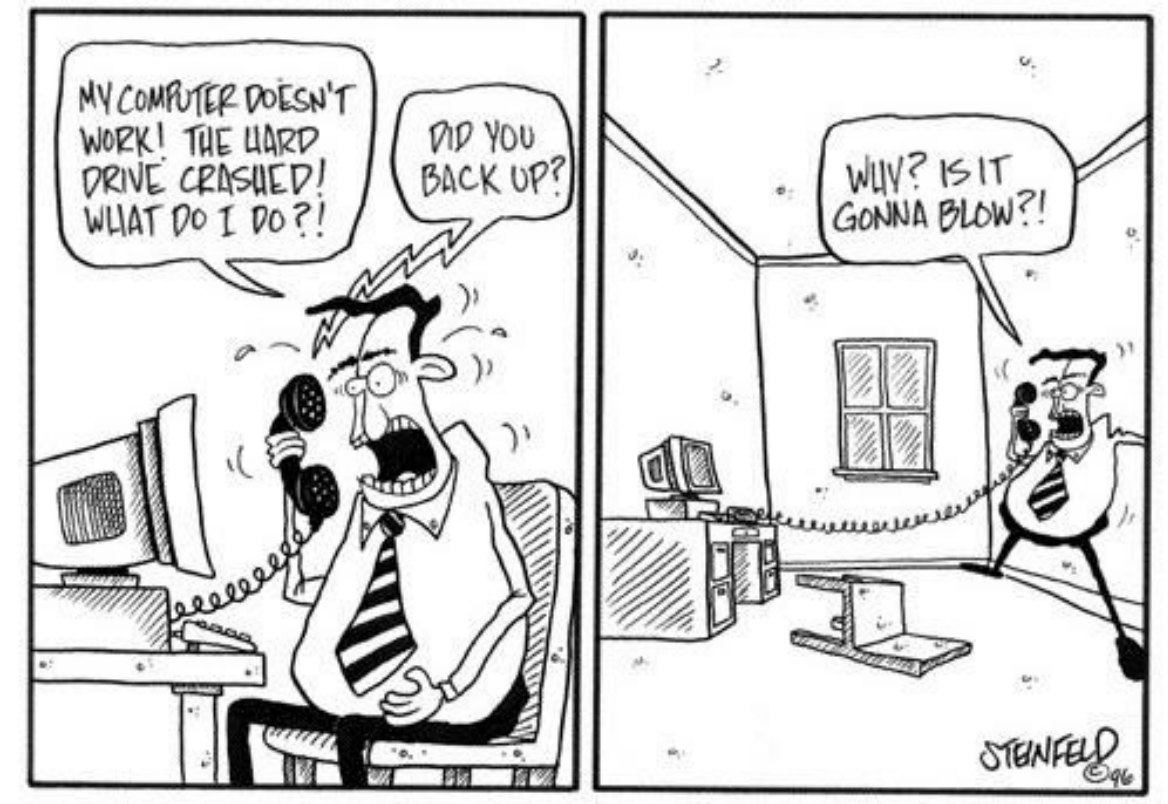
Universitetet i Sørøst-Norge

Leksjon 9: Backup og Recovery



Jarle Håvik

DAT2000 – Database 2

Agenda

- Gjennomgang av transaksjoner
- Introduksjon
- Sikkerhetskopiering av databasen
- Teknikker for sikkerhetskopiering av databaser
- Database gjennoppretting
- Teknikker for gjenoppretting av databaser

Kunnskaps- og ferdighetsmål

- Kunnskap:
  - *sentrale oppgaver i databaseadministasjon og -sikkerhet, som autorisasjon og tilgangskontroll, sikkerhetskopiering (backup) og gjenoppretting (recovery), replikering og synkronisering*

- Ferdigheter:
  - utføre grunnleggende administrasjon og drift av et databasesystem

Litteratur

- Silberschatz, Abraham et.al: Database System Concepts
  - Kapittel 19: Recovery System (s. 907 -948)
- Kristoffersen, Bjørn: Databasesystemer
  - Kapittel 11.2.1 Sikkerhetskopiering og gjenoppbygging (s. 316 – 318)
- Dubois, Paul: MySQL
  - kapittel 14 : Database Maintenance, Backups, and Replication
- Korry og Susann Douglas: PostgreSQL, 2nd edition: Kapittel 21. PostgreSQL Administration
- EDB: PostgreSQL Backup & Recovery: The Ifs, Ands & Buts
- PostgreSQL manual : Chapter 25. Backup and Restore

Introduction

- Databaseadministratorer har stort ansvar for sikkerheten til databasen og maksimerer tilgjengeligheten for brukere:
  - Angi tilgangskontroller på databaseobjekter og brukere
  - Beskytte data mot potensielle trusler og etterfølgende trusler (angrep)
  - Å bruke sikkerhetskopierings- og gjenopprettingsprosedyrer ved regelmessig å lage kopier av databasen på et sikkert sted for å gjenopprette databasen i tilfelle feil

Typer feil

- Datamaskinsfeil: maskinvare, programvare eller nettverksfeil.
  - System krasj noe som resulterer i tap av primærminne.
  - Mediefeil, noe som resulterer i tap av deler av sekundær lagring.
  - Programvarefeil.
- Transaksjonsfeil.
- Naturkatastrofer: jordskjelv, orkaner.
- Menneskelig svikt: Uforsiktighet eller utilsiktet ødeleggelse av data eller fasiliteter.
- Ondsinnede handlinger.
- Sabotasje.

Database Backup

Prosessen med periodisk kopiering av databasen og loggfilen til offline lagringsmedier.

Enhver hendelse av feil kan gjøre databasen ubrukelig. Dermed blir sikkerhetskopien og detaljene fanget i loggfilen brukt til å gjenopprette databasen til den siste mulige, konsistente tilstanden.

Journalføring

Prosessen med å opprettholde og vedlikeholde en loggfil (eller journal) over alle endringer som er gjort i databasen for å gjøre det mulig å gjenopprette effektivt i tilfelle en feil.

Loggen er lagret på disken, ikke i minnet
- Hvis systemet krasjer, blir det bevart

Loggbar aktivitet:
–Start
–Commit
–Rollback
–oppdatert post
–Checkpoint

Database Backup

DBMS-er gir verktøy for å gjenopprette fra feil og gjennskape databasen til en konsistent tilstand:

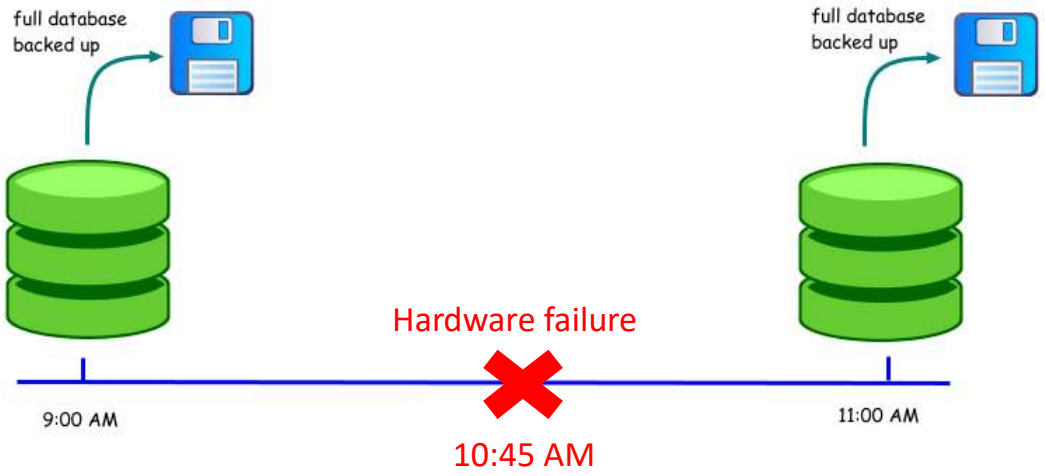
- Backup-fasiliteter for gjenoppretting av en database fra feil
- Loggingsmetoder og verktøy for å holde oversikt over gjeldende status for transaksjoner og databaseendringer, og dermed støtte til gjenopprettingsprosedyrer

Backup vs. Journalføring

- Med journalføring kan databasen gjenopprettes til sin siste konsistente tilstand ved hjelp av sikkerhetskopier og loggfilen.
- Uten journalføring er den eneste måten å gjenopprette databasen på å gjenopprette databasen ved å bruke den siste sikkerhetskopien, men eventuelle endringer som er gjort i databasen etter den siste sikkerhetskopien, vil gå tapt!

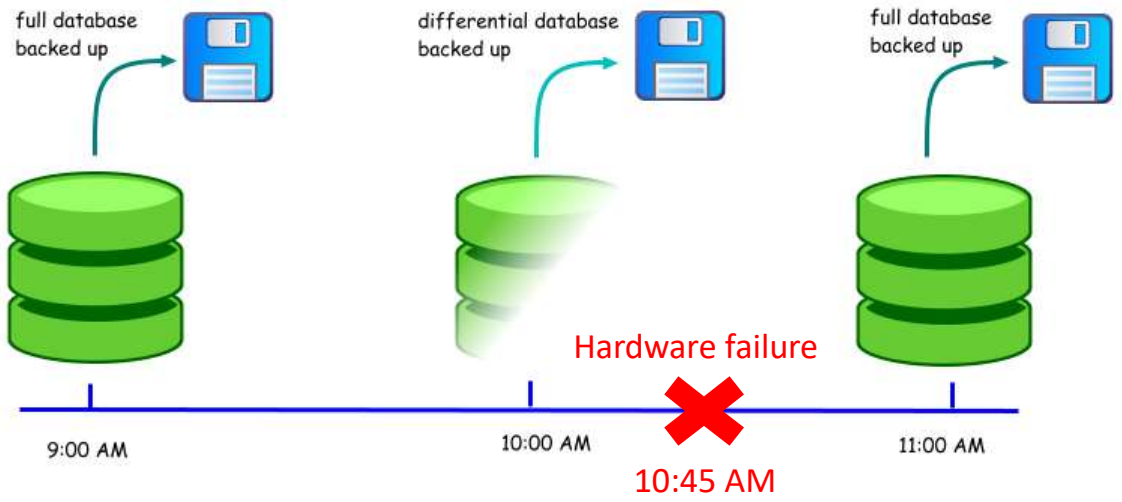
Eksempel: Full backup

- Anta at det var en maskinvarefeil klokken 10:45. Hvis du bare brukte sikkerhetskopier, hadde du mistet 105 minutters arbeid.
- den eneste måten å gjenopprette databasen på er å gjenopprette databasen ved hjelp av den siste sikkerhetskopien som er gjort, men eventuelle endringer som er gjort i databasen etter den siste sikkerhetskopien vil gå tapt!



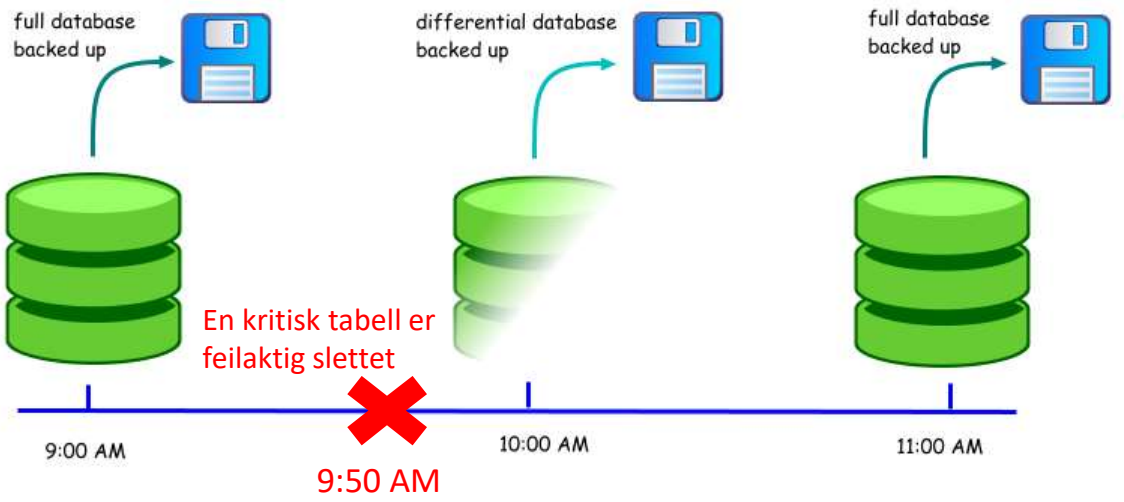
Eksempel 2: Differensiell backup

- Du kan planlegge at differensielle sikkerhetskopier skal kjøre med jevne mellomrom, og i dette tilfellet vil du miste 45 minutters arbeid.
- Differensiell sikkerhetskopi er en kumulativ sikkerhetskopi av alle endringer som er gjort siden forrige fullstendige sikkerhetskopiering.



Eksempel 2: Differensiell backup ..

- Anta at en bruker slettet en kritisk tabell klokken 09.50. Kan en gjennopprette databasen til rett før sletting?
- Differensial-sikkerhetskopien inneholder bare de endrede datasidene. Den kan ikke brukes til å gjenopprette til et bestemt tidspunkt. Du blir nødt til å gjenopprette databasen til klokka 09.00 og gjøre om 49 minutters arbeid. Deretter må du også gjøre om arbeidet som ble utført etter slettingen, helt til feilen ble oppdaget.

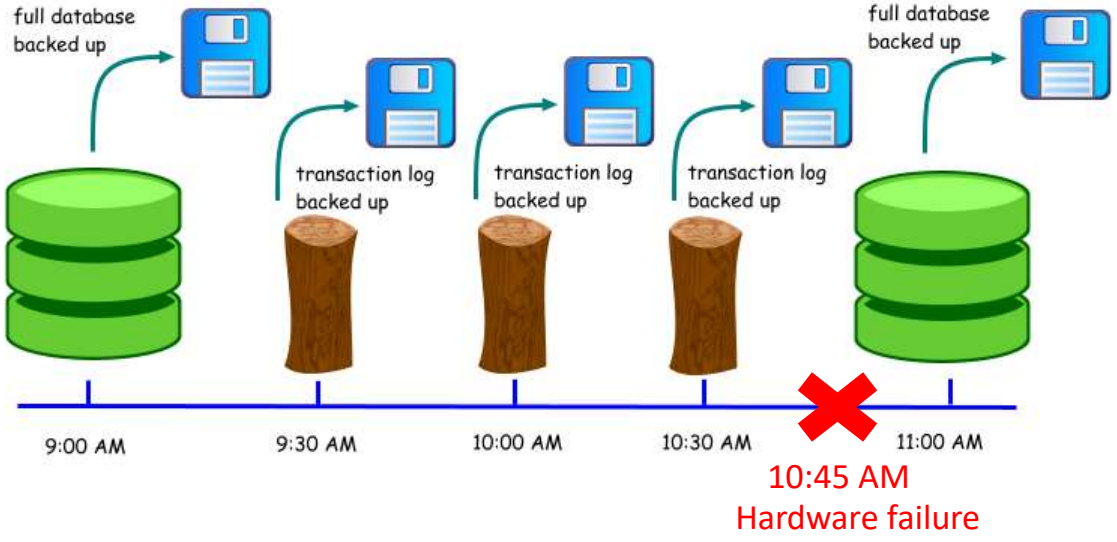


Eksempel: Bruk av logger

Anta nå at transaksjonsloggen er sikkerhetskopiert hvert 30. minutt.

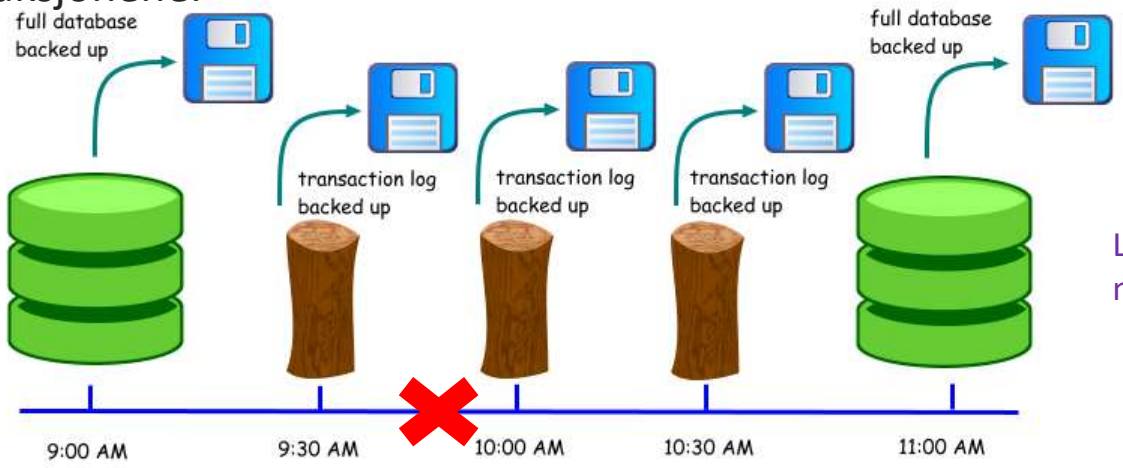
Hvis det oppstår en maskinvarefeil klokka 10:45, vil en miste 15 minutters arbeid.

Vi kan bruke full sikkerhetskopi av databasen klokka 9.00 og så bruke transaksjonsloggene for å bringe databasen til sin tilstand kl. 10.30.



Eksempel: Logger ..

Hva om noen kritiske data ble slettet klokka 09.50?
Ved å bruke transaksjonsloggen som ble laget klokka 10.00, kan vi gjenopprette databasen til tilstanden klokka 09.49. Vi vil fremdeles måtte gjøre om arbeidet som ble utført fra tidspunktet for sletting til feilen ble oppdaget, ettersom vi ikke kan gjenopprette opp til 09:49, hoppe over transaksjonene 09:50 og gjenopprette de senere transaksjonene.



9:50 AM
Kritiske data ble slettet ved en feil

Log is backed up every 30 minutes.

Viktige punkter en må vurdere når du en plan for backup

- Hvor stort datatap er akseptabelt?
- Vil sikkerhetskopiene påvirke vanlig databasedrift?
- Er det et vedlikeholdsvindu der sikkerhetskopieringsoperasjoner med stor innvirkning kan utføres?
- Hva er den akseptable tiden for gjennoppretting?
- Er det behov for å kunnr gjenopprette til et bestemt tidspunkt ?, der sikkerhetskopier av transaksjonslogg er påkrevd
- Er det tilstrekkelig lagringsplass for å lagre alle nødvendige sikkerhetskopier?

DB Gjennoppretting
(Recovery)

Database Recovery

- Prosessen med å gjenopprette databasen til riktig tilstand i tilfelle en feil.
- Behov for gjenopprettingskontroll
- To typer lagring: flyktig (primærminne) og ikke-flyktig.
- Flyktig lagring overlever ikke systemkrasj.
- Stabil lagring representerer informasjon som er replikert i flere ikke-flyktige lagringsmedier med uavhengige feilmodus.

Database Recovery ..

- Pålitelighet: er motstandsdyktigheten til DBMS mot forskjellige typer feil og dens evne til å komme seg fra dem.
- Viktige effekter av en feil som bør vurderes for utvinning:
  - Tap av primærminne og databasebuffere
  - Tap av diskkopien av databasen

Transaksjoner og recovery

Transaksjoner representerer grunnleggande gjenopprettingsenhet.

- Recovery manager ansvarlig for atomicitet og holdbarhet.
- Hvis det oppstår feil mellom commit og databasebuffere som flushes til sekundær lagring, må recovery manageren gjøre om (redo) transaksjonens oppdateringer for å sikre holdbarhet.

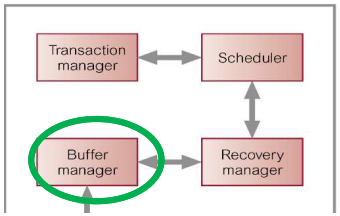
- Ønsket atferd etter at systemet har startet på nytt:
- T1, T2 & T3 bør være holdbare (durable).

Transaksjoner og recovery ..

- Hvis transaksjonen ikke var committed ved feiltidspunktet, må recovery manager angre (undo) alle effekter av transaksjonen for å bevare atomicity.
- Delvis angre - bare én transaksjon må angres.
- Global angre - alle transaksjoner må angres.

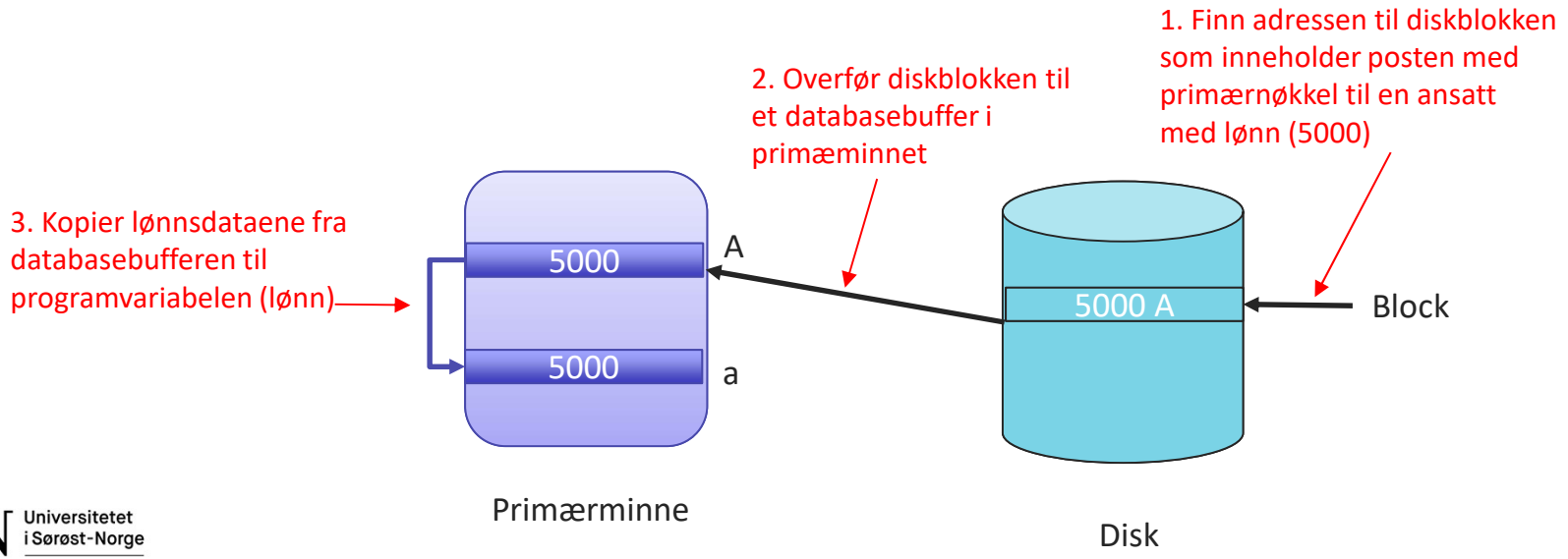
- Ønsket atferd etter at systemet har startet på nytt:
- T1, T2 & T3 bør bevares (durable.)
  - T4 & T5 bør aborteres (effektene av disse bør ikke vises).

Database Recovery



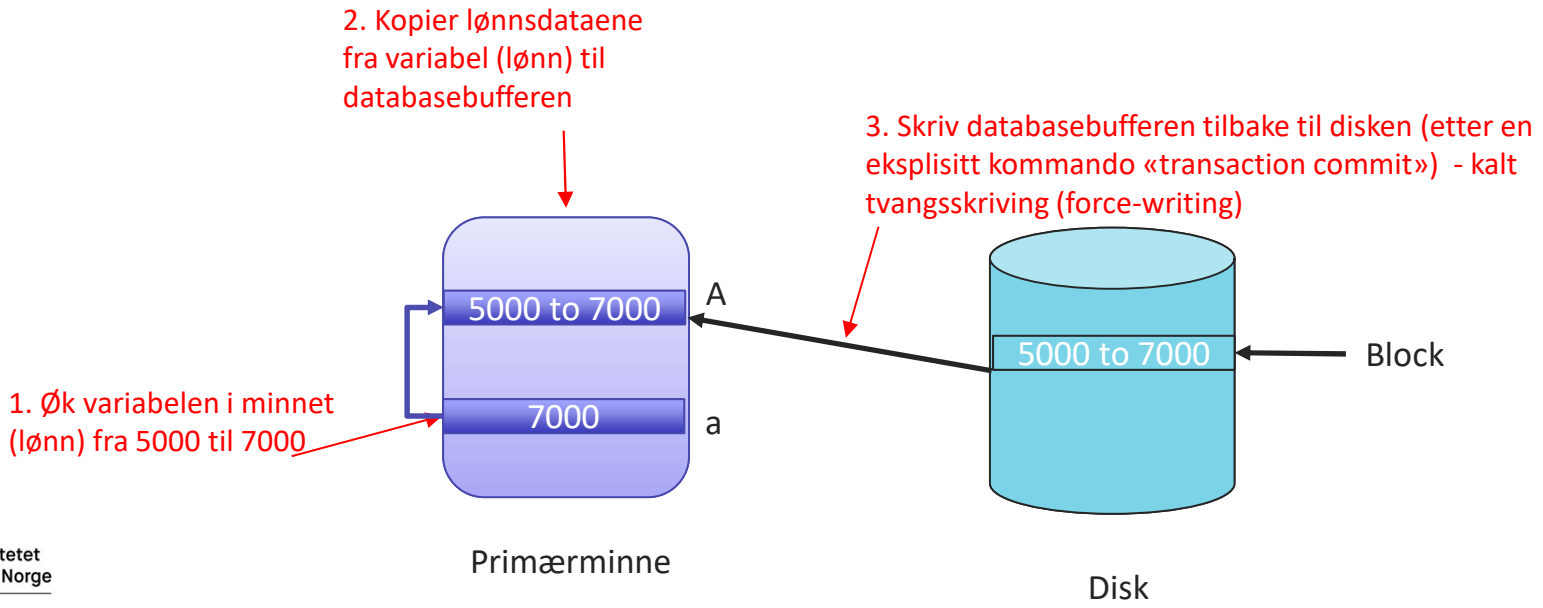
Eksempel: les operasjonene til en transaksjon

Database-buffere opptar et område i primærminnet der data overføres til og fra sekundærlager.

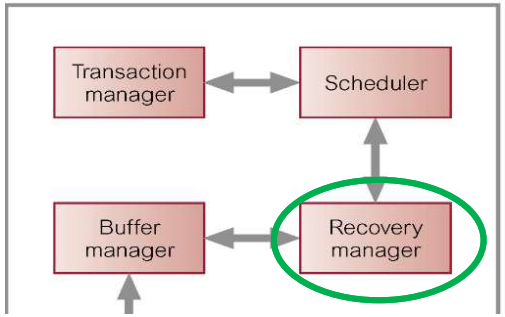


Database Recovery

Eksempel: skrive operasjoner til en transaksjon

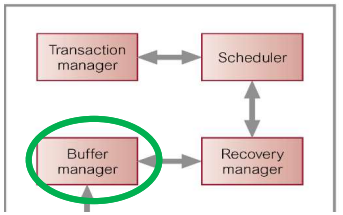


Undo/redo transaksjoner



- I tilfelle feil, så må recovery manager bestemme statusen for transaksjonen som utførte skrivingen på tidspunktet for feilen:
  - Den gode transaksjonen som utstedte sin «commit», får sine oppdateringer gjort om igjen - redone til databasen (rollforward) → sikrer holdbarhet (durability)
  - De «dårlege» transaksjonene som ikke er «committed», får sine oppdateringer til databasen ugjort - undone (rollback) → sikrer atomicity

Undo/redo transaksjoner …



- Buffer management: viktig for gjenopprettingsprosessen for å bestemme «hvilke buffere som skal tvinges til disk (force-write) for å gi plass til at flere data kan leses fra disken. Data bør heller ikke leses fra disken hvis de allerede er i databasebufferen.

Recovery fasiliteter

- DBMS bør tilby følgende fasiliteter for å hjelpe til med recovery:
  - Sikkerhetskopieringsmekanisme, som lager periodiske sikkerhetskopier av databasen.
  - Loggingsverktøy som holder oversikt over gjeldende status for transaksjoner og databaseendringer.
  - Checkpoint-anlegg, som gjør at oppdateringer til pågående database kan gjøres permanente.
  - Recovery manager, som lar DB S gjenopprette databasen til konsistent tilstand etter en feil.

Logg-fil

- Inneholder informasjon om alle oppdateringer til databasen:
  - Transaksjonsposter
  - Checkpoint poster.
- Brukes ofte til andre formål (for eksempel revisjon).

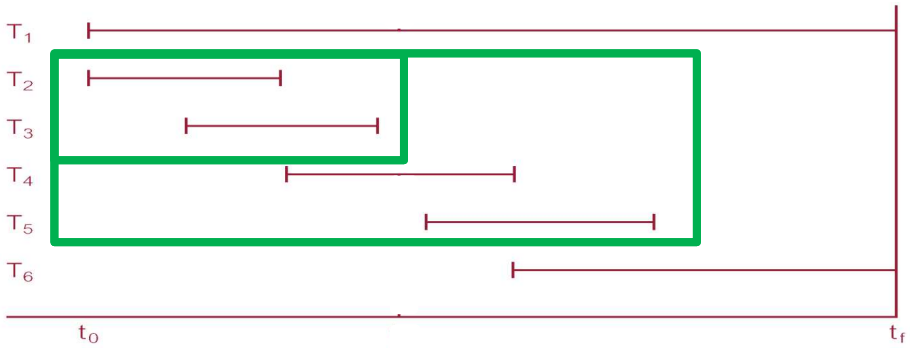
Logg-filer..

- Transaksjonsposten kan inneholde:
  - Transaksjonsidentifikator.
  - Type loggpost det er , (transaction start, insert, update, delete, abort, commit).
  - Identifikator av dataelement som er påvirket av databaseaksjon(insert, delete, og update handlinger).
  - Førbilde av dataelementet.(gammel verdi)
  - Etterbilde av dataelement (ny verdi).
  - Logg administrasjonsinformasjon.

Tid	Time	Operation	Object	Before image	After image	pPtr	nPtr
T1	10:12	START				0	2
T1	10:13	UPDATE	STAFF SL21	(old value)	(new value)	1	8
T2	10:14	START				0	4
T2	10:16	INSERT	STAFF SG37		(new value)	3	5
T2	10:17	DELETE	STAFF SA9	(old value)		4	6
T2	10:17	UPDATE	PROPERTY PG16	(old value)	(new value)	5	9
T3	10:18	START				0	11
T1	10:18	COMMIT				2	0
	10:19	CHECKPOINT	T2, T3				
T2	10:19	COMMIT				6	0
T3	10:20	INSERT	PROPERTY PG4		(new value)	7	12
T3	10:21	COMMIT				11	0

Eksempel: Transaksjoner og Recovery

- . DBMS starter på tidspunktet t_0 og mislykkes på tidspunktet t_f. Transaksjoner T_2, T_3, T_4, og T_5 er bekreftet (committed) før feilen inntreffer. Dataene for T_2, og T_3, er skrevet til disken før feilen, men ingen informasjon er lagret om det i loggfilen. T1 og T6 var ikke bekreftet på tidspunktet for krasj
  - Hvilke transaksjoner vil bli angret (undone), og hvilke vil bli omgjort (redone)?



- T_1 og T_6 må angres (undone). I mangel på annen informasjon, må recovery manageren gjøre om (redo) T_2, T_3, T_4, og T_5.

Bruk av sjekkpunkt - Checkpointing (1 of 2)

Checkpoint

Synkroniseringspunkt mellom database og loggfil. Alle buffere er tvangsskrevet til sekundær lagring.

- Det opprettes sjekkpunktsposter som inneholder identifikatorer for alle aktive transaksjoner.
- Når det oppstår feil, må recovery manageren gjøre om alle transaksjoner som ble begått siden kontrollpunktet, og angre alle transaksjoner som var aktive på tidspunktet for krasj.

Sjekkpunkt ..

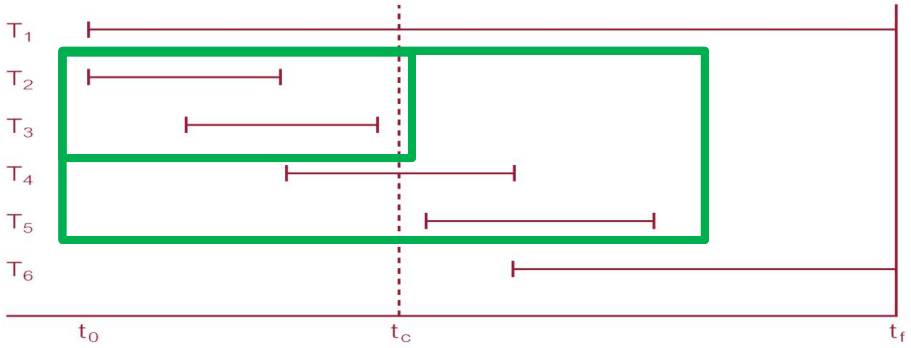
- Hvorfor trenger vi sjekkpunkter?
  - Når en feil oppstår, vet vi kanskje ikke hvor langt vi trenger å søke i loggfilen, og vi kan ende opp med å gjøre om transaksjoner som er trygt skrevet til disken.
  - Sjekkpunkter vil derfor begrense søkemengden i loggfilen og antall omgjøringer (redos) av transaksjoner som allerede er lagret på disken.

Eksempel: Sjekkpunkt …

Undo/redo transaksjoner med sjekkpunkt:

DBMS starter på tidspunktet t_0 og feiler ved tidspunkt t_f. Transaksjonene T_2, T_3, T_4, og T_5 have er bekreftet før feilen inntreffer. Data for T_2 og T_3 har blitt skrevet til disk før feilen oppstod, men ingen informasjon om dette er lagret i loggfilen. T_1, og T_6 er ikke bekreftet ved kræsjtidspunktet.

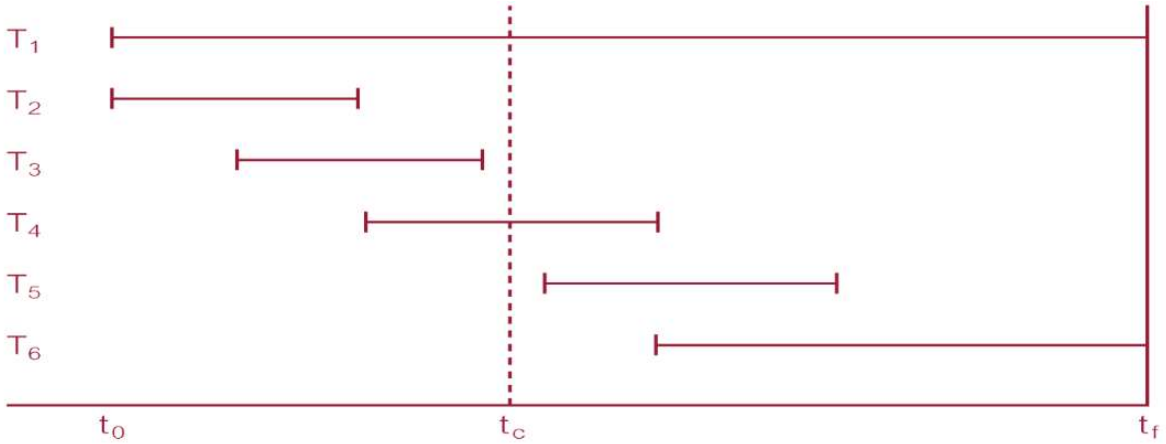
Hvilke transaksjoner vil nå bli angret (undone), og hvilke vil bli omgjort (redone)?



- Med sjekkpunkt ved t_c, vil endringer gjort av T_2 og T_3 ha blitt skrevet ut til sekundær minne. Vi har da:
  - Trenger bare redo på T_4 og T_5,
  - Undo (rollback transactionene) T_1 og T6.

Database Recovery

- Eventuelle transaksjoner som er commited og som har fått sine dataoppdateringer skrevet til disken før sjekkpunktet vil ikke bli omgjort (redone)!
- Det er bare transaksjoner som var aktive ved sjekkpunktet, og eventuelle etterfølgende transaksjoner som har sine «start» - og «committed» poster lagret i loggfilen. Som blir omgjort.
- Hvis en transaksjon er aktiv på tidspunktet for feilen, må den angres hvis den startet før siste kontrollpunkt.



Gjennopprettingsteknikker

- Dersom database er skadet:
  - Trenger å gjenopprette siste sikkerhetskopi av databasen og bruke oppdateringer av committede transaksjoner på nytt ved hjelp av loggfilen.
- Dersom database bare er inkonsistent:
  - Må angre endringer som forårsaket inkonsekvensen. Kan også trenge å gjøre om noen transaksjoner for å sikre at oppdateringer blir skrevet til sekundærlager.
  - Trenger ikke sikkerhetskopi, men kan gjenopprette database ved hjelp av før- og etterbilder i loggfilen.

Hovedgjenopprettingsteknikker

- Tre hovedgjenopprettingsteknikker:
  - Deferred Update
  - Immediate Update
  - Shadow Paging

Deferred Update

- Oppdateringer skrives ikke til databasen før etter at en transaksjon har nådd sitt forpliktelsespunkt.
- Hvis transaksjonen mislykkes før forpliktelsen, vil den ikke ha endret database, og det er derfor ikke nødvendig å angre endringene.
- Det kan være nødvendig å gjøre om oppdateringer av forpliktede transaksjoner, ettersom effekten ikke har nådd databasen.

Immediate Update

- Oppdateringer blir brukt på databasen når de oppstår.
- Trenger å gjøre om oppdateringer av forpliktede transaksjoner etter en feil.
- Må kanskje angre effekten av transaksjoner som ikke var bekreftet på tidspunktet for feil.
- Viktig at loggposter blir skrevet før COMMIT behandling:
  - Write-Ahead-Log (Skriv-frem-loggprotokoll).

Logg/Recovery oppsummert

- Loggen lagres på disken, ikke i minnet. Hvis systemet krasjer, så er den bevart
- Recovery Manager garanterer Atomicity & Durability.
- Checkpointing: En rask måte å begrense loggmengden som må skannes ved gjenoppretting.
- Deferred → vi trenger ikke å angre endringene av en avbrutt transaksjon fordi disse endringene ikke er skrevet til disken.
- Immediate →vi trenger ikke å gjøre om endringene i en forpliktet transaksjon hvis det oppstår en påfølgende krasj, fordi alle disse endringene garantert er skrevet til disken på kommisjonstidspunktet.