# Datenbanken Advanced
Projektarbeit

## Projektarbeit – Auftragsverwaltung

**Ausgangslage Auftrag**

Das im Unterricht erworbene Wissen bzgl. Datenbankentwicklung soll im Rahmen dieser Projektarbeit weiter vertieft werden.

Es soll für die Muster AG eine Datenbankapplikation entwickelt werden, in der folgende Daten verwaltet werden können:

- Kunden und Adressen
- Artikel und Artikelgruppen
- Aufträge und Auftragspositionen

Die Theorie für die Umsetzung dieser Anforderungen, wird im Unterricht laufend erarbeitet. Es besteht jedoch nicht zwingend eine Korrelation der Reihenfolge der Anforderungen und der im Unterricht vermittelten Inhalte.

Grundlagen wurden bereits im Modul «Datenbanken» erarbeitet (Projektarbeit ERP-Lösung sowie Workshop bzgl. ADO.NET).

Erstellen Sie über das gesamte Projekt eine Dokumentation in welcher Sie Ihr Vorgehen sowie Ihre Ergebnisse festhalten. Verwenden Sie dazu konzeptionelle Elemente bzgl. Softwareengineering und Design wie: ERM, UseCase-Diagramme, Komponentendiagramme, Klassendiagramme, Sequenzdiagramme usw.

Diese Projektarbeit wird als Testat in Form einer Gruppenarbeit (2er-Gruppen) geführt.
Die Dokumentation sowie die konkrete Umsetzung bilden die Basis für das Bestehen des Testats und in der Folge des Moduls.

***Funktionale Anforderungen***

**Die Muster AG wünscht für Ihre Datenbankapplikation die Umsetzung folgender funktionaler Anforderungen:**

*Auftragsverwaltung*

Erstellen Sie eine Applikation, welche die Verwaltung der eingangs erwähnten Daten zulässt. Entwickeln Sie hierfür eine eigenständige Windows-Desktop-Applikation. Für diese Applikation soll folgender Technologie-Stack Verwendung finden:

- Windows Desktop Application mit C#
- Presentation-Layer wahlweise mit Windows Forms oder WPF
- Data-Layer mit Entity Framework und CodeFirst

Die Applikation soll folgende UseCases ermöglichen:

- Kunden verwalten (suchen, erstellen, bearbeiten, löschen)
- Artikelgruppen verwalten (suchen, erstellen, bearbeiten, löschen)
- Artikel verwalten (suchen, erstellen, bearbeiten, löschen)
- Aufträge verwalten (suchen, erstellen, bearbeiten, löschen)

Für die einzelnen Entitäten sollen im Minimum folgende Attribute verwaltet werden können:

- Kunden und Adressen: Name, Strasse, PLZ, Ort
- Artikelgruppe: Name, übergeordnete Artikelgruppe (Hierarchie)
- Artikel: Artikelnummer, Bezeichnung, Preis, Artikelgruppe
- Aufträge und Positionen: Auftragsnummer, Datum, Kunde; Positionen mit Nummer, Artikel, Anzahl

Beachten Sie die Normalisierungsregeln, die Sie im Modul Datenbanken erarbeitet haben.

Als Datenbanksystem verwenden Sie MS SQL Server.

*Hierarchische Struktur der Artikelgruppen*

Die Artikelgruppen können hierarchisch gegliedert werden.

Entwickeln Sie eine rekursive CTE, um die Artikelgruppe sowie deren Hierarchie ausgeben zu können. Diese soll anschliessend in Form einer TreeView in der Applikation zur Verfügung gestellt werden.

*Jahresvergleich*[1]

Entwickeln Sie eine Abfrage, die pro Quartal für die vergangenen 3 Jahre folgende Zahlen (Kategorien) auflistet:

- Anzahl Aufträge
- Anzahl verwaltete Artikel
- Durchschnittliche Anzahl Artikel pro Auftrag
- Umsatz pro Kunde
- Gesamtumsatz

Verwenden Sie für diese Abfrage zwingend Window Functions.
Ggf. müssen Sie Ihr Datenmodell erweitern, damit die erforderlichen Daten für die Abfrage auch zur Verfügung stehen (z.B. Verwaltung der Daten auf der Zeitachse).

Die Struktur der Darstellung soll folgendermassen aussehen (wobei die Jahreszahlen und die Quartale den aktuellen Zeitpunkt repräsentieren sollen – d.h. aktuelle Quartal/Jahr sowie die vergangenen drei Vergleichsquartale):

| Kategorie | YOY | Q4 2020 | Q3 2020 | Q2 2020 | Q1 2020 | … |
|---|---|---|---|---|---|---|
| … | … | … | … | … | … | … |
| … | … | … | … | … | … | … |

Die Ausgabe soll ein einem GridView erfolgen.

Für die Umsetzung dieser Darstellung kann es hilfreich sein, wenn Sie sich mit den SQL-Server-Operatoren PIVOT und UNPIVOT auseinandersetzen. Da diese Operatoren nicht aktiv im Unterricht behandelt werden, müssen Sie sich selbstständig mit diesen auseinandersetzen.

[1] https://de.wikipedia.org/wiki/Jahresvergleich

*Nachvollziehbare Adressänderungen*

Erstellen Sie eine Abfrage, die alle Fakturen mit folgenden Feldern auflistet:

- Kundennummer
- Name
- Strasse
- PLZ
- Ort
- Land
- Rechnungsdatum
- Rechnungsnummer
- Rechnungsbetrag netto
- Rechnungsbetrag brutto

Die Ausgabe soll ein einem GridView erfolgen. Der Benutzer soll die Möglichkeit haben, Filter zu setzen, die Sie in Ihrer Abfrage berücksichtigen.

Die Datenbank erlaubt es, die Adresse eines Kunden jederzeit zu ändern (z.B. wenn der Kunde umzieht oder den Firmennamen ändert usw.). Nun müssen Sie dafür sorgen, dass obige Abfrage für jede Rechnung die am Rechnungsdatum gültige Adresse ausgibt.

Folgendes Szenario soll diese Anforderung verdeutlichen:
Der Kunde Arpanet AG erhält am 31.03.2017 eine Rechnung über 1205.—
Am 02.04.2017 wechselt der Kunde seinen Namen zu Isernet AG. Am 30.04.2017 erhält er eine weitere Rechnung über 340.—

Wenn nun am 01.05.2017 Ihre Abfrage ausgeführt wird, soll folgendes Resultat erscheinen (das Beispiel listet aus Platzgründen nicht alle erforderlichen Spalten auf):

| KdNr | Name | Datum | Nr | Netto | Brutto |
|---|---|---|---|---|---|
| 1234 | Arpanet AG | 31.03.2017 | 2041 | 1205.00 | 1301.40 |
| 123 | Isernet AG | 30.04.2017 | 2369 | 340.00 | 367.20 |

Verwenden Sie für diese Anforderung eine temporale Datenstruktur.

| | |
|---|---|
| **Lieferergebnisse** | **Es muss folgendes abgeliefert werden:**<br><br>• Dokumentation (PDF) über das Vorgehen sowie der Ergebnisse. Verwenden Sie dazu konzeptionelle Elemente bzgl. Softwareengineering und Design wie: ERM, UseCase-Diagramme, Komponentendiagramme, Klassendiagramme, Sequenzdiagramme usw.<br>• Ausführbare Windows-Applikation sowie kompletter Sourcecode inkl. den DB-Migrations. Die Dokumentation muss einen Abschnitt beinhalten, wie die Applikation in Betrieb genommen werden kann – inkl. Hinweise, wie die DB mit den Migrations erstellt werden kann.<br>• Die Migrations müssen zwingend geeignete Testdaten erstellen um die funktionalen Anforderungen testen zu können.<br>• Kurzes Arbeitsjournal (PDF). Hier soll ersichtlich sein, welcher Gruppenteilnehmer welche Aspekte erarbeitet hat sowie wieviel Zeit aufgewendet wurde.<br><br>Das Testat gilt nur dann als bestanden, wenn alle Lieferergebnisse erfüllt sind. Insbesondere müssen zwingend geeignete Testdaten vorhanden sein, um die Ergebnisse prüfen zu können. Zudem muss der Abgabetermin eingehalten werden (siehe Moodle).<br><br>Es werden keine gleichen Lösungen akzeptiert. Werden Plagiate festgestellt (auch nur auszugsweise), gilt das Testat für alle beteiligten Gruppen als nicht bestanden (auch für jene Gruppe, die die Vorlage für das Plagiat geliefert hat).<br><br>Das erfolgreiche Bestehen dieses Testats ist eine Voraussetzung, um das Modul zu bestehen. |