ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ
И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

ВОВЕД ВО ОБРАБОТКА НА ТРАНСАКЦИИ

БАЗИ НА ПОДАТОЦИ - предавања

Проф. д-р Слободан Калајџиски

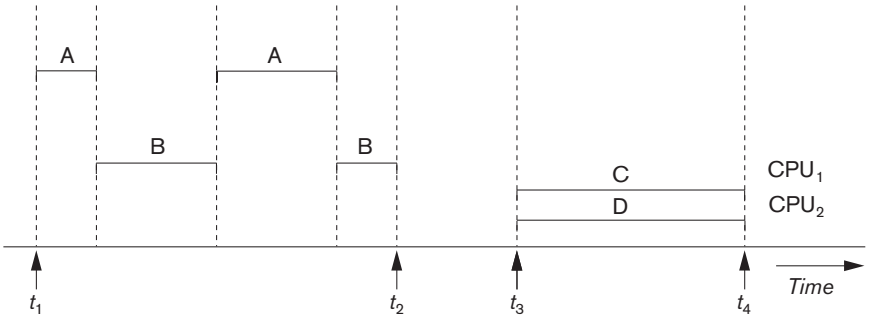
Универзитет “Св. Кирил и Методиј” – Факултет за информатички науки и компјутерско инженерство

Преглед

- Вовед во обработката на трансакции
- Поим за трансакции и посакувани својства на трансакциите
- Одредување на распореди врз основа на серијабилност
- Подршка за трансакции кај SQL

Вовед во обработка на трансакции

- **Систем со еден корисник**:
  - Најмногу еден корисник во даден момент може да го користи системот.
- **Повеќе-кориснички систем**:
  - Повеќе корисници може да пристапат до системот истовремено.



БАЗИ НА ПОДАТОЦИ – предавања

Вовед во обработка на трансакции

- **Трансакција** е:
  - Логичка целина на обработката на базите на податоци којашто вклучува една или повеќе **операции за пристап**
    - операции за пристап
      - read – вчитување
      - write – додавање
      - промена или бришење
- Една трансакција (множество на операции) може да биде:
  - самостојно специфицирана во еден виш јазик како SQL испратена интерактивно, или
  - можеби да биде вградена во некоја програма.
- **Граници на трансакцијата**:
  - **Begin transaction** (за почеток) и **End transaction** (за крај).

Упростен модел на база на податоци

- **БП** е збирка на именувани податочни ставки
- **Грануларност** на податоците – може да биде поле, запис или цел блок од дискот (поимите се независни од грануларноста)
- Основните операции се **read** и **write**
  - **Читај_ставка (X) (на анг. read_item(X))**
    Ја чита ставката од базата на податоци именувана како X во некоја програмска променлива. За да се поедностави нашата нотација, претпоставуваме дека *програмската променлива е исто така именувана како X.*
  - **Пишувај_ставка (X) (на анг. write_item(X))**
    Ја запишува вредноста на програмската променлива X во ставката од базата на податоци именувана како X.

БАЗИ НА ПОДАТОЦИ – предавања

Упростен модел на база на податоци

- Извршувањето на наредбата **читај_ставка (X)** ги вклучува следниве чекори:
  1. Најди ја адресата на блокот од дискот што ја содржи ставката X.
  2. Копирај го блокот од дискот во меѓумеморијата на главната меморија (ако тој блок на дискот не е веќе во некоја меѓумеморија на главна меморија).
  3. Копирај ја ставката X од меѓумеморијата во програмската променлива X.

ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО
© ФИНКИ 2020 – online предавање

Упростен модел на база на податоци

- Извршувањето на **пишувај_ставка(X)** командата ги вклучува следниве чекори:
  1. Најди ја адресата на блокот од дискот што ја содржи ставката X.
  2. Копирај го блокот од дискот во меѓуменоријата на главната меморија (ако тој блок на дискот не е веќе во некоја меѓумеморија на главната меморија).
  3. Копирај ја ставката X од програмската променлива X во нејзината правилна локација во меѓуменоријата.
  4. Зачувај го ажурираниот блок од меѓуменоријата назад на дискот (или веднаш или после некое време).

Чекорот 4 всушност ја ажурира базата на податоци на дискот

Пример

- Пример за две едноставни трансакции:
  - Нека X и Y се резервирани авионски седишта, а N и M се број на седишта што се резервираат или ослободуваат, соодветно

(a)

T_1

read_item(X);
$X := X - N$;
write_item(X);
read_item(Y);
$Y := Y + N$;
write_item(Y);

(b)

T_2

read_item(X);
$X := X + M$;
write_item(X);

Пример

- Пример за две едноставни трансакции:
  - Нека X и Y се резервирани авионски седишта, а N и M се број на седишта што се резервираат или ослободуваат, соодветно

(a)

T_1

read_item(X);
$X := X - N$;
write_item(X);
read_item(Y);
$Y := Y + N$;
write_item(Y);

(b)

T_2

read_item(X);
$X := X + M$;
write_item(X);

Што се случува кога T1 и T2 се извршуваат испреплетено?

Зошто е потребна контрола на конкурентност

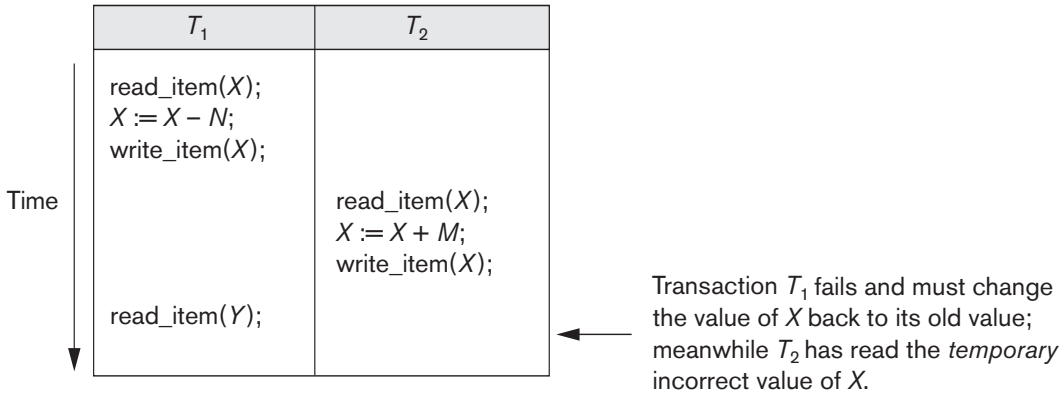
- **Проблемот на изгубено ажурирање (the lost update problem)**
  - Овој проблем се појавува кога две трансакции што пристапуваат до истите податочни ставки имаат испреплетени операции на таков начин што ја прави неточна вредноста на некои од податочните ставки во базата на податоци (се пребришува претходно запишана вредност со друга вредност)

Time ↓

T_1	T_2	
read_item(X); $X := X - N$;		
	read_item(X); $X := X + M$;	
write_item(X); read_item(Y);		
	write_item(X);	← Item X has an incorrect value because its update by T_1 is *lost* (overwritten).
$Y := Y + N$; write_item(Y);		

Зошто е потребна контрола на конкурентност

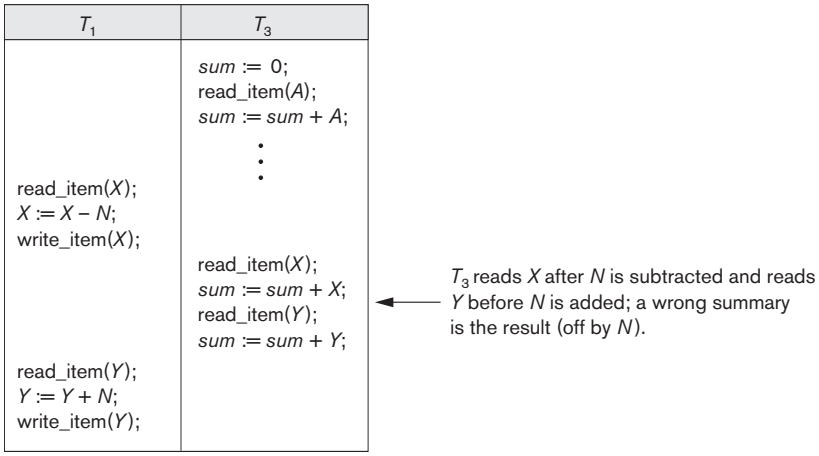
- **Проблемот на привремено ажурирање (или на валкано читање)**
  - Овој проблем се јавува кога една трансакција ажурира една податочна ставка, а потоа трансакцијата паѓа од некоја причина.
  - Ажурираната ставка е пристапена од некоја друга трансакција, пред да се промени ставката назад во својата првична вредност.



БАЗИ НА ПОДАТОЦИ – предавања

Зошто е потребна контрола на конкурентност

- **Проблемот на неточно сумирање**
  - Ако една трансакција прави агрегација, на пр. сумирање на повеќе записи, додека други трансакции ажурираат некои од овие записи, збирната функција може да пресмета некои вредности пред тие да бидат ажурирани, а некои други да ги додаде после ажурирањето.



ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

© ФИНКИ 2020 – online предавање

Зошто е потребна контрола на конкурентност

- **Проблемот на неповторливо читање**
  - Ако трансакцијата T чита еден запис двапати и записот е сменет од страна на друга трансакција T' помеѓу двете читања. Оттука, T добива различни вредности за двете читања на истиот запис.
  - Пример:

    Ако за време на трансакција за резервација на авионски билет, некој корисник се информира за достапноста на седиштата на неколку летови. Кога корисникот ќе се одлучи за одреден лет, тогаш трансакцијата го чита бројот на седишта на тој лет по вторпат пред да се комплетира резервацијата.

Типови на (прекини) падови на трансакции

1. Пад на компјутерот (системски пад)
2. Грешка во трансакцијата или на системот
3. Локални грешки или услови за исклучоци
4. Спроведување на контрола на конкурентноста
5. Пад на дискот
6. Физички проблеми и природни катастрофи

Сите овие ќе бидат разгледувани од страна на делот за обнова на СУБП кога ќе настане некој пад на системот!

СУБП треба да чува доволно информации за да може да се опорави од падовите (најчесто падовите од 1 до 4)

Трансакциски концепти

- **Трансакција (transaction)** претставува неделлива единица работа која или е извршена во својата целост или не се извршува воопшто.
- Системот за обнова треба да води информација за следните работи:
  - **begin_transaction**:
    Го означува *почетокот на извршувањето* на трансакцијата.
  - **read** или **write**:
    Спецификација на операциите за *читање* или *запишување* врз ставките во БП кои се извршуваат како дел од трансакцијата.
  - **end_transaction**:
    Операциите за читање или запишување завршиле, и го означува *крајот на извршувањето* на трансакцијата.

Трансакциски концепти

- Системот за обнова треба да води информација за следните работи (продолжение):
  - **commit_transaction**:
    Ова сигнализира *успешен крај* на трансакцијата, така што секакви промени (ажурирања) извршени од трансакцијата можат безбедно да бидат **потврдени** во базата на податоци.
  - **rollback** (или **abort**):
    Ова сигнализира дека трансакцијата *завршила неуспешно*, така што какви било промени или ефекти што трансакцијата можеби ги применила на базата на податоци мора да се *вратат назад.*

Трансакциски концепти

- **Состојби на трансакциите**:
  - Активна состојба
  - Делумно потврдена состојба
  - Потврдена состојба (committed state)
  - Неуспешна состојба (failed state)
  - Прекината состојба (terminated state)

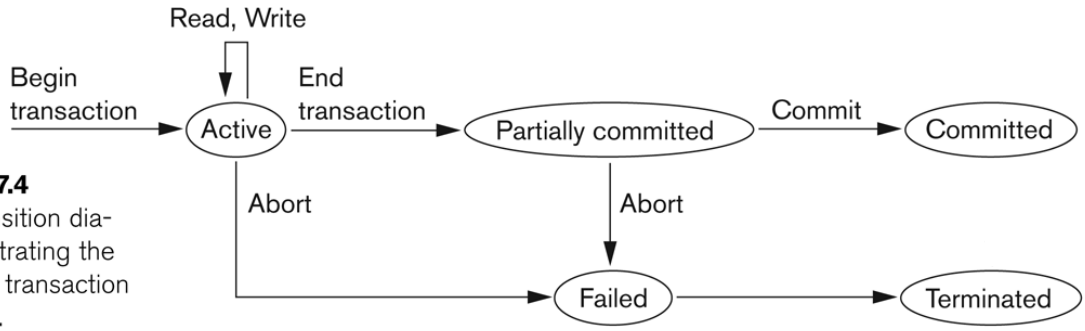


Figure 17.4
State transition diagram illustrating the states for transaction execution.

Системски дневник

- **Системскиот дневник** или **Log** води евиденција на сите трансакциски операции што влијаат на вредноста на ставките во БП.
  - Оваа информација се користи за обнова на системот после падови на трансакциите.
  - Дневникот се чува на диск, па истиот не е подложен на никаков друг вид на пад освен на падовите на дискот или катастрофалните падови.
  - По потреба, системскиот дневник се архивира на некој уред за архивирање (магнетни ленти).

Системски дневник

- Во системскиот дневник се чуваат следните информации:
  - [**start_transaction, T**]
    Зачувува дека трансакцијата T започнала со извршувањето.
  - [**write_item, T, X, old_value, new_value**]
    Покажува дека трансакцијата T ја променила вредноста на податочната ставка X од old_value во new_value.
  - [**read_item, T, X**]
    Забележува дека трансакцијата T ја прочитала вредноста на ставката X од базата на податоци.
  - [**commit, T**]
    Покажува дека трансакцијата T успешно завршила и потврдува дека нејзиното дејство може да биде потврдено (commit) во базата на податоци.
  - [**abort, T**]
    Означува дека трансакцијата T била откажана (abort).

Системски дневник

(a)

T_1	T_2	T_3
read_item(A)	read_item(B)	read_item(C)
read_item(D)	write_item(B)	write_item(B)
write_item(D)	read_item(D)	read_item(A)
	write_item(D)	write_item(A)

		A	B	C	D
		30	15	40	20
	[start_transaction,T_3]				
	[read_item,T_3,C]				
*	[write_item,T_3,B,15,12]		12		
	[start_transaction,T_2]				
	[read_item,T_2,B]				
**	[write_item,T_2,B,12,18]		18		
	[start_transaction,T_1]				
	[read_item,T_1,A]				
	[read_item,T_1,D]				
	[write_item,T_1,D,20,25]				25
	[read_item,T_2,D]				
**	[write_item,T_2,D,25,26]				26
	[read_item,T_3,A]				

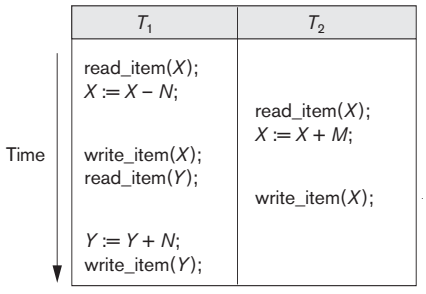
Пример за системски дневник за три трансакции Т1, Т2 и Т3.

Посакувани својства на трансакциите

- **Атомичност (Atomicity)**. Трансакцијата е атомична (неделива) процесирачка единица. Или се извршува целосно или воопшто не се извршува.
- **Зачувување на доследност (Consistency preservation).** Трансакцијата ја зачувува доследноста доколку нејзиното целосно извршување ја пренесува базата на податоци од една доследна состојба во друга.
- **Изолација (Isolation).** Треба да изгледа дека трансакцијата се извршува во изолација од другите трансакции. Тоа значи, извршувањето на една трансакција не треба да влијае на другите трансакции кои се извршуваат истовремено со неа.
- **Трајност или постојаност (Durability or permanency).** Промените што се извршуваат на базата на податоци од страна на потврдени трансакции мора да бидат трајни, и не смеат да бидат изгубени поради некој пад.

Распореди (истории) на трансакции

- **Распоред (schedule)** (или **историјат (history)**) S на n трансакции T_1, T_2, …, T_n:
  - Тоа е редоследот на операциите на трансакциите таков што, за секоја трансакција T_i што учествува во S, операциите на T_i во S мора да се појават по истиот редослед по кој што тие се појавуваат во T_i.
  - Сепак, операциите од другите трансакции T_j може да бидат испреплетени со операциите во T_i во S.



S_a: $r_1(X)$; $r_2(X)$; $w_1(X)$; $r_1(Y)$; $w_2(X)$; $w_1(Y)$

Распореди (истории) на трансакции

- Две операции се во **спор** (conflict) доколку:
  - припаѓаат на различни трансакции
  - тие пристапуваат на иста ставка *X* и
  - барем една од операциите е запиши_ставка (*X*).

$$S_a: r_1(X); r_2(X); w_1(X); r_1(Y); w_2(X); w_1(Y)$$

БАЗИ НА ПОДАТОЦИ – предавања

Распореди (истории) на трансакции

- Две операции се во **спор** (conflict) доколку:
  - припаѓаат на различни трансакции
  - тие пристапуваат на иста ставка *X* и
  - барем една од операциите е запиши_ставка (*X*).

- Пример:

$$S_a: r_1(X); r_2(X); w_1(X); r_1(Y); w_2(X); w_1(Y)$$

  - операциите $r_1(X)$ и $w_2(X)$ се спорни исто како и операциите $r_2(X)$ и $w_1(X)$ и операциите $w_1(X)$ и $w_2(X)$
  - операциите $r_1(X)$ и $r_2(X)$ не се спорни бидејќи и двете се операции на читање
  - операциите $w_2(X)$ и $w_1(Y)$ не се спорни, бидејќи двете се однесуваат на различни податочни ставки *X* и *Y*
  - операциите $r_1(X)$ и $w_1(X)$ не се спорни бидејќи припаѓаат на истата трансакција.

ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

© ФИНКИ 2020 – online предавање

Распореди според серијабилноста (serializability)

- **Сериски распоред**:
  - Некој распоред S е сериски ако, за секоја трансакција T која што учествува во распоредот, сите операции од T се извршуваат едно-по-друго во распоредот.

- **Пример:**
  Манипулација на слободни седишта во системот за авионски резервации

(a)

T_1	T_2

read_item(X);
$X := X - N$;
write_item(X);
read_item(Y);
$Y := Y + N$;
write_item(Y);
	read_item(X);
	$X := X + M$;
	write_item(X);

Time

Schedule A

(b)

T_1	T_2

read_item(X);
$X := X - N$;
write_item(X);
read_item(Y);
$Y := Y + N$;
write_item(Y);
	read_item(X);
	$X := X + M$;
	write_item(X);

Time

Schedule B

Што се случува во пракса ако сите распореди на извршување на трансакциите беа сериски?

БАЗИ НА ПОДАТОЦИ – предавања

Распореди според серијабилноста (serializability)

- **Сериски распоред**:
  - Некој распоред S е сериски ако, за секоја трансакција T која што учествува во распоредот, сите операции од T се извршуваат едно-по-друго во распоредот.

- **Пример:**
  Манипулација на слободни седишта во системот за авионски резервации

(a)

T_1	T_2

read_item(X);
$X := X - N$;
write_item(X);
read_item(Y);
$Y := Y + N$;
write_item(Y);
	read_item(X);
	$X := X + M$;
	write_item(X);

Time

Schedule A

(b)

T_1	T_2

read_item(X);
$X := X - N$;
write_item(X);
read_item(Y);
$Y := Y + N$;
write_item(Y);
	read_item(X);
	$X := X + M$;
	write_item(X);

Time

Schedule B

Што ако е дозволено испреплетено извршување на операциите во трансакциите?

Колку различни распореди може да се добијат?

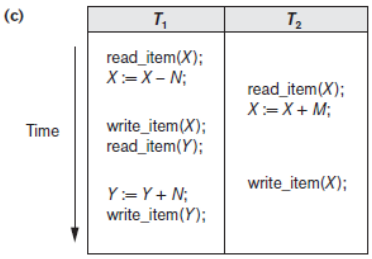
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

© ФИНКИ 2020 – online предавање

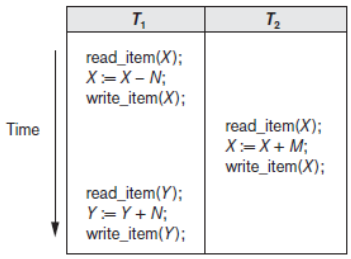
Распореди според серијабилноста

- **Подредлив (serializable) серијабилен распоред**:
  - Некој распоред S е серијабилен, ако е еквивалентен со некој сериски распоред на истите трансакции.

(c)



Schedule C



Schedule D

```
X = 50, Y = 30
M = 5,  N = 2
```

Ќе не интересираат оние несериски распореди кои се серијабилни!

БАЗИ НА ПОДАТОЦИ – предавања

Распореди според серијабилноста

- Серијабилен распоред **не** е исто што и сериски распоред
- Под поимот серијабилен распоред се подразбира дека распоредот е **правилен**.
  - Ќе ја остави базата во конзистентна состојба.
  - Преплетувањето е дозволено и ќе резултира во иста состојба како да трансакциите биле извршувани сериски, а сепак ќе постигне поголема ефикасност заради истовременото извршување на трансакциите.
- Серијабилноста тешко се проверува.
  - Преплетувањето на операциите се случува во оперативниот систем преку некој распоредувач
  - Тешко е однапред да се одреди како ќе бидат испреплетени операциите во распоредот.

ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

© ФИНКИ 2020 – online предавање

Распореди според серијабилноста

Практичен пристап:

- Да се измислат методи (протоколи) кои ќе обезбедуваат подредливост.
- Не е можно да се одреди кога еден распоред почнува, а кога завршува.
  - Оттука, го редуцираме проблемот на проверка на целиот распоред, на проверка само на **потврдениот дел (committed part)** од распоредот (т.е. само на операциите од потврдените трансакции)
- Современ пристап користен во повеќето СУБП:
  - Употреба на брави со дво-фазно заклучување

БАЗИ НА ПОДАТОЦИ – предавања

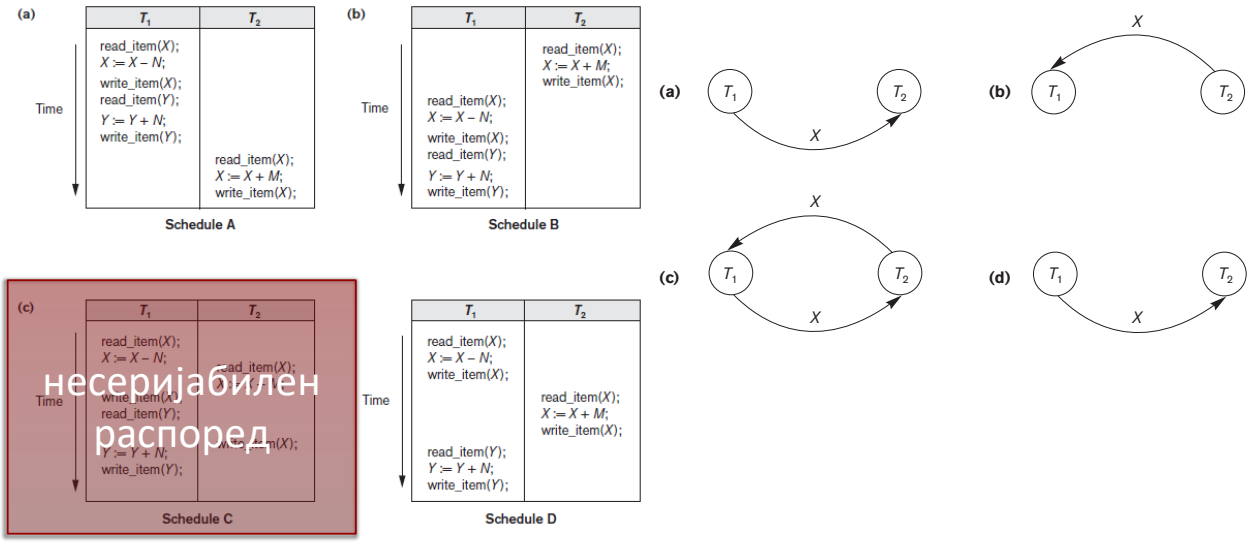
Распореди според серијабилноста

Тестирање за серијабилност според конфликт: Algorithm 17.1:

- Разгледувај ги само read_Item (X) и write_Item (X) операциите
- Конструирај граф на приоритети (граф на серијализација) – насочен граф каде јазлите ќе бидат трансакциите кои се разгледуваат
  - Се креира врска помеѓу јазлите T_i и T_j ако некоја од операциите во T_i се појавува пред конфликтната операција во T_j
- Распоредот е серијабилен ако и само ако конструираниот граф на приоритети не содржи циклуси.

ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО
© ФИНКИ 2020 – online предавање

Распореди според серијабилноста



Градење на графовите на приоритети (Precedence Graphs)

БАЗИ НА ПОДАТОЦИ – предавања

Распореди според серијабилноста

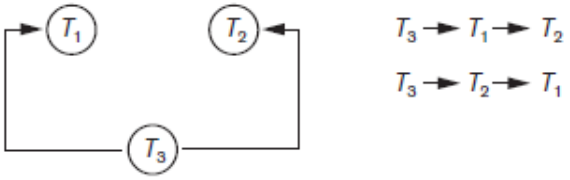
Figure 17.8
Another example of serializability testing. (a) The read and write operations of three transactions T_1, T_2, and T_3. (b) Schedule E. (c) Schedule *F*.

(a)

Transaction T_1
read_item(X);
write_item(X);
read_item(Y);
write_item(Y);

Transaction T_2
read_item(Z);
read_item(Y);
write_item(Y);
read_item(X);
write_item(X);

Transaction T_3
read_item(Y);
read_item(Z);
write_item(Y);
write_item(Z);



ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

© ФИНКИ 2020 – online предавање

Распореди според серијабилноста

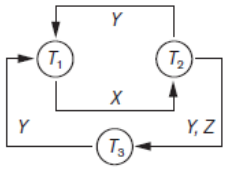
Figure 17.8
Another example of serializability testing. (a) The read and write operations of three transactions T_1, T_2, and T_3. (b) Schedule E. (c) Schedule F.

(b)

Time

Transaction T_1	Transaction T_2	Transaction T_3
	read_item(Z);	
	read_item(Y);	
	write_item(Y);	
		read_item(Y);
		read_item(Z);
read_item(X);		
write_item(X);		
		write_item(Y);
		write_item(Z);
	read_item(X);	
read_item(Y);		
write_item(Y);		
	write_item(X);	

Schedule E



Equivalent serial schedules

None

Reason

Cycle $X(T_1 \rightarrow T_2), Y(T_2 \rightarrow T_1)$
Cycle $X(T_1 \rightarrow T_2), YZ(T_2 \rightarrow T_3), Y(T_3 \rightarrow T_1)$

БАЗИ НА ПОДАТОЦИ – предавања

Распореди според серијабилноста

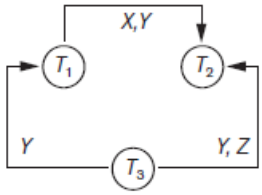
Figure 17.8
Another example of serializability testing. (a) The read and write operations of three transactions T_1, T_2, and T_3. (b) Schedule E. (c) Schedule F.

(c)

Time

Transaction T_1	Transaction T_2	Transaction T_3
		read_item(Y);
		read_item(Z);
read_item(X);		
write_item(X);		
		write_item(Y);
		write_item(Z);
	read_item(Z);	
read_item(Y);		
write_item(Y);		
	read_item(Y);	
	write_item(Y);	
	read_item(X);	
	write_item(X);	

Schedule F



Equivalent serial schedules

$T_3 \rightarrow T_1 \rightarrow T_2$

ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

© ФИНКИ 2020 – online предавање

Подршка за трансакции кај SQL

- **Една** SQL наредба секогаш се смета за неделлива (**atomic**).
  - Или наредбата се извршува без грешка, или не успева и ја остава базата непроменета.
- Кај SQL, не постои експлицитна Begin Transaction наредба.
  - Почетокот на трансакцијата е имплицитен со SQL наредбата.
- Секоја трансакција мора да има експлицитна наредба за крај, која е или
  - COMMIT (потврда) или
  - ROLLBACK (враќање назад).

Својства одредени со SET TRANSACTION

- **Режим на пристап (Access mode)**:
  - READ ONLY или READ WRITE.
    - Предодреден е READ WRITE освен кај READ UNCOMITTED нивото, (таму се подразбира READ ONLY).
- **Дијагностичка област (Diagnostic size)** n, одредува цел број n, кој што го покажува бројот на услови коишто можат да се чуваат симултано во делот за дијагностика.
  - овие услови обезбедуваат повратна информација за *n*-те последно извршени SQL наредби

Својства одредени со SET TRANSACTION

- **Isolation level** <isolation>, каде <isolation> може да биде една од:
  - **READ UNCOMMITTED**
  - **READ COMMITTED**
  - **REPEATABLE READ** или
  - **SERIALIZABLE.**
  - Предодредената вредност е SERIALIZABLE.
- Со SERIALIZABLE: испреплетеното извршување на трансакциите ќе се придржува до нашето сфаќање за подредливост.
  - Сепак, било која трансакција која се извршува на пониско ниво, во тие случаи серијабилноста може да биде нарушена.

Потенцијални проблеми со пониските изолациски нивоа

- **Валкано читање (Dirty Read)**:
  - Трансакцијата T_1 може да го прочита ажурирањето од трансакцијата T_2, кое сè уште не е потврдено.
  - Ако T_2 е неуспешна или се прекине, тогаш T_1 би прочитала вредност што не постои и е погрешна.
- **Неповторливо читање (Nonrepeatable Read)**:
  - Трансакцијата T_1 може да прочита определена податочна ставка од некоја табела.
  - Ако друга трансакција T_2 подоцна ја ажурира таа ставка, а T_1 ја прочита истата ставка повторно, T_1 ќе добие различна вредност.
    - Пример: Ако T_1 ја прочита платата на Smith. Потоа, T_2 ја променува платата на Smith. Ако T_1 повторно ја прочита платата на Smith, тогаш ќе добие различна вредност за платата на Smith.

Потенцијални проблеми со пониските изолациски нивоа

- **Фантоми (Phantoms)**:
  - Нови редици се читаат со истата наредба за читање (SELECT) со истите услови.
    - Трансакцијата T_1 може да прочита едно множество на редици од табелата, врз основа на истите услови одредени со SQL WHERE делот.
    - Во меѓувреме, некоја трансакција T_2 вметнува во истата табела нов ред којшто исто така би го задоволувал условот во WHERE делот на T_1.
    - Ако T_1 се повтори, тогаш T_1 ќе види нов ред којшто претходно не постоел, наречен фантом (phantom).

Подршка за трансакции кај SQL

- Можни нарушувања на подредливоста:

Table 21.1 Possible Violations Based on Isolation Levels as Defined in SQL

	Type of Violation		
Isolation Level	**Dirty Read**	**Nonrepeatable Read**	**Phantom**
READ UNCOMMITTED	Yes	Yes	Yes
READ COMMITTED	No	Yes	Yes
REPEATABLE READ	No	No	Yes
SERIALIZABLE	No	No	No

Пример

```
EXEC SQL WHENEVER SQLERROR GO TO UNDO;
EXEC SQL SET TRANSACTION
          READ WRITE
          DIAGNOSTICS SIZE 5
          ISOLATION LEVEL SERIALIZABLE;
EXEC SQL
          INSERT INTO EMPLOYEE (FNAME, LNAME, SSN, DNO, SALARY)
          VALUES ('Robert','Smith','991004321',2,35000);
EXEC SQL
          UPDATE EMPLOYEE SET SALARY = SALARY * 1.1
          WHERE DNO = 2;
EXEC SQL COMMIT;
GOTO  THE_END;
UNDO: EXEC SQL ROLLBACK;
THE_END:  ... ;
```


Преглед на лекцијата

- Поими за трансакциите
- Посакувани својства на трансакциите
- Одредување на распореди врз основа на подредливост
- Подршка за трансакции кај SQL

ПРИМЕР

- Нека се дадени три трансакции
  T_1: $r_1(X)$; $r_1(Z)$; $w_1(X)$;
  T_2: $r_2(Z)$; $r_2(Y)$; $w_2(Z)$; $w_2(Y)$;
  T_3: $r_3(X)$; $r_3(Y)$; $w_3(Y)$;

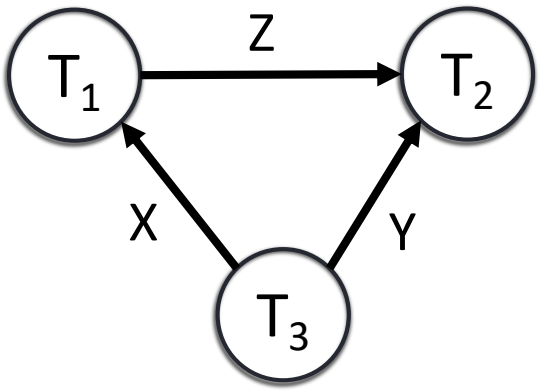
  и двата распореди:
  S_1: $r_1(X)$; $r_2(Z)$; $r_1(Z)$; $r_3(X)$; $r_3(Y)$; $w_1(X)$; $w_3(Y)$; $r_2(Y)$; $w_2(Z)$; $w_2(Y)$;
  S_2: $r_1(X)$; $r_2(Z)$; $r_3(X)$; $r_1(Z)$; $r_2(Y)$; $r_3(Y)$; $w_1(X)$; $w_2(Z)$; $w_3(Y)$; $w_2(Y)$;

- Проверете дали распоредите S_1 и S_2 се серијабилни или не?

ПРИМЕР

S_1: $r_1(X)$; $r_2(Z)$; $r_1(Z)$; $r_3(X)$; $r_3(Y)$; $w_1(X)$; $w_3(Y)$; $r_2(Y)$; $w_2(Z)$; $w_2(Y)$;

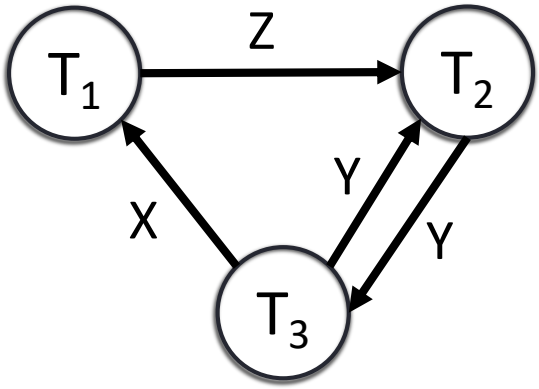
T1	T2	T3
read(X)		
	read(Z)	
read(Z)		
		read(X)
		read(Y)
write(X)		
		write(Y)
	read(Y)	
	write(Z)	
	write(Y)	



ПРИМЕР

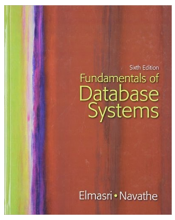
S_2: $r_1(X)$; $r_2(Z)$; $r_3(X)$; $r_1(Z)$; $r_2(Y)$; $r_3(Y)$; $w_1(X)$; $w_2(Z)$; $w_3(Y)$; $w_2(Y)$;

T1	T2	T3
read(X)		
	read(Z)	
		read(X)
read(Z)		
	read(Y)	
		read(Y)
write(X)		
	write(Z)	
		write(Y)
	write(Y)	



Користена литература

- **Глава 17** (609 - 639)



- **Глава 21** (743-775)

- **Глава 18** (969 - 1044)
- **Глава 19** (1047 - 1081)