Replication

- Bei Replikation handelt es sich um Kopien eines bestimmten Datensatzes zum Zeitpunkt X
- Prozesse, Server oder andere Operationen greifen immer auf das Datenobjekt mit dem Zeitstempel X zu
- Im Idealfall erhält jeder Prozess bei jedem Replikat immer dasselbe Ergebnis

Vorteile von Replikation:

- Mobilität, Ausfallsicherheit, Verlustsicherheit
- Steigerung der Verlässlichkeit
  (Beispiel: Ausfallsicherheit bei Replikats Verlust)
- Schutz vor zerstörten oder gefälschten Daten
  (Beispiel: Bei gleichzeitigem Zugriff auf mehrere Replikate entscheidet das Ergebnis der Mehrheit über die Änderung)
- Steigerung der Leistungsfähigkeit
  (Beispiel: Verteilung von Replikaten auf verschieden Speicherobjekte, um Zugriffszeit zu minimieren)

Consistency

- Bei Consistency kann man auch von einem Vertrag zwischen dem Datenspeicher und den Prozessen sprechen
- Bei einem Read Befehlt wird immer der letzte Write Befehl erwartet
- Je stärker die Konsistenz desto schlechter die Performance
- Je schwächer die Konsistenz desto besser ist die Performance
- Zwei Arten von Konsistenzmodellen
  (Client-zentrierte Konsistenzmodelle, Datenzentrierte Konsistenzmodelle)
- Konsistenzmodelle lassen sich mit der Hilfe von Protokollen implementieren

Verteilungsprotokolle

- Zwei Arten von Protokollarten (Primary-based Protokolle und Replicated-Write Protokolle
- Primary-based
  (Write-Operationen finden immer auf einem Primären Replikat statt)
- Replicated-based
  (Write-Operationen können auf einem beliebigen Replikat durchgeführt werden)
- Verteilt werden die Updates via Multicast oder Unicast
- Zur Verteilung wird das Pull-/Push-Prinzip angewandt
  Push (Server basierte Verteilung)
  Pull (Client basierte Verteilung)
- Quorum-based Protokolle können durch logischen Aufbau schon eine gewisse Fehlertoleranz gewehrleisten