732G16 Databaser; Design och programmering

Fö 7 Transaktionshantering och säkerhet

2026

LINKÖPINGS UNIVERSITET

Innehåll

- Säkerhetsproblem
  - Åtkomst vs tillgänglighet
  - Transaktioner
- Fleranvändarproblem
  - mer Transaktioner
  - Låsning

Säkerhetsproblem

- Data i databasen
  - ska vara korrekt
  - beständig
  - måste vara åtkomlig när den behövs
  - får inte spridas till obehöriga
- Men: om datorn hänger sig, hårddisken kraschar, obehöriga försöker ta sig in… eller någon är bara klantig?

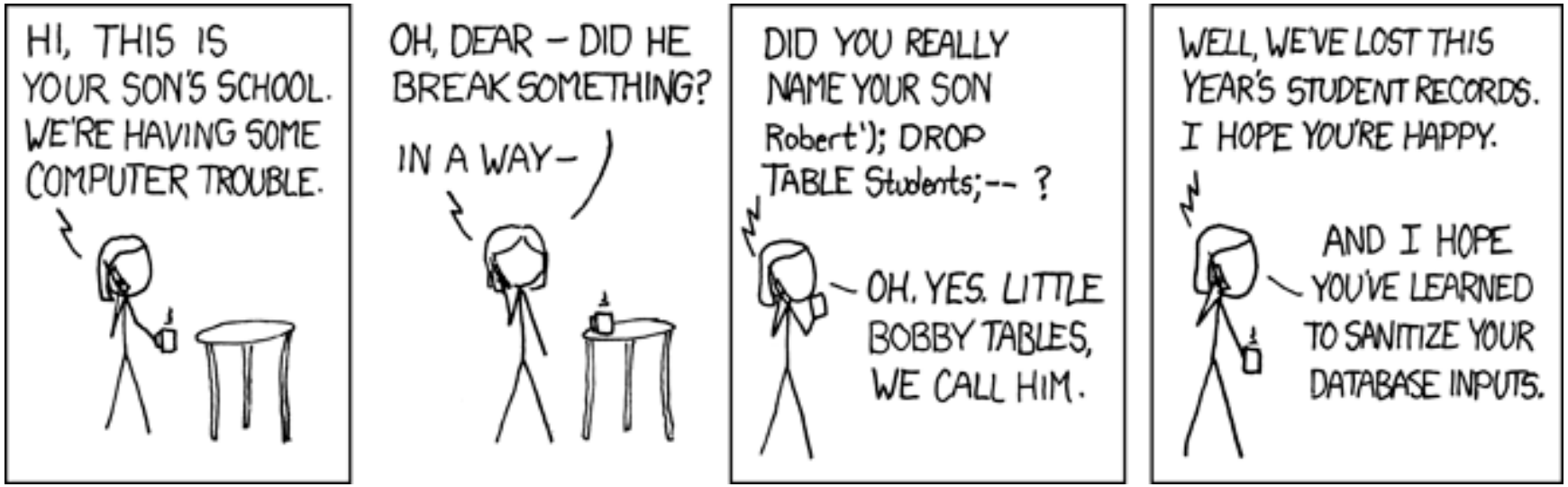
Obehörig åtkomst

- Databashanteringssystemet har mekanismer:
  - Inloggning
  - Säkerhetsnivåer (olika användare olika behörighet)
    SQL-kommandon:
    `grant` och `revoke`
    `grant` `select` **`on`** `employee` **`to`** `user-id`
  - vyer

Statistiska databaser och integritet

- Känsliga data som samlas in för analys
  - ex politisk åsikt - utan att kunna identifiera vad grannen röstade på
- Krav: kunna analysera men inte identifiera individer
  - vyer?
  - bara tillåta aggregerade frågor?

SQL-injektion



Källa: xkcd.com/327/

Korrekthet: Exempel

Du ska betala en räkning genom att överföra pengar från ditt konto till mottagarens.

- Begär överföring från mitt konto till annans.
- Tryck “bekräfta”
  - beloppet dras från mitt konto
  - sedan hänger sig systemet.

- insättningen registrerades inte!

Hur datorn fungerar

- En processor
- Flera program/appar körs ”parallellt”
- Round Robin

- Ett program kan avbrytas när som helst!

Transaktion

- Definition: Logiskt sammanhängande serie interaktioner med databasen, som är:
  - Atomär (**A**tomic) : odelbar
  - Konsistent (**C**onsistent): integritetsbevarande
  - Isolerade (**I**solated): oberoende av annat i db.
  - Bestående (**D**urable): inte kunna försvinna.

Transaktion

- Ex: Överföring mellan två konton (X och Y)
- Interaktioner: läsning och skrivning i databasen!
- Pseudokod:

```
Läs(X)
X = X - uttag
Skriv(X)
Läs(Y)
Y = Y + uttag
Skriv(Y)
```

Transaktion

- Utför allt eller inget: markera start och slut!

- Pseudokod:

```
Start transaction
Läs(X)
X = X - uttag
Skriv(X)
Läs(Y)
Y = Y + uttag
Skriv(Y)
Commit
```

Loggfil - spara interaktionen med db

```
start transaction
Läs(X)
X = X - uttag
Skriv(X)
Läs(Y)
Y = Y + uttag
Skriv(Y)
commit transaction
```

```
begin (t234)

read (t234, X)


write (t234, X, 360, 260)

read (t234, Y)


write (t234, Y, 120, 220)

commit (t234)
```

Återställa transaktion: Rollback

- **Om** transaktionen avbryts innan commit:
  - Återställ de tidigare värdena med hjälp av loggfilen
- Hur långt tillbaka som helst?
  - `Checkpoint` (inga öppna transaktioner)

Loggfilen:

```
start (t234)
read (t234, X)
write (t234, X, 360, 260)
read (t234, Y)
… eof …
```

Hårddiskkrasch - Backup

- Kopia av databasen på annan disk!
- Loggfilen ska vara på annan hårddisk än databasen!

1. Läs tillbaka backupen
2. Läs igenom Loggfilen:
   - Transaktioner som slutförts efter senaste backup: repeteras
   - Transaktioner som ej slutförts: ignoreras

OBS: Skriv till loggfilen först, sedan till databasen!

Transaktion

A (atomär)	-	`Start`/`Commit`/Loggfil/Rollback
C (konsistent)	-	A + kontroll av integritetsvillkor i transaktionen
I (oberoende)	-	?
D (bestående)	-	Backup + Loggfil

Problem med parallella transaktioner

Varor

Artikelnr	Namn	i lager	Pris
2398475	Rosor, 10-p	3	79,9
9834576	Tulpan, 10-p	15	49,9

Dagens försäljning

Artikelnr	Antal	Dag
2398475	3	230506
9834576	1	230507
2398475	2	230507

Ex: Varutabellen: uppdatering

Kassaförsäljning:

X = antal rosor i lager

Y = dagens försäljning av rosor

```
start transaction
Läs(X)
X = X - 1
Skriv(X)
Läs(Y)
Y = Y + 1
Skriv(Y)
commit transaction
```

Lagerhantering:

X = antal rosor i lager

```
start transaction
Läs(X)
X = X + Levererat ant
Skriv(X)
commit transaction
```

Tid	Kassa försäljning	Lagerhantering
1		Start transaction
2		Läs(X)
3		X = X + Levererat
4	Start Transaction	
5	Läs (X)	
6	X = x-1	
7		Skriv (X)
8		Commit Transaction
9	Skriv (X)	
10	Läs (Y)	
11	Y = Y +1	
12	Skriv (Y)	
13	Commit Transaction	

Bortkastad uppdatering

Ex: Varutabellen: Automatisk beställning

Kassaförsäljning:

X = antal rosor i lager

```
start transaction
Läs(X)
X = X - 1
Skriv(X)
Läs(Y)
Y = Y + 1
Skriv(Y)
commit transaction
```

Automatisk beställning:

Körs vid ändring av antal i lager

X = antal rosor i lager

```
Läs(X)
om X<3
så skicka beställning
på nya rosor
```

Tid	Kassaförsäljning	Automatisk beställning:
1	Start Transaction	
2	Läs (X)	
3	X = x-1	
4	Skriv (X)	
5		Läs(X)
6		X <3
7		Beställ!
8	**Rollback**	
9	~~Läs (Y)~~	
10	~~Y = Y +1~~	
11	~~Skriv (Y)~~	
12		

Smutsig läsning

Ex: Konton: Överföring och summering

Överföring mellan egna konton:

X = lönekonto

Y = sparkonto

```
start transaction
Läs(X)
X = X - 1000
Skriv(X)
Läs(Y)
Y = Y + 1000
Skriv(Y)
commit transaction
```

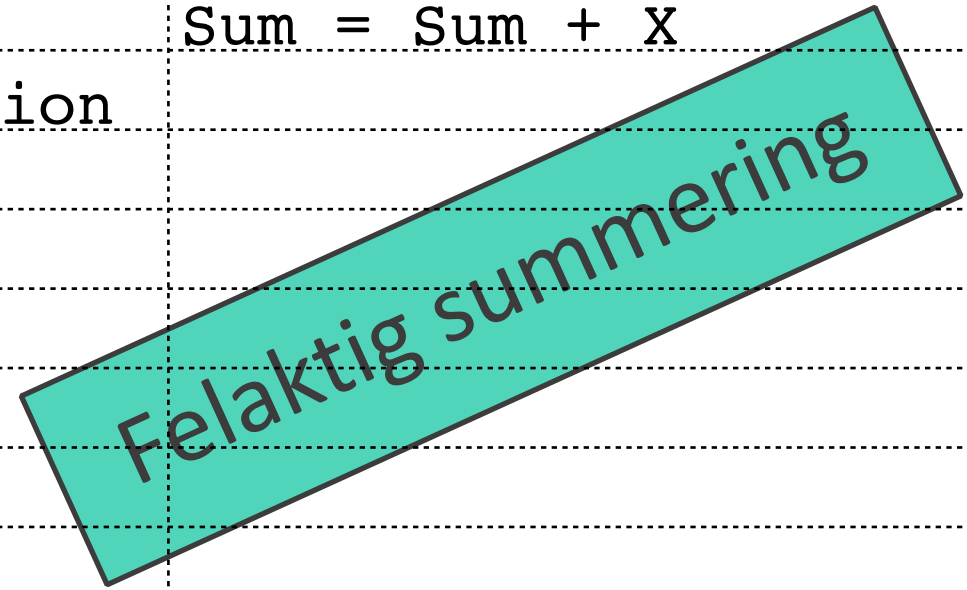
Summering av tillgångar:

X = lönekonto

Y = sparkonto

```
start transaction
sum = 0
Läs(X)
sum = sum + X
Läs(Y)
sum = sum + y
commit transaction
```

Tid	Överföring	Summering
1		Start transaction
2		Sum=0
3		Läs (X)
4		Sum = Sum + X
5	Start Transaction	
6	Läs (X)	
7	X = x-m	
8	Skriv (X)	
9	Läs (Y)	
10	Y = Y +m	
11	Skriv (Y)	
12	Commit Transaction	
13		Läs (Y)
14		Sum = Sum + Y



Parallella transaktioner: Lås

- Bara den transaktion som låst objektet får komma åt det
- `Lås(X), LåsUpp(X)`
- Vad X är avgörs av låsets granularitet (tabell/rad/cell)

- (Alternativ: t.ex. serialiserbarhet och tidsstämpling, studeras ej)

Låsning - Binära lås

- Två tillstånd: **Låst, Olåst**.
- **Protokoll** för binära lås (måste följas!):
  1. `Lås(X)` måste utföras innan någon `Läs(X)` eller `Skriv(X)`-operation utförs.
  2. `LåsUpp(X)` måste utföras när läsning och skrivning av X är klar.
  3. man får inte göra `Lås(X)` om man redan har låst X.
  4. man får inte göra `LåsUpp(X)` om man inte har låst X för tillfället.

Binära lås

- Mellan `Lås(X)` och `LåsUpp(X)` kan bara den transaktion som gjorde `Lås(X)` komma åt X.

- Om någon annan transaktion försöker får de vänta!

  – Tillgänglighet?

- Lås så kort tid som möjligt!

Exempel lås

Försäljning:

```
Start transaction
Läs(X)
X = X-1
Skriv(X)

Läs(Y)
Y = Y + 1
Skriv(Y)
Commit
```

Lagerhantering:

```
Start trans
Läs(X)
X = X + Levererat
Skriv(X)
Commit
```

Exempel lås

Kassaförsäljning:

```
Start transaction
Lås(X)
Läs(X)
X = X-1
Skriv(X)
LåsUpp(X)
Lås(Y)
Läs(Y)
Y = Y + 1
Skriv(Y)
LåsUpp(Y)
Commit
```

Lagerhantering:

```
Start trans
Lås(X)
Läs(X)
X = X + Levererat
Skriv(X)
LåsUpp(X)
Commit
```

Läs- och skrivlås

- Parallell läsning fungerar bra: skilj på läsning och skrivning.
- `LäsLås(X), SkrivLås (X), LåsUpp(X)`

- LäsLås går bara igenom om inget SkrivLås är satt.
- SkrivLås går bara igenom om inget lås alls är satt.

- Dvs: flera transaktioner kan ha låst X för läsning, men bara en kan låsa för skrivning (och bara om ingen annan har låst alls).

Läs- och skrivlås: protokoll:

1. Innan någon `Läs(X)`-operation utförs måste `LäsLås(X)` **eller** `SkrivLås(X)` utföras.
2. Innan någon `Skriv(X)`-operation utförs måste `SkrivLås(X)` utföras.
3. `LåsUpp(X)` måste utföras när läsning och skrivning av X är klar.
4. man får inte göra `LäsLås(X)` om man redan har låst X.
5. man får inte göra `SkrivLås(X)` om man redan har låst X för skrivning.
6. man får inte göra `LåsUpp(X)` om man inte har låst X för tillfället.

Men...

Tid	Överför	Summer
1		Sum=0
2		LäLås(X)
3		Läs(X)
4		Sum=Sum+x
5		LåsUpp(X
6	SkLås(X)	
7	Läs(X)	
8	x=x-m	
9	Skriv(x)	
10	LåsUpp(X	
11		

Tid	Överför	Summer
12	SkLås(Y)	
13	Läs Y	
14	y=y+m	
15	Skriv(y)	
16	LåsUpp(Y	
17		LäLås(Y)
18		Läs(y)
19		Sum=Sum+
20		y
21		LåsUpp(y
22		

Felaktig summering

Tvåfaslåsning

- Protokoll som tidigare plus:
- Två faser:

  1. Lås

  2. Lås upp
- Alltså: Inte låsa upp något förrän allt som behövs för transaktionen har låsts!

- Not: fungerar både med binära lås och läs/skrivlås

Men…

Tid	**Kassaförs.**	**Lagerh**
1	Start t	
2	**Lås(X)**	
3	Läs(X)	
4	x=x-1	
5	Skriv(x)	
6	**Lås(Y)**	
7	**LåsUpp(X**	
8		Start t
9		Lås(X)
10		Läs(X)
11		x=x+lev

Tid	**Kassaförs.**	**Lagerh**
12		Skriv(x)
13		LåsUpp(
14		X
15		Commit
16	ROLLBACK	
17	~~Läs Y~~	
18	~~y=y+1~~	
19	~~Skriv(y)~~	
20	~~LåsUpp(Y~~	

Read uncommited

Kaskad-rollback?

- Om en committad transaktion beror av en annan som gör Roll-back?
- Kaskad-rollback: Rulla tillbaka även den commit-ade transaktionen.

- Men då förlorar vi ju D (Durable, bestående)??
- Tvåfaslåsning inte tillräcklig!

Förbättrad tvåfaslåsning

- Rigorös tvåfaslåsning:
  - Släpp inte något lås förrän hela transaktionen är commitad!
- Strikt tvåfaslåsning:
  - Läslås kan släppas i upplåsningsfasen men skrivlås släpps inte förrän hela transaktionen är committad
- Nu är våra transaktioner ACID!

Men...

Tid:	**Trans 1:**	**Trans 2:**
1	SkrivLås(X)	
2		SkrivLås(Y)
3	…	…
4	SkrivLås(Y)	
5	VÄNTAR!	SkrivLås(X)
6	…	VÄNTAR!
.		…
.	LåsUpp(X)	LåsUpp(Y)
.	LåsUpp(y)	LåsUpp(X)

Deadlock!

Dödläge (deadlock):

- Definition:

  Korsvis (eller cirkulär, om flera transaktioner är inblandade) låsning av objekt i databasen,

  sådan att ingen kan släppa en artikel förrän den fått låsa en artikel som är låst av någon som väntar på den artikel man redan låst.

Deadlock: strategier

- Förebyggande:
  - Konservativ tvåfaslåsning (lås allt på en gång)
  - Dataobjekten låses alltid i en viss ordning
- Upptäckande
  - Time-out
  - undersöka wait-for-grafen

Transaktioner och säkerhet - begrepp

- Säkerhet: inloggning, rättigheter, back-up
- Transaktioner är ACID (odelbara, konsistens-bevarande, isolerade och bestående).
- Mekanismer för detta:
  - Backup, Loggfil, start/commit,
  - Rollback (kaskad-rollback)
  - Lås: Binära lås, Läs/Skrivlås
    - Protokoll: grund, tvåfas, strikt/rigorös
    - Deadlock, strategier

Frågor?

www.liu.se

LINKÖPINGS UNIVERSITET