**Thema 1: Oracle APEX - Datenbanksätze Übersetzung in andere Sprachen mit KI-gestütztes Tool**

**Verbesserungsvorschlag:** **KI-basierte Übersetzung für mehrsprachige Datenbankanwendungen**

**Zielsetzung:**Das Ziel dieser Arbeit ist die Konzeption und Entwicklung einer zentralen Übersetzungsanwendung in Oracle APEX. Diese Anwendung soll eine effiziente und konsistente Verwaltung sowie die KI-gestützte Übersetzung von Datenbankeinträgen aus mehreren Anwendungen ermöglichen. Dabei wird ein besonderer Fokus auf die Automatisierung, Datenintegrität, Nachvollziehbarkeit und Skalierbarkeit gelegt. Die Übersetzungsanwendung soll dazu beitragen, den mehrsprachigen Betrieb von Anwendungen zu erleichtern und gleichzeitig die Anforderungen an Qualität, Sicherheit und Benutzerfreundlichkeit zu erfüllen.

**Kernidee:**

* Anwendung Entwickeln
* Package bauen / API
* Von der daten Entnahme
* Daten Abgabe
* In der Zielanwendung die Erstellung der Übersetzungstabelle / oder Tabellen Erweiterung mit den sprachen
* ~~Metadaten der Anwendung entnehmen; für die aktuellen Versionserfassung sowie Name der Anwendung und APP ID~~

Theorieabschnitt

* Oracle APEX 🡪 Max. 1 Seite
  + Überblick über die Architektur
  + Einsatzmöglichkeiten
* Datenbanken
  + Grundkonzept der Datenbank
  + Datenintegrität
    - Eindeutige Daten Zuordnung
    - Sicherstellung wie alle Spalten und Datentypen im Zielanwendung korrekt befüllt werden
    - Intakte Verknüpfung zwischen Original- und Übersetzungsdaten
    - Validation auf Vollständige Einträge
  + Datenkonsistenz
    - Übersetzungen müssen in den Originaldaten Synchron gehalten werden
    - Sicherstellung, dass die Übersetzungen den ursprünglichen Bedeutung Zusammenhang bleiben
    - Initiale Übersetzungsprüfung
      * Prüfen, ob der Wert in der Zielsprache schon übersetzt ist
    - Aufbereitung der Daten
  + Statusverwaltung
    - Statusfeld zu überwachen verwenden
  + Auditspalten für Überwachung der Übersetzung
* Künstliche Intelligenz in der Übersetzung
  + Arten von KI-gestützten Übersetzungen
  + Funktionsweise der KI-Übersetzungsdienste
  + Welche Herausforderung entsteht bei der KI-Übersetzung
  + Integration von KI-Übersetzungsdienstleistung in die Übersetzungsanwendung
* API-Schnittstelle
  + Grundlagen von APIs
  + API für Datenübertragung
  + Sicherheitsaspekte die bei einer API-Nutzung geschaut werden muss
  + Herausforderungen bei der API-Integration
  + Struktur der APIs
  + API-Monitoring und Wartung
* ~~Revisionsmanagement und Änderungsverfolgung~~
* ~~Mehrsprachigkeit in Anwendungen~~
* ~~Technische und organisatorische Herausforderungen~~

Praxisabschnitt

* Konzept der Anwendung
  + Beschreibung der Übersetzungsanwendung und deren Zielsetzung
  + Datenfluss; Wie genau Daten importiert / Exportiert werden
  + Berechtigungsverwaltung
    - Welche Benutzer hat Zugriff auf die Anwendung
  + Wann soll eine Übersetzung stattfinden
* Datenmodell
  + Datenbankstruktur der Zentrale Übersetzungsanwendungen
    - Darstellung des Datenmodells
    - Originaleinträge, Übersetzungen, Benutzeranpassungen, Änderungsprotokolle
    - Statusfeld
* Implementierung der API-Schnittstellen
  + Darstellung der APIs
    - Welche Verbindungen müssen entstehen
  + Eingehende API
    - Was muss alles aus der eingehende API kommen
    - ~~Unterstützung beim Daten Eingang verschiedener Datenformate;(zb. XML, CSV, Oracle Text?)~~
  + Übersetzung API
    - Was muss alles an die API weitergegeben werden
    - Was liefert die API zurück
  + Ausgehende API
    - Was muss der API mitgegeben werden, das die Übersetzten Einträge im Zieldatenbank hinzugefügt wird
    - ~~Unterstützung verschiedene Datenformate;(zb. XML, CSV, Oracle Text?)~~
    - ~~Möglichkeit, Exportfilter anzuwenden (zb. Nur eine bestimmte Sprache)~~
    - Erstellung der Zieltabellen für die Übersetzungstabelle
  + ~~WebHook – Integration~~
    - ~~Sobald neue Einträge im Zielsystem freigegeben wurde als Benachrichtigung im Zielsystem anzeigen~~
* Kosten durch die API-Aufrufe
  + Grundlagen der API-Aufrufe
  + Kosten pro Übersetzungsanfrage
* Benutzeroberfläche
  + Übersichtseiten
    - Dashboard
      * wie viele Einträge importiert sind von welcher Anwendung
    - Liste zu übersetzende Einträge
    - Sprachen welche übersetzt werden kann
    - ~~Änderungsübersicht~~
      * ~~wer wann was geändert hat~~
      * ~~Wie viele Einträge wurden geändert~~
* Automatisierungen und Statusprüfung
  + Initiale Übersetzung
    - Kennzeichnung von Einträgen die Übersetzt worden sind und die nicht übersetzt worden, da Sie „bereits“ übersetzt sind
* Fehlerbehandlung und Sicherheit
  + Fehlerbehandlung
    - Protokollierung von Fehlern
  + Sicherheit
    - Zugriffsbeschränkungen für APIs und Benutzer
    - Datenverschlüsselung bei sensiblen Datenübertragungen
  + ~~Qualitätssicherung~~
    - ~~Zu lange Zeichenketten oder Platzhalter~~
    - ~~Berichte welche Sprache am häufigsten manuell angepasst wird~~
* Tests und Evaluierungen
  + Testszenarien
    - Import aller Datensätzen
    - Übersetzung der Datensätze mit Berücksichtigung auf Übersetzungsbedarf
    - API-Verwendung im Zielsystem
  + ~~Performance-Tests~~
    - ~~Prüfung der Skalierbarkeit bei großen Datenmengen~~
    - ~~Latenzzeiten der Übersetzungs-API~~
  + Benutzerfeedback (Formular)
    - Evaluierung der Benutzeroberfläche
    - Evaluierung der Funktionalitäten
* Fazit
  + Welche Funktionalitäten haben, gut funktioniert
  + Welche Herausforderung sind aufgetaucht
  + Welche Optimierungen und Erweiterungen sollten vorgenommen werden

**Thema 2: Oracle APEX - Datenbanksätze Bereinigung mit KI-gestütztes Tool**

**Verbesserung: Erhöhung der Datenqualität mittels KI**

**Zielsetzung:**

Das Ziel dieser Arbeit ist die Entwicklung einer zentralisierten Bereinigungsplattform, die sowohl automatisierte als auch manuelle Datenbereinigungsprozesse unterstützt. Diese Plattform wird in Oracle APEX realisiert und soll Datenbankeinträge aus verschiedenen Anwendungen Vereinheitlichen, analysieren und optimieren. Durch den Einsatz moderner Technologien, wie KI-gestützter Datenkorrektur und API-Schnittstellen, wird die Plattform darauf ausgelegt, Datenintegrität, Konsistenz und Effizienz zu gewährleisten

Kernidee:

* Zentralisierte Bereinigungsplattform
* Automatisierte und manuelle Bereinigung
* ~~Metadaten der Anwendung entnehmen; für die aktuellen Versionserfassung sowie Name der Anwendung und APP ID~~
* API-Schnittstelle

Theorieabschnitt

* Oracle APEX
  + Überblick über die Architektur
  + Einsatzmöglichkeiten
* Datenbanken
  + Eindeutige Datenzuordnung
  + Datenintegrität
  + Datenkonsistenz
  + Statusverwaltung
  + Auditspalten
* Datenbereinigung
  + Duplicate Detection
  + Rechtschreibprüfung
  + Grammatikprüfung
  + Datenklassifikation
* Künstliche Intelligenz in der Datenkorrektur
  + Arten von KI-Modellen
  + Funktionsweise der KI-Datenkorrektur
  + Herausforderungen der KI-Datenkorrektur
  + Integration von KI-Diensten
* API-Schnittstelle
  + Grundlagen von APIs
  + API für Datenübertragung
  + Sicherheitsaspekte die bei einer API-Nutzung geschaut werden muss
  + Herausforderungen bei der API-Integration
  + Struktur der APIs
  + API-Monitoring und Wartung
* ~~Revisionsmanagement und Änderungsverfolgung~~
* ~~Mehrsprachigkeit in Anwendungen~~
* ~~Technische und organisatorische Herausforderungen~~

Praxisabschnitt

* Konzept der Anwendung
  + Beschreibung der Übersetzungsanwendung und deren Zielsetzung
  + Datenfluss; Wie genau Daten importiert / Exportiert werden
  + Berechtigungsverwaltung
    - Welche Benutzer hat Zugriff auf die Anwendung
  + ~~Mehrsprachige Benutzeroberfläche~~
* Datenmodell
  + Datenbankstruktur der Zentrale Übersetzungsanwendungen
    - Darstellung des Datenmodells
    - Originaleinträge, bereinigten Daten, Vorschläge, Benutzeranpassungen, Änderungsprotokolle
    - Statusfeld
* Implementierung der API-Schnittstellen
  + Darstellung der APIs
    - Welche Verbindungen müssen entstehen
  + Eingehende API
    - Was muss alles aus der eingehende API kommen
    - ~~Unterstützung beim Daten Eingang verschiedener Datenformate;(zb. XML, CSV, Oracle Text?)~~
  + Bereinigung API
    - Was muss alles an die API weitergegeben werden
    - Was liefert die API zurück
  + Ausgehende API
    - Was muss der API mitgegeben werden, das die Übersetzten Einträge im Zieldatenbank hinzugefügt wird
    - ~~Unterstützung verschiedene Datenformate;(zb. XML, CSV, Oracle Text?)~~
    - Möglichkeit, Exportfilter anzuwenden (zb. Nur eine bestimmte Sprache)
    - Erstellung der Zieltabellen für die Übersetzungstabelle
  + ~~WebHook – Integration~~
    - ~~Sobald neue Einträge im Zielsystem freigegeben wurde als Benachrichtigung im Zielsystem anzeigen~~
* Kosten durch die API-Aufrufe
  + Grundlagen der API-Aufrufe
  + Kosten pro Übersetzungsanfrage
* Benutzeroberfläche
  + Übersichtseiten
    - Liste zu bereinigten Einträgen
    - Form der Datenbereinigung
    - Änderungsübersicht
      * wer wann was geändert hat
      * Wie viele Einträge wurden geändert
    - Dashboard
      * wie viele Einträge importiert sind von welcher Anwendung
* Automatisierungen und Statusprüfung
  + Initiale „Bereinigung“
    - Kennzeichnung von Einträgen, die bereinigt werden können und zeigen welche der Einträge in Ordnung sind, auch die Empfehlung was für eine ob es im richtigen Tabelle aufgehoben ist.
* Fehlerbehandlung und Sicherheit
  + Fehlerbehandlung
    - Protokollierung von Fehlern
  + Sicherheit
    - Zugriffsbeschränkungen für APIs und Benutzer
    - Datenverschlüsselung bei sensiblen Datenübertragungen
* Tests und Evaluierungen
  + Testszenarien
    - Import aller Datensätzen
    - Bereinigung der Datensätze mit Berücksichtigung auf Bereinigungsbedarf
    - API-Verwendung im Zielsystem
  + ~~Performance-Tests~~
    - ~~Prüfung der Skalierbarkeit bei großen Datenmengen~~
    - ~~Latenzzeiten der Bereinigungs-API~~
  + Benutzerfeedback
    - Evaluierung der Benutzeroberfläche
    - Evaluierung der Funktionalitäten
* Fazit
  + Welche Funktionalitäten haben, gut funktioniert
  + Welche Herausforderung sind aufgetaucht
  + Welche Optimierungen und Erweiterungen sollten vorgenommen werden

~~Thema 3: Oracle APEX: Qualitätsmanagement mit KI-gestütztes Tool~~

~~Zielsetzung:~~

~~Die Zielsetzung dieser Arbeit ist die Entwicklung und Integration eines automatisierten Codeprüfungs- und Validierungssystems für Skripte (z. B. SQL und PL/SQL) in eine CI/CD-Pipeline mithilfe von Jenkins. Dieses System soll mithilfe eines KI-Modells die Skripte hinsichtlich ihrer Qualität, Effizienz und Fehlerfreiheit analysieren. Dabei wird das KI-Modell Fehler und Optimierungspotenziale identifizieren und direktes Feedback an Entwickler geben. Wenn das System als „Unstable“ oder „Failing“ in Jenkins erkannt wird, soll eine Datei mit konkreten Korrekturvorschlägen generiert werden.~~

~~Kernidee:~~

* ~~Codeprüfung als Build-Schritt in Jenkins: Die Codeanalyse wird in den Jenkins-Build-Prozess integriert, sodass jede Änderung im Code durch das KI-Modell überprüft wird.~~
* ~~Codeanalyse: Das KI-Modell analysiert den Code hinsichtlich potenzieller Fehler, Code-Smells, Sicherheitslücken und Optimierungsmöglichkeiten.~~
* ~~Feedbackmechanismus:~~
  + ~~Success: Der Code ist fehlerfrei und entspricht den Qualitätsstandards.~~
  + ~~Unstable: Warnung, dass der Code potenzielle Probleme aufweist, aber der Build weiterhin erfolgreich ist.~~
  + ~~Failure: Kritischer Fehler im Code, der den Build-Prozess stoppt.~~
* ~~Automatische Korrekturvorschläge: Bei "Unstable" oder "Failure" liefert das System automatisch Korrekturvorschläge und Optimierungstipps, die den Entwicklern helfen, den Code zu verbessern.~~

~~Theorieabschnitt:~~

~~Einführung in CI/CD~~

* ~~Definition von CI/CD: CI (Continuous Integration) und CD (Continuous Deployment) sind Praktiken der Softwareentwicklung, bei denen Änderungen am Code kontinuierlich integriert, getestet und bereitgestellt werden. Sie fördern eine schnelle und zuverlässige Bereitstellung von Software.~~
* ~~Vorteile von CI/CD:~~
  + ~~Erhöhte Softwarequalität durch kontinuierliche Tests.~~
  + ~~Schnelleres Feedback für Entwickler.~~
  + ~~Automatisierung der Builds und Deployments zur Reduzierung menschlicher Fehler.~~
  + ~~Effizientere Nutzung von Entwicklungsressourcen.~~

~~Jenkins als Automatisierungsserver~~

* ~~Jenkins ist ein Open-Source-Automatisierungsserver, der in der CI/CD-Pipeline weit verbreitet ist. Jenkins übernimmt die Aufgabe, Code automatisch zu integrieren, zu testen und bereitzustellen. Es bietet zahlreiche Plugins zur Integration von Tools wie SonarQube, DeepCode und CodeGuru, die für die Codeanalyse und das Feedbackmanagement genutzt werden können.~~

~~Codequalität und Codeprüfung~~

* ~~Codequalität: Codequalität ist entscheidend für die Wartbarkeit, Lesbarkeit und Performance einer Softwareanwendung. Automatisierte Codeprüfungen helfen, Probleme frühzeitig zu erkennen und zu beheben.~~
* ~~Automatisierte Codeanalyse: Tools wie SonarQube, DeepCode und CodeGuru analysieren den Code auf verschiedene Aspekte wie Fehler, Code-Smells und Sicherheitslücken. Diese Tools nutzen teilweise KI und maschinelles Lernen, um genauere Analysen und Vorschläge zur Verbesserung zu liefern.~~

~~SQL/PLSQL-Analyse und Best Practices~~

* ~~SQL-Optimierung: Best Practices für die Erstellung effizienter SQL-Abfragen, z. B. die Nutzung von Indexen, Abfrageoptimierung und Vermeidung unnötiger Joins.~~
* ~~PLSQL Best Practices: Richtlinien für die Erstellung von performantem und sicherem PL/SQL-Code, z. B. die Handhabung von Exceptions, der Einsatz von Bulk-Operationen und das Vermeiden von Kursoren.~~
* ~~Sicherheitsaspekte: SQL-Injection-Vermeidung und Best Practices für die sichere Verarbeitung von Benutzereingaben.~~

~~Integration von KI in CI/CD-Pipelines~~

* ~~KI-gestützte Tools: Tools wie DeepCode, SonarQube, CodeGuru und Codacy bieten KI-basierte Funktionen, die eine tiefergehende Codeanalyse ermöglichen. Sie helfen, nicht nur einfache Fehler zu finden, sondern auch potenzielle Sicherheitslücken und Optimierungspotenziale zu identifizieren. Die Integration dieser Tools in eine Jenkins-Pipeline ermöglicht eine automatisierte Codeprüfung, die ständig aktualisiert und verbessert wird.~~

~~Praxisabschnitt:~~

* ~~Zielsetzung und Anforderungen der Praxisanwendung~~
  + ~~Das Ziel der Praxisanwendung ist die Erstellung eines Systems, das es ermöglicht, SQL- und PL/SQL-Skripte automatisch auf Qualität und Fehler zu überprüfen. Das System soll Fehler erkennen, Optimierungspotenziale aufzeigen und Korrekturvorschläge generieren, wenn der Code in Jenkins als „Unstable“ oder „Failing“ eingestuft wird.~~
* ~~Design der Automatisierten CI/CD-Pipeline mit KI~~
  + ~~Pipeline-Architektur Die CI/CD-Pipeline besteht aus mehreren Schritten:~~
    - ~~Code-Commit: Der Entwickler committet Code in das Versionskontrollsystem (z. B. Git).~~
    - ~~Code-Analyse: Beim Build-Schritt wird der Code von einem KI-gestützten Tool wie DeepCode oder CodeGuru analysiert.~~
    - ~~Build-Status: Jenkins überprüft den Build-Status und kategorisiert ihn als „Success“, „Unstable“ oder „Failure“.~~
    - ~~Feedback: Bei Problemen wird ein Bericht mit konkreten Korrekturvorschlägen erstellt und an den Entwickler weitergegeben.~~
* ~~Implementierung der KI-Codeprüfung und Integration von Tools~~
  + ~~Auswahl des KI-Tools: Die Integration von DeepCode, SonarQube oder CodeGuru in die Jenkins-Pipeline wird durchgeführt, um den Code auf Fehler und Optimierungspotenziale zu überprüfen.~~
  + ~~Automatische Korrekturvorschläge: Wenn das System als „Unstable“ oder „Failing“ eingestuft wird, werden automatisch Verbesserungsvorschläge in einer Datei generiert und dem Entwickler zur Verfügung gestellt.~~
* ~~Ergebnisse und Auswertung~~
  + ~~Testen der Pipeline: Es wird eine Testpipeline erstellt, die Code aus verschiedenen Quellen zieht, analysiert und Ergebnisse zurückmeldet. Die Ergebnisse werden ausgewertet, um zu prüfen, wie genau und zuverlässig die KI-gestützte Analyse funktioniert und welche Korrekturvorschläge sie bietet.~~
  + ~~Vergleich von Erfolg und Misserfolg: Eine Analyse der Performance des Systems und die Bewertung der KI-gestützten Vorschläge im Vergleich zu manuellen Code-Reviews.~~