**Deel 3 - Snapshots en backups**

**Snapshots**

* Een (read-only) moment in tijd
* Afhankelijk van de bron database
* Bewaar de toestand van je database op het punt in tijd waarop de snapshot
* Snel en niet resource intensief
* Meerdere snapshots mogelijk voor dezelfde bron database

**Transacties**

* Belangrijk bij het werken met database is:
  + Performantie
  + Consistent zijn van de database
* Hoe kunnen we hiervoor zorgen?

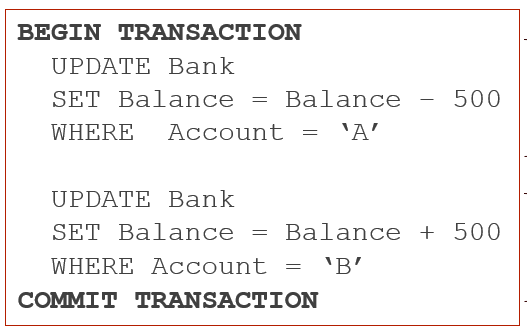
**Performantie**

* SQL Server leest data van schijf naar werkgeheugen (buffer) met pages van 8 KB
* Verwerking in werkgeheugen sneller dan op disk
* Eens page in het werkgeheugen is worden daarna
  + Leesoperaties op die page in het werkgeheugen gedaan
  + Wijzigingen op gegevens in die page ook in het werkgeheugen gedaan
  + Bij wijziging wordt page dan als “Dirty” gemarkeerd
  + Aantal dirty pages mogelijk kleiner dan aantal wijzigingen
  + Hierdoor minder I/O operaties (1 page wegschrijven i.p.v. x-aantal wijzigingen)  
      
    => Betere performantie
* Werken met gegevens in werkgeheugen zorgt voor
  + Betere performantie
  + Groot risico op dataverlies!  
      
     => Evenwicht vinden tussen performantie en betrouwbaarheid data!

**Consistente database mbv transacties**

* Wijzigingen wegschrijven naar permanent opslagmedium
* Gebruik maken van transacties
  + Record(s) die geaffecteerd worden door INSERT/UPDATE/DELETE instructies
  + Worden in hun totaliteit bewaard (=commit)
  + Ofwel niet waarbij eventuele wijzigingen ongedaan gemaakt worden (=rollback)
* Transactie
  + Kan impliciet (insert/update/delete) of expliciet (begin transaction) gestart worden
  + Garandeert ook relationele integriteit

Vb: Overschrijven $500 van rekening A naar B



* Tijdens de transactie kan de server crashen om diverse redenen
  + Power outage
  + Disk failure
* Dit kan ook gebeuren tijdens het wegschrijven van gegevens
* Hoe oplossen?

=> “Transaction log” file gebruiken

**Transaction log**

* Extra bestand dat bij een database hoort (extensie .ldf)
* DBMS schrijft “gebeurtenissen” naar dit bestand
* Gebeurtenissen worden sequentieel (in volgorde van gebeuren) weggeschreven
* **Gebeurtenissen** zijn:
  + Insert van record(s)
  + Update van record(s)
  + Delete van record(s)
* Transaction log file is nodig voor transactie en database recovery
* **Write-ahead logging:**
  + Eerst wijzigingen wegschrijven naar transaction log
  + Daarna wegschrijven naar database
* Wegschrijven naar transaction log is sequentiële write
  + Sneller dan random writes
* Er wordt een entry in de transaction log geschreven voor:
  + Start en einde van een transactie
  + Rollback operaties
  + Wijziging van data, ongeacht hoe deze tot stand kwam
    - Insert
    - Update
    - Delete
  + Wijziging (create, drop, alter) van tabellen en/of indexen

=> Transacties lopen door elkaar

* Elke entry in de transaction log file krijgt een unieke nummer  
   => Dit noemen we de **Log Sequence Number (LSN)** (numerieke waarde die telkens ophoogt)
* Als transacties door elkaar lopen en door elkaar worden weggeschreven
* Hoe weten we tot welke transactie een entry behoort?

=> Start transactie is beginpunt  
=> Elke transactie krijgt een ID

=> Elke (data-gerelateerde) entry krijgt een verwijzing naar die ID

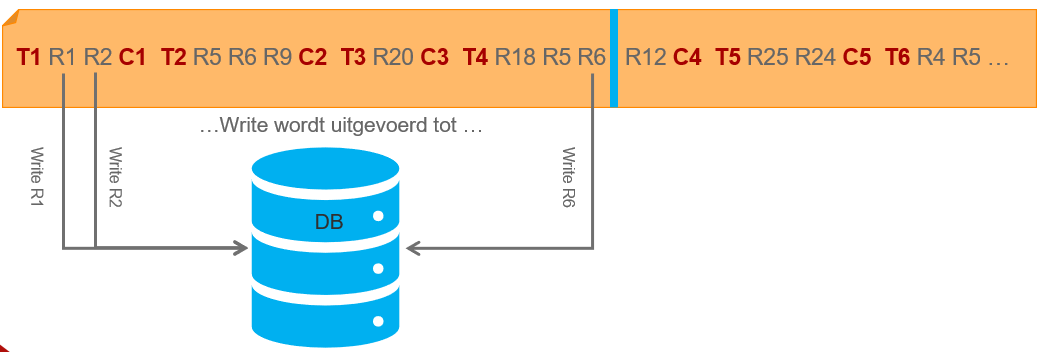
**Durability**

* Ook transaction log file wordt in geheugen geladen
* Writes naar TL worden initieel in geheugen geschreven

=> Risico op dataverlies bij crash

* 2 manieren om hier mee om te gaan
  + **Full transaction durability** (synchroon)
    - Gebruiken indien dataverlies niet tolereerbaar is
  + **Delayed transaction durability** (=asynchroon)
    - Potentieel dataverlies groter dan bij Full transaction durability
    - Gebruiken indien dataverlies minder erg is

**Recovery mechanisme**



* Blauwe lijn markeert een bepaald tijdstip
* Hier beslist SQL Server om alle rijen (R) vanaf vanaf T1 tot blauwe lijn weg te schrijven naar database op disk
* Na de laatste Write, wordt een ijkingspunt toegevoegd aan de transaction log file
  + We noemen dit een Checkpoint
* Het geeft aan vanaf waar we bij de volgende “Write to disk” moeten starten
* Wanneer treedt een Checkpoint op? (4 mogelijkheden)
  + Automatisch op basis van een ingesteld tijdsinterval
  + Indirect
  + Manueel via het T-SQL commando CHECKPOINT
  + Intern bij diverse server operaties zoals back-up en snapshot creatie
* Transaction log file is circulair bestand
  + Vullen start aan het begin van de file
* Op regelmatige tijdstippen gebeurt een truncate van de transaction log file
* **Truncate** is verwijderen van niet gebruikte data
* Truncate <> File shrink
* Truncate gebeurt automatisch, tijdstip afhankelijk van recovery model

**Back-up’s**

**Recovery models**

* Recovery Model van een database
  + Heeft invloed op hoe transaction logging kan gebeuren
  + Bepaalt of een back-up van de de transaction log moet/kan gemaakt worden
  + Bepaalt welke restore mogelijkheden er zijn

Drie modellen

* **Simple recovery model**
  + Meest eenvoudige model
  + Beter niet gebruiken indien data-verlies niet tolereerbaar is
  + Laat toe om database te herstellen tot meest recente back-up
  + Geen point-in-time recovery (herstellen tot op een bepaald tijdstip)
  + Voert bij elk checkpoint automatisch een truncate uit van transaction log tot aan oudste niet gecommitte transactie



* **Bulk-logged recovery model**
  + Te gebruiken bij veelvuldig toepassen van bulk-operaties
  + Geen point-in-time recovery
* **Full recovery model**
  + Point-in-time recovery
  + Transacties met commit worden hersteld
  + Transacties zonder commit worden teruggedraaid

**Types en werking**

* Verschillende back-up types
  + **Full back-up**
  + **Differential back-up**
  + **Transaction log back-up**
* Alle back-up types hangen af van bestaan van full back-up
* Back-ups worden bewaard in een back-up toestel (back-up device)

**Full back-up**

* Traag
* Resource intensief
* Geen leegmaken van de Transaction log file

**Differential back-up**

* De differentiële back-up bewaart alle wijzigingen aan de database sinds de laatste volledige back-up
* Hangt af van het bestaan van de volledige back-up
* Is sneller en kleiner dan een full back-up
* Geen leegmaken van de Transaction log file

**Transaction log**

* De Transaction log back-up bewaart secties van de transaction log file
* Laat toe om terug te zetten tot aan het punt van falen of eender welk punt in de tijd (na volledige- of differentiële back-up).
* Hangt af van het bestaan van de volledige back-up.
* Snel
* De Transaction log back-up verwijdert oude transacties van de transaction log file

**Welk type van back-up wanneer te gebruiken:**

* Alleen volledige back-ups
  + Ok voor kleine databases (backup time < 4 uren)
  + Snelle terugzetting
  + Geen leegmaken van Transaction log file
* Volledige back-ups met differentiële back-ups
  + Ok voor grote databases
  + Snelle differentiële back-up
  + Trage terugzetting (Volledig + Diff)
  + Geen leegmaken van Transaction log file
* Volledige, differentiële en Transaction log back-ups
  + Punt-in-tijd terugzetten
  + Leegmaken van Transaction log file
  + Snelle differentiële- en trans- back-up