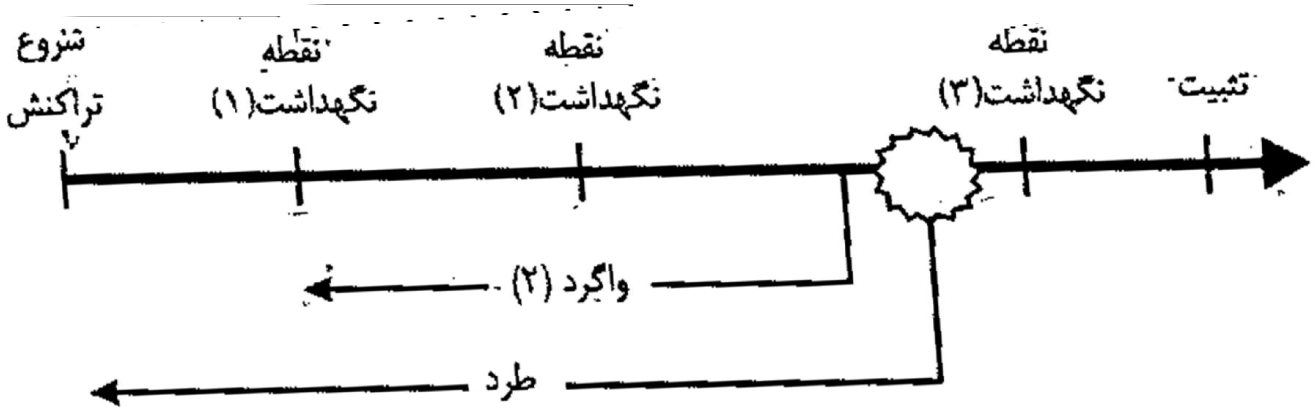
- در بعضی از کاربردها اصل **همه یا هیچ**، کارایی سیستم را کاهش می‌دهد (مخصوصاً زمانی که تراکنش طولانی باشد)

- برای حل این مشکل دو راه‌حل وجود دارد
  - ایجاد تراکنش تو در تو
  - ایجاد نقطه نگهداشت

تکنیک نقطه نگهداشت

- برای ایجاد نقطه نگهداشت، سیستم با اجرای تابعی مانند Save Work وضعیت جاری پردازش را ظبط می‌کند
- این تابع یک عدد به برنامه کاربردی برمی‌گرداند و برنامه می‌تواند با درخواست Rollback Work و دادن شماره دریافتی به نقطه نگهداشت مورد نظرش برگردد
- آنچه که بین دو نقطه نگهداشت اجرا می‌شود، اصطلاحاً کنش یا عمل تجزیه‌ناپذیر نامیده می‌شود

تکنیک نقطه نگهداشت



شکل ۱-۷ : تکنیک نقطه نگهداشت