Lernzettel

Anwendungen in Wirtschaftsinformatik/Business Intelligence: Data Processing, ETL, Berichte mit Scala.

Universität:	Technische Universität Berlin
Kurs/Modul:	Programmieren II für Wirtschaftsinformatiker
Erstellungsdatum:	September 6, 2025

Lernzettel: Anwendungen in Wirtschaftsinformatik/Business Intelligence: Data Processing, ETL, Berichte mit Scala

(1) Kontext und Zielsetzung. In BI-Anwendungen werden große Datenmengen aus unterschiedlichen Quellen verarbeitet, transformiert und in Berichte überführt. Scala bietet durch seine funktionale Programmierung, seine Interoperabilität mit Java und sein Ökosystem für Nebenläufigkeit gute Grundlagen für robuste Data-Processing-Pipelines und ETL-Prozesse.

(2) Data Processing mit Scala. - Datenverarbeitung als Pipelineschrittfolge: Eingabe (Raw Data) → Transformation → Aggregation → Ausgabe. - Grundlagen: unveränderliche Collections, map/flatMap/filter, sowie Sequenzen und Streams zur Modellierung von Datenströmen. - Nebenläufigkeit und Parallelität: parallele Sammlungen, Futures, Akka-Modelle für verteilte oder asynchrone Abläufe. - Interoperabilität: einfache Nutzung von Java-Bibliotheken und -Datenstrukturen aus Scala heraus. - Typische Muster: Funktionsorientierte Transformationsketten, Try/Catch-Strategien für Fehlerbehandlung, Options/Either für Fehler- bzw. Null-Behandlung.

(3) ETL-Prozess in Scala. - ETL-Phasen: Extract (Extraktion), Transform (Umwandlung/Reinigung), Load (Laden in Zielsystem). - Beispielhafte Struktur: Datenquellen wie relationale DBs, CSV/Parquet-Dateien oder APIs werden gelesen, bereinigt und in das Data Warehouse transportiert. - Wichtige Konzepte: Datengenauigkeit, Datenqualität, Logging, Fehler-Handling, Idempotenz und Rückverfolgbarkeit (Data Lineage). - Mathematisch formuliert lässt sich der ETL-Prozess oft als Abbildung darstellen:

$$\mathrm{ETL}(D) = \mathrm{Load}\big(\mathrm{Transform}\big(\mathrm{Extract}(D)\big)\big),$$

wobei D die Rohdatenmengen beschreibt. - Typische Transformationsaufgaben: Normalisierung, Typkonvertierung, Join von Datenquellen, Berechnungen und Aggregationen. - Praktische Hinweise: modularisierte Pipelines, Fehlertoleranz, Monitoring und Skalierbarkeit durch Streaming/Batch-Ansätze.

(4) Berichte und Reporting mit Scala. - Berichte dienen der Verteilung aussagekräftiger Kennzahlen an Fachbereiche und Entscheidungsträger. - Exportformate: CSV, JSON, HTML-Berichte, PDF-Templates, oder direkte Anbindungen an BI-Tools. - Reporting-Workflows: Aggregationen nach Dimensionen (Region, Produkt, Zeitraum), Zeitreihen, Drill-Down-Analysen. - Technische Umsetzung: Nutzung von DataFrames/Datasets (z. B. Spark mit Scala), user-defined Functions (UDFs), plattformunabhängige Serialisierung. - Automatisierung: regelmäßige Generierung von Reports, Benachrichtigungen, Logging von Report-Generierungen.

(5) Schlüsselkonzepte und Best Practices. - Funktionen als first-class Bürger: Höherordentliche Funktionen, Closures, immutable Datenstrukturen. - Nebenläufigkeit: efektive Nutzung von Futures, Actors (Akka) für Message-Passing-Concurrency. - Java-Scala-Interoperabilität: reaktive Nutzung vorhandener Java-Bibliotheken, klare Grenzlinien zwischen Java- und Scala-Code. - Fehlerbehandlung und Robustheit: Exceptions, monadische Typen (Option, Either), sinnvolle Fail-Fast-Strategien. - Testing und Qualitätssicherung: Unit-Tests für Transformationslogik, Data-Quality-Checks, End-to-End-Tests der Pipelines.

(6) Beispiel-Szenarien (anwendungsnah). - ETL-Pipeline, die Kundendaten aus einer API, einer CSV-Datei und einer Datenbank zusammenführt, bereinigt und als Parquet-Dateien für das Data Lake speichert. - Berichte, die Verkaufszahlen pro Region wöchentlich aggregieren

und als HTML-Reports an das Management versenden. - Streaming-basierte Verarbeitung von Transaktionen mit Akka, um Echtzeit-Alerts bei Anomalien zu generieren.

(7) Weiterführende Konzepte (Ausblick). - Vertiefung der Konzepte Closures, Traits und Exceptions in Scala. - Fortgeschrittene Concurrency-Techniken und Fehlerbehandlung in verteilten Systemen. - Vertiefung von Akka: Actor-Modell, Mailboxen, Supervision, Remoting.