Internet Buchhandlung

Gruppe A : Aysel Aydogan, Atefeh Aghababaei

Data Engineering 16 Oct. 2025

1.1 Projektbeschreibung

Das Ziel des Projekts ist es, eine Datenbank für eine Online-Buchhandlung zu entwickeln.
In dieser Datenbank sollen Informationen über Bücher, Bestellungen, Kunden und Rezensionen gespeichert werden.
Das System soll zeigen, wie man Daten richtig strukturiert, damit sie einfach, sicher und ohne doppelte Einträge verwaltet werden können (Business-DB).

Am Ende soll mit der Datenbank z. B. herausgefunden werden können:

- welche Bücher verkauft wurden
- welche Kunden Bestellungen gemacht haben
- und welche Rezensionen zu einem Buch existieren

So entsteht ein funktionierendes Grundmodell einer Online-Buchhandlung, das später erweitert oder für Analysen genutzt werden kann (DWH).

1.2 Business DB – Struktur und Design

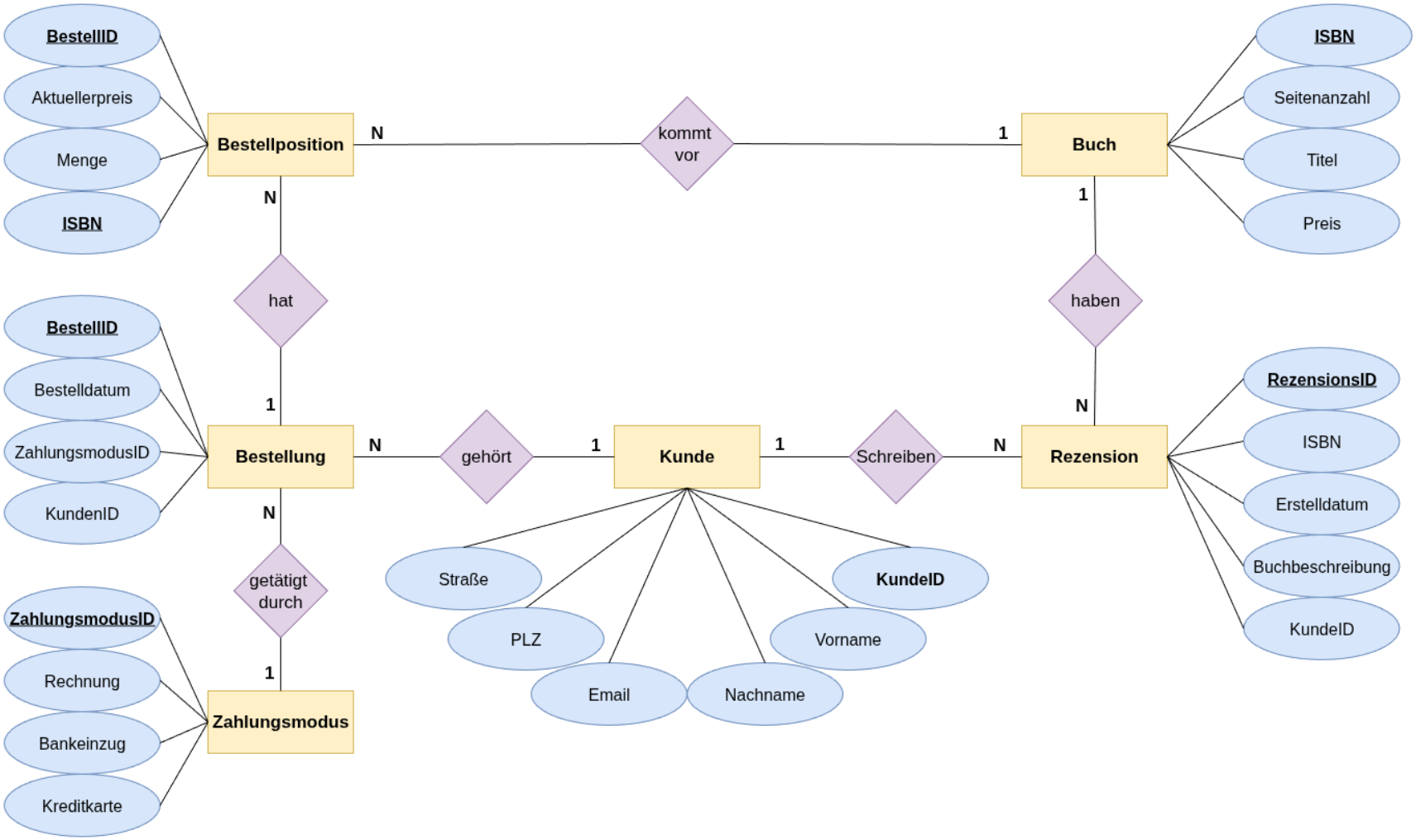
1.2.1 ER-Diagramm

Haupttabellen (Entitäten):

- **Buch** (ISBN PK, Titel, Seitenanzahl, Preis)
- **Kunde** (KundeID PK, Vorname, Nachname, Email, Straße, PLZ)
- **Bestellung** (BestellID PK, Bestelldatum, KundeID FK, ZahlungsmodusID FK)
- **Bestellposition** (BestellID PK FK, ISBN PK FK, Menge, Aktuellerpreis)
- **Rezension** (RezensionID PK, KundeID FK, Beschreibung, Erstelldatum, ISBN FK)
- **Zahlungsmodus** (ZahlungsmodusID PK, Rechnung, Bankeinzug, Kreditkarte)

Beziehungen:

- **Buch *(1)–(N)* Rezension:**
  → Ein Buch kann mehrere Rezensionen haben, eine Rezension gehört zu genau einem Buch
- **Kunde *(1)–(N)* Bestellung:**
  → Ein Kunde kann mehrere Bestellungen tätigen, jede Bestellung gehört zu einem Kunden.
- **Bestellung *(N)–(M)* Buch:**
  → Eine Bestellung kann mehrere Bücher enthalten, und ein Buch kann in mehreren Bestellungen vorkommen.
  → Diese Beziehung wird über die Zwischentabelle Bestellposition realisiert
- **Bestellung *(1)–(N)* Bestellposition:**
  → Eine Bestellung hat viele Bestellpositionen, eine Bestellposition gehört genau zu einer Bestellung.
- **Buch *(1)–(N)* Bestellposition:**
  → Eine Bestellposition bezieht sich auf genau ein Buch, aber ein Buch kann in vielen Bestellpositionen vorkommen
- **Kunde *(1)–(N)* Rezension:**
  → Ein Kunde kann mehrere Rezensionen schreiben (z. B. zu verschiedenen Büchern), Jede Rezension gehört zu genau einem Kunden
- **Zahlungsmodus *(1)–(N)* Bestellung:**
  → Ein Zahlungsmodus kann in vielen Bestellungen vorkommen, eine Bestellung nutzt einen bestimmten Zahlungsmodus


BestellID
Aktuellerpreis
Menge
ISBN
Bestellposition
N
kommt vor
1
Buch
ISBN
Seitenanzahl
Titel
Preis
N
1
hat
haben
BestellID
Bestelldatum
ZahlungsmodusID
KundenID
1
N
Bestellung
N
gehört
1
Kunde
1
Schreiben
N
Rezension
RezensionsID
ISBN
Erstelldatum
Buchbeschreibung
KundeID
N
getätigt durch
Straße
PLZ
Email
Nachname
Vorname
KundeID
ZahlungsmodusID
Rechnung
Bankeinzug
Kreditkarte
1
Zahlungsmodus

1.2.2 Objektbeschreibung

Buch: Enthält Informationen zu den angebotenen Büchern

ISBN

Typ: char(13)

Länge: (13)

Wertebereich: 0-9

Anzahl Wiederholung: 0

Definiertheit: 100%

Identifizierend: Ja

Seitenanzahl

Typ: Integer

Länge: (5)

Anzahl Wiederholung: 0

Definiertheit: 100%

Identifizierend: nein

Titel

Typ: Nvarchar(50)

Länge: (50)

Anzahl Wiederholung: 0

Definiertheit: 100%

Identifizierend: nein

Preis

Typ: Decimal (6,2)

Länge: (6)

Anzahl Wiederholung: 0

Definiertheit: 100%

Identifizierend: nein

Rezension: Enthält Kundenbewertungen und Beschreibungen zu einem Buch

RezensionID

Typ: Integer

Länge: (10)

Wertebereich: 1–9999999999

Anzahl Wiederholung: 0

Definiertheit: 100%

Identifizierend: Ja

IDBN

Typ: char(13)

Länge: (13)

Anzahl Wiederholung: 0

Definiertheit: 100%

Identifizierend: nein

Erstelldatum

Typ: date

Länge: TT.MM.JJJJ

Anzahl Wiederholung: 0

Buchbeschreibung

Typ: Nvarchar

Länge: (2000)

Anzahl Wiederholung: 0

Definiertheit: 100%

Identifizierend: nein

Definiertheit: 100%

Identifizierend: nein

KundeID
Typ: Integer

Länge: (10)

Anzahl Wiederholung: 0

Definiertheit: 100%

Identifizierend: nein

Kunde: Enthält Informationen über registrierte Kunden

KundeID

Typ: Integer

Länge: (10)

Wertebereich: 1–9999999999

Anzahl Wiederholung: 0

Definiertheit: 100%

Identifizierend: Ja

Vorname

Typ: Nvarchar(50)

Länge: (50)

Anzahl Wiederholung: 0

Definiertheit: 100%

Identifizierend: nein

Nachname

Typ: Nvarchar(50)

Länge: (50)

Anzahl Wiederholung: 0

Definiertheit: 100%

Identifizierend: nein

Email

Typ: Nvarchar(100)

Länge: (100)

Anzahl Wiederholung: 0

Definiertheit: 100%

Identifizierend: nein

Straße

Typ: Nvarchar(100)

Länge: (100)

Anzahl Wiederholung: 0

Definiertheit: 100%

Identifizierend: nein

PLZ

Typ: varchar(10)

Länge: (10)

Anzahl Wiederholung: 0

Definiertheit: 100%

Identifizierend: nein

Bestellung: Enthält Informationen zu Kundenbestellungen, inkl. Bestelldatum und Zahlungsmodus

BestellID

Typ: Integer

Länge: (10)

Wertebereich: 1–9999999999

Anzahl Wiederholung: 0

Definiertheit: 100%

Identifizierend: Ja

Bestelldatum

Typ: date

Länge: TT.MM.JJJJ

Anzahl Wiederholung: 0

Definiertheit: 100%

Identifizierend: nein

ZahlunsmodusID

Typ: Integer

Länge: (10)

Anzahl Wiederholung: 0

Definiertheit: 100%

Identifizierend: nein

KundenID

Typ: Integer

Länge: (10)

Anzahl Wiederholung: 0

Definiertheit: 100%

Identifizierend: nein

Bestellposition: Enthält Details zu jedem Buch in einer Bestellung (Stückzahl und zum Bestellzeitpunkt angewendeter Preis

BestellID

Typ: Integer

Länge: (10)

Wertebereich: 1–9999999999

Anzahl Wiederholung: 0

Definiertheit: 100%

Identifizierend: Ja

ISBN

Typ: char(13)

Länge: (13)

Wertebereich: 0-9

Anzahl Wiederholung: 0

Definiertheit: 100%

Identifizierend: Ja

Aktuellerpreis

Typ: decimal(6,2)

Länge: (6)

Anzahl Wiederholung: 0

Definiertheit: 100%

Identifizierend: nein

Menge

Typ: Integer

Länge: (5)

Anzahl Wiederholung: 0

Definiertheit: 100%

Identifizierend: nein

Zahlungsmodus: Enthält Information zur Zahlungsart (Rechnung, Bankeinzug, Kreditkarte)

ZahlungsmodusID

Typ: Integer

Länge: (10)

Wertebereich: 1–9999999999

Anzahl Wiederholung: 0

Definiertheit: 100%

Identifizierend: Ja

Rechnung

Typ: Bit

Länge: –

Anzahl Wiederholung: 0

Definiertheit: 100%

Identifizierend: nein

Bankeinzug

Typ :Bit

Länge: –

Anzahl Wiederholung: 0

Definiertheit: 100%

Identifizierend: nein

Kreditkarte

Typ:Bit

Länge: –

Anzahl Wiederholung: 0

Definiertheit: 100%

Identifizierend: nein

1.2.3 Relationsmodell

Entität	Beschreibung	Attribute	Primärschlüssel	Fremdschlüssel
Buch	Enthält Informationen zu angebotenen Büchern	ISBN, Titel, Seitenanzahl, Preis	ISBN	
Rezension	Enthält Kundenbewertungen und Beschreibungen zu einem Buch	RezensionID, Beschreibung, Erstelldatum, KundenID, ISBN	RezensionID	ISBN → Buch KundeID→ Kunde
Kunde	Enthält Informationen über registrierte Kunden	KundenID, Vorname, Nachname,Email, Straße, PLZ	KundenID	–
Bestellung	Enthält Informationen über getätigte Bestellungen	BestellID, Bestelldatum, ZahlungsmodusID, KundenID	BestellID	KundeID → Kunde ZahlunsmodusID → Zahlunsmodus
Bestellposition	Verbindung zwischen Bestellung und Buch	BestellID, ISBN, Menge,Preis	BestellID, ISBN	BestellID → Bestellung, ISBN → Buch
Zahlungsmodus	Enthält Information zur Zahlungsart	ZahlunsmodusID, Rechnung, Bankeinzug, Kreditkarte	ZahlungsmodusID	–

1.2.4 NF

1. Normalform (1NF) – Atomarität

- Alle Attribute sind atomar (z. B. Vorname, Nachname, Email, ISBN, Preis).
- Jede Spalte enthält nur einen einzelnen, unteilbaren Wert.
  Keine Listen, keine Wiederholungsgruppen.

Ergebnis: Alle Tabellen erfüllen die 1. Normalform.

2. Normalform (2NF) – keine Partialabhängigkeiten

- Bestellposition hat zusammengesetzten Primärschlüssel (BestellID, ISBN).
  → Menge und Aktuellerpreis (Nicht-Schlüssel-Attribut) hängen vollständig von beiden Primärschlüssel ab (nicht nur von einem).

Alle anderen Tabellen haben **e**infache Primärschlüssel → automatisch 2NF.

Ergebnis: Alle Tabellen erfüllen die 2. Normalform.

3. Normalform (3NF) – keine transitive Abhängigkeit

Tabelle	Mögliche transitive Abhängigkeit
Kunde	Straße hängt nur von KundeID ab
Buch	Preis und Seitenanzahl hängen von ISBN ab
Bestellung	KundeID und ZahlungsmodusID hängen direkt von BestellID ab
Bestellposition	Aktuellerpreis hängt von ISBN ab, aber wird bewusst gespeichert, da Preis sich ändern kann (technisch Redundanz, aber funktional nötig)
Zahlungsmodus	Rechnung, Bankeinzug, Kreditkarte hängen nur von ZahlungsmodusID ab
Rezension	Buchbeschreibung, Datum hängen direkt von RezensionID ab

Ergebnis: Alle Tabellen erfüllen die 3. Normalform

Kein Nicht-Schlüssel-Attribut hängt indirekt über ein anderes Nicht-Schlüssel-Attribut vom Primärschlüssel ab. (Ausnahme: Aktuellerpreis in Bestellposition ist *historisch gewollte Redundanz* → trotzdem akzeptiert).

Das Datenbankschema ist somit normalisiert und redundanzfrei.

1.2.4 Tabelle für Speicherberechnung

Relation	Feldname	Anz.DS	Storage Bytes	Datentyp	Länge	PK	FK	IDENTITY	Default	Computed	Index	Unique	NOT NULL	check
Buch	ISBN	50	13	char	13	x		X (1,1)			x	x	x	
Buch	Seitenanzahl		4	Integer										
Buch	Titel		100	Nvarchar	50								x	
Buch	Preis		5	Decimal	6								x	
		6100	**122**											
Rezension	RezensionID	30	4	Integer		x		X (1,1)			x	x	x	
Rezension	ISBN		13	char	13		x						x	
Rezension	Erstelldatum		3	date										
Rezension	Buch-beschreibung		4000	Nvarchar	2000									
Rezension	KundeID		4	Integer			x						x	
		120000	**4024**											
Kunde	KundeID	20	4	Integer		x		X (1,1)			x	x	x	
Kunde	Vorname		100	Nvarchar	50								x	
Kunde	Nachname		100	Nvarchar	50								x	
Kunde	Email		100	Nvarchar	50								x	
Kunde	Straße		100	Nvarchar	50								x	
Kunde	PLZ		100	Nvarchar	50								x	
		10080	**504**											
Bestellung	BestellID	40	4	Integer		x		X (1,1)			x	x	x	
Bestellung	Bestelldatum		3	date										

Bestellung	KundeID		4	Integer			x						x	
Bestellung	ZahlunsmodusID		4	Integer			x						x	
		600	**15**											
Bestellposition	BestellID	40	4	Integer		x	x	**X (1,1)**			x	x	x	
Bestellposition	ISBN		13	char	13	x	x				x	x	x	
Bestellposition	Menge		4	Integer										
Bestellposition	Aktuellerpreis		5	Decimal	6									
		1040	**26**											
Zahlunsmodus	ZahlunsmodusID	3	4	Integer		x		**X (1,1)**			x	x	x	
Zahlunsmodus	Rechnung		1	Bit										
Zahlunsmodus	Bankeinzug		1	Bit										
Zahlunsmodus	Kreditkarte		1	Bit										
		21	**7**											
Tabelle gesamt			nnn											
Gesamt in Bytes		**29841**	nnnn											
Gesamt in MB/GB		**0,02845/ 0,0000278**	nnnn											

1.3 DWH Fragen

Im Data-Warehouse (DWH) sollen auf Basis der Business-Datenbank verschiedene Auswertungen und Analysen ermöglicht werden. Die folgenden Fragestellungen dienen als Grundlage für spätere Analysen und Kennzahlenbildung.

1.3.1 Verkaufs- und Umsatzanalysen:

- Wie viele Bücher wurden im letzten Monat verkauft?
- Welcher Umsatz wurde im letzten Quartal erzielt?
- Welche Bücher verkaufen sich am Besten (Top-Seller)
- Wie hoch ist der durchschnittliche Bestellwert pro Kunde?
- Wie viele Bestellungen wurden pro Zahlungsart getätigt (Kreditkarte, Rechnungen etc.)?

1.3.2 Produktanalyse:

- Wie viele Seiten haben die meistverkauften Bücher im Durchschnitt?
- Wie viel ist der durchschnittliche Preis der meist rezensierten Bücher?
- Welche Bücher haben die besten/schlechtesten Rezensionen?

1.3.3 Kundenanalysen:

- Welche Kunden haben am meisten gekauft (Top-Kunden)?
- Wie viele Neukunden kamen im letzten Jahr dazu?
- Wie viele Rezensionen schreiben Kunden im Durchschnitt?
- Wie viele Kunden haben mehrfach bestellt?

1.3.4 Zeitbezogene Trends:

- Wie entwickelt sich der Umsatz in die letzten 12 Monaten?
- Welche Wochentage oder Monate bringen die meisten Bestellungen?
- Wie viele Rezensionen werden monatlich geschrieben?
- Wie lange dauert es im Schnitt von der ersten Rezension eines Buches bis zur ersten Bestellung?

1.3.5 Zahlungsanalysen:

- Wie verteilt sich der Umsatz auf die verschiedenen Zahlungsarten?
- Welche Zahlungsmethode wird am häufigsten genutzt?

- Gibt es Unterschiede im durchschnittlichen Bestellwert je nach Zahlungsart?

2.1 DWH Struktur

2.1.1 mER und logisches Modell

Logisches Modell

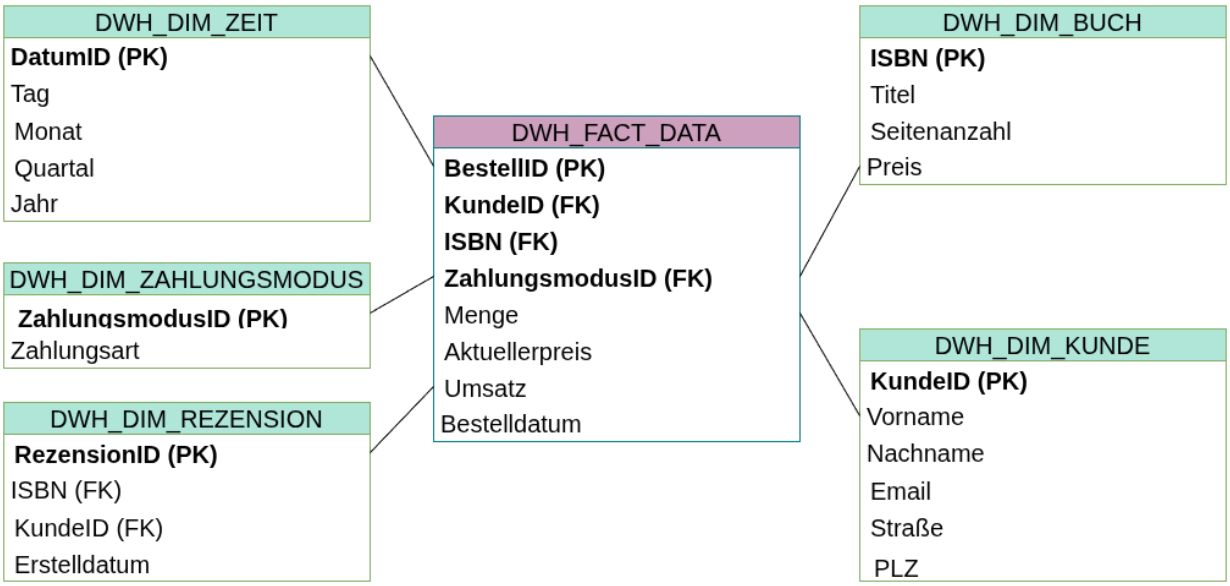
Das logische DWH-Modell basiert auf einem **Star-Schema**:

Zentrale Faktentabelle:

- Fakt_Bestellung (BestellID PK, KundeID FK ,ISBN FK , ZahlungsmodusID FK,Menge, Aktuellerpreis,Umsatz,Bestelldatum)

Dimensionen:

- Dim_Buch (ISBN PK,Titel,Seitenanzahl,Preis,gueltig_von,gueltig_bis,ativ)
- Dim_Kunde (KundeID PK,Vorname,Nachname,Email, Straße, PLZ, gueltig_von, gueltig_bis,aktiv)
- Dim_Zeit (DatumID PK,Tag, Monat,Quartal,Jahr)
- Dim_Zahlungsmodus (ZahlungsmodusID PK,Zahlungsart)
- Dim_Rezension (RezensionID PK, ISBN FK, KundeID FK, Buchbeschreibung,Erstelldatum)



Faktentabelle: Fakt_Bestellung

Diese Tabelle erfasst alle messbaren Fakten rund um Bestellungen.

Attribut	Beschreibung
BestellID (PK)	Eindeutige Identifikation der Bestellung
KundeID (FK)	Referenz auf den Kunden, der bestellt hat
ISBN (FK)	Referenz auf das Buch
ZahlungsmodusID (FK)	Verweis auf die verwendete Zahlungsart
Menge	Anzahl der bestellten Exemplare
Aktuellerpreis	Preis pro Buch zum Zeitpunkt der Bestellung
Umsatz	Berechnetes Feld (Menge × Aktuellerpreis)#ff0000
Bestelldatum	Verknüpfung mit Zeitdimension

Dimensionstabellen

1. **Dim_Buch**
   Enthält beschreibende Informationen zu Büchern.

Attribut	**Beschreibung**
ISBN (PK)	Eindeutige Buch-ID
Titel	Buchtitel
Seitenanzahl	Anzahl der Seiten
Preis	Listenpreis des Buches

2. **Dim_Kunde**
   Enthält Kundendaten aus der Tabelle *Kunde*.

Attribut	**Beschreibung**
KundeID (PK)	Eindeutige Kundenkennung
Vorname	Vorname des Kunden
Nachname	Nachname
Email	E-Mail-Adresse
Straße	Adresse
PLZ	Postleitzahl

3. **Dim_Zahlungsmodus**
   Basierend auf der Entität *Zahlungsmodus*.

Attribut	**Beschreibung**
ZahlungsmodusID (PK)	ID der Zahlungsart
Zahlungsart	Textuelle Bezeichnung (z. B. „Kreditkarte“, „Rechnung“, „Bankeinzug“)

4. **Dim_Zeit**
   Dient zur zeitlichen Analyse.

Attribut	Beschreibung
DatumID (PK)	Schlüssel für Datum
Tag	Numerischer Tag
Monat	Monat
Quartal	Quartal
Jahr	Jahr

5. **Dim_Rezension**
   Diese Dimension kann integriert werden, um Text- oder Bewertungsanalysen durchzuführen.

Attribut	Beschreibung
RezensionID (PK)	Eindeutige ID der Rezension
ISBN (FK)	Referenz auf das Buch
KundeID (FK)	Referenz auf den Kunden
Erstelldatum	Datum der Rezension
Buchbeschreibung	Kurztext der Rezension

Diese Struktur spiegelt die Beziehungen im ER-Diagramm exakt wider:

- Jede **Bestellung** besteht aus einer oder mehreren **Bestellpositionen**.
- Über die Bestellposition werden Buch, Menge und Preis mit der Bestellung verknüpft.
- Jede Bestellung ist eindeutig einem Kunden und einem Zahlungsmodus zugeordnet.

3.1 ETL – Mapping Tabelle, Beschreibung und Umsetzung

3.1.1 Beschreibung

- **Ziel:**
  Übertragung der Daten aus der operativen Business-Datenbank in das Data-Warehouse. Dabei werden alle Datenquellen vereinheitlicht, bereinigt und transformiert.

- **Datenfluss:**

    - Quelle: Tabellen Buch, Kunde, Bestellung, Bestellposition, Rezension, Zahlungsmodus.
    - Staging: Zwischenspeicherung in temporären Tabellen oder CSV-Dateien.
    - Ziel: DWH-Tabellen (Dim_*, Fakt_Bestellung).

- **Validierung:**

    - Kontrolle der Feldlängen und Datentypen.
    - Überprüfung der Fremdschlüsselbeziehungen.
    - Null-Werte in Pflichtfeldern ausgeschlossen.
    - Einheitliches Datumsformat (TT.MM.JJJJ).

- **Reihenfolge des Ladevorgangs:**

    - Laden der Dimensionstabellen (Dim_Buch, Dim_Kunde, Dim_Zahlungsmodus, Dim_Zeit, Dim_Rezension)
    - Laden der Faktentabelle (Fakt_Bestellung) mit allen Fremdschlüsselverknüpfungen

- **Fehlerprotokollierung:**

    - Alle Ladefehler (Datentyp, fehlende Schlüssel, Null-Werte) werden automatisch protokolliert.
    - Fehlerhafte Datensätze werden in einer separaten Log-Tabelle gespeichert.

3.1.2 Umsetzung

1. Extraktion
Daten werden über SQL-Skripte aus der Business-DB gelesen:

```
SELECT * FROM Buch;
SELECT * FROM Kunde;
SELECT * FROM Bestellung
JOIN Bestellposition ON Bestellung.BestellID = Bestellposition.BestellID;
SELECT * FROM Rezension;
```

Die Ergebnisse werden in Staging-Tabellen exportiert.

2. Transformation

- Trimmen von Strings, Normalisierung von Namen.
- Rundung der Preise auf 2 Dezimalstellen.
- CASE-Mapping der Zahlungsarten:

```
CASE
 WHEN Rechnung = 1 THEN 'Rechnung'
 WHEN Bankeinzug = 1 THEN 'Bankeinzug'
 WHEN Kreditkarte = 1 THEN 'Kreditkarte'
END AS Zahlungsart
```

- Berechnung der Umsatzspalte:

```
SELECT Menge * Aktuellerpreis AS Umsatz FROM Bestellposition;
```

3. Laden (Load)
Einfügen der bereinigten Daten in die Zieldatenbank:

```
cursor.execute("""
INSERT INTO Fakt_Bestellung (BestellID, KundeID, ISBN, Menge, Aktuellerpreis, Umsatz, Bestelldatum)
VALUES (?, ?, ?, ?, ?, ?, ?)
""", (bestell_id, kunde_id, isbn, menge, preis, umsatz, datum))
```

Alle Dimensionen werden vor den Faktentabellen geladen, um gültige Fremdschlüssel zu gewährleisten.

4. Kontrolle

- Vergleich von Zeilenanzahl zwischen Quelle und Ziel.
- Prüfung der referenziellen Integrität.

- Stichprobenkontrolle von Umsatzberechnung und Datumsformat.

Ergebnis:
cursor.execute(Nach der ETL-Umsetzung sind alle Business-Daten vollständig, geprüft und bereinigt in den DWH-Tabellen verfügbar.
Das System ist bereit für die Analyse- und Reportingprozess

3.1.3 Mapping-Tabelle

Source DB / CSV		Transformation / Regel	DWH DB	
Table Name	**Field**		**DIM / Fact-NAme**	**Field Name**
Buch	ISBN	Direkt übernehmen	Dim_Buch	ISBN
	Titel	Trimmen, Sonderzeichen entfernen		Titel
	Seitenanzahl	Direkt übernehmen		Seitenanzahl
	Preis	Rundung auf 2 Dezimalstellen		Preis
Kunde	KundeID	Direkt übernehmen	Dim_Kunde	KundeID
	Vorname	Groß-/Kleinschreibung normalisieren		Vorname
	Nachname	Groß-/Kleinschreibung normalisieren		Nachname
	Email	Validierung mit LIKE, CLR, oder externe Funktion		Email
	Straße	Trimmen		Straße
	PLZ	Formatierung auf 5-stellig		PLZ
Zahlungsmodus	ZahlungsmodusID	Direkt übernehmen	Dim_Zahlungsmodus	ZahlungsmodusID
	Rechnung / Bankeinzug / Kreditkarte	Case-When-Mapping: Wenn true → entsprechende Bezeichnung		Zahlungsart
Bestellung	BestellID	Direkt übernehmen	Fakt_Bestellung	BestellID
	KundeID	Join mit Dim_Kunde		KundeID
Bestellposition	ISBN	Join mit Dim_Buch		ISBN
	Menge	Direkt übernehmen		Menge
	Aktuellerpreis	Rundung auf 2 Dezimalstellen		Aktuellerpreis
	Menge * Bestellposition.Aktuellerpreis	Berechnung		Umsatz
Bestellung	Bestelldatum	Mapping auf Kalenderdimension	Dim_Zeit	DatumID
Rezension	RezensionID	Direkt übernehmen	Dim_Rezension	RezensionID
	Buchbeschreibung	Trimmen, Textlänge begrenzen		Buchbeschreibung
	Erstelldatum	Konvertierung zu Date-Format		Erstelldatum

	KundeID	Join mit Dim_Kunde		KundeID
	ISBN	Join mit Dim_Buch		ISBN

3.1.4 Mapping-Transformation und komplette Abfrage

Table Name	Field	Mapping – Transformation	Complete Query for Selecting Data
Buch	ISBN	Übernahme ohne besonderes Mapping (1:1)	CREATE TABLE DWH.dbo.Dim_Buch (ISBN CHAR(13) PRIMARY KEY, Titel NVARCHAR(50), Seitenanzahl INT, Preis DECIMAL(6,2)); INSERT INTO DWH.dbo.Dim_Buch (ISBN, Titel, Seitenanzahl, Preis) SELECT DISTINCT ISBN, Titel, Seitenanzahl, ROUND(Preis,2) FROM BusinessDB.dbo.Buch;
	Titel	Trimmen, Sonderzeichen entfernen	UPDATE DWH.dbo.Dim_Buch SET Titel = LTRIM(RTRIM(REPLACE(Titel, CHAR(13), '')));
	Seitenanzahl	Direkt übernommen	–
	Preis	Rundung auf 2 Dezimalstellen	UPDATE DWH.dbo.Dim_Buch SET Preis = ROUND(Preis,2);
Kunde	KundeID	Übernahme ohne Mapping (1:1)	CREATE TABLE DWH.dbo.Dim_Kunde (KundeID INT PRIMARY KEY, Vorname NVARCHAR(50), Nachname NVARCHAR(50), Email NVARCHAR(50), Straße NVARCHAR(50), PLZ NVARCHAR(10)); INSERT INTO DWH.dbo.Dim_Kunde SELECT DISTINCT KundeID, Vorname, Nachname, Email, Straße, PLZ FROM BusinessDB.dbo.Kunde;

	Vorname / Nachname	Normalisierung Groß-/Kleinschreibung	UPDATE DWH.dbo.Dim_Kunde SET Vorname = UPPER(LEFT(Vorname,1)) + LOWER(SUBSTRING(Vorname,2,LEN(Vorname)));
	Email	Validierung mit LIKE oder Regex	SELECT * FROM DWH.dbo.Dim_Kunde WHERE Email NOT LIKE '%_@__%.__%';
	Straße / PLZ	Trimmen und Formatierung	UPDATE DWH.dbo.Dim_Kunde SET PLZ = RIGHT('00000' + PLZ, 5);
Zahlungsmodus	ZahlungsmodusID	Direkt übernommen	CREATE TABLE DWH.dbo.Dim_Zahlungsmodus (ZahlungsmodusID INT PRIMARY KEY, Zahlungsart NVARCHAR(20)); INSERT INTO DWH.dbo.Dim_Zahlungsmodus SELECT ZahlungsmodusID, CASE WHEN Rechnung=1 THEN 'Rechnung' WHEN Bankeinzug=1 THEN 'Bankeinzug' WHEN Kreditkarte=1 THEN 'Kreditkarte' END FROM BusinessDB.dbo.Zahlungsmodus;
Bestellung	BestellID	Übernahme 1:1	CREATE TABLE DWH.dbo.Fakt_Bestellung (BestellID INT PRIMARY KEY, KundeID INT, ISBN CHAR(13), Menge INT, Aktuellerpreis DECIMAL(6,2), Umsatz DECIMAL(8,2), Bestelldatum DATE); INSERT INTO DWH.dbo.Fakt_Bestellung (BestellID, KundeID, ISBN, Menge, Aktuellerpreis, Umsatz, Bestelldatum) SELECT bp.BestellID, b.KundeID, bp.ISBN, bp.Menge, ROUND(bp.Aktuellerpreis,2), (bp.Menge * bp.Aktuellerpreis) AS Umsatz, b.Bestelldatum FROM BusinessDB.dbo.Bestellposition bp JOIN BusinessDB.dbo.Bestellung b ON bp.BestellID = b.BestellID;
	KundeID	Join mit Dim_Kunde	(Im obigen Insert integriert)

	ISBN	Join mit Dim_Buch	(Im obigen Insert integriert)
	Umsatz	Berechnet aus Menge * Aktuellerpreis	(Im obigen Insert integriert)
Dim_Zeit	DatumID	Mapping auf Kalenderdimension	CREATE TABLE DWH.dbo.Dim_Zeit (DatumID INT IDENTITY(1,1) PRIMARY KEY, Datum DATE, Tag INT, Monat INT, Jahr INT, Quartal INT); INSERT INTO DWH.dbo.Dim_Zeit (Datum, Tag, Monat, Jahr, Quartal) SELECT DISTINCT Bestelldatum, DAY(Bestelldatum), MONTH(Bestelldatum), YEAR(Bestelldatum), DATEPART(QUARTER, Bestelldatum) FROM BusinessDB.dbo.Bestellung;
Rezension	RezensionID	Direkt übernehmen	CREATE TABLE DWH.dbo.Dim_Rezension (RezensionID INT PRIMARY KEY, ISBN CHAR(13), KundeID INT, Buchbeschreibung NVARCHAR(2000), Erstelldatum DATE); INSERT INTO DWH.dbo.Dim_Rezension (RezensionID, ISBN, KundeID, Buchbeschreibung, Erstelldatum) SELECT RezensionID, ISBN, KundeID, LEFT(Buchbeschreibung,2000), CONVERT(DATE, Erstelldatum, 104) FROM BusinessDB.dbo.Rezension;
	Buchbeschreibung	Trimmen, Textlänge begrenzen	(Im obigen Insert integriert)
	Erstelldatum	Konvertierung zu Date-Format	(Im obigen Insert integriert)

3.2 SCD – Änderung im logischen Modell und Szenarien

Um die Historisierung im DWH zu ermöglichen, wurden bei den Dimensionstabellen Anpassungen am logischen Modell vorgenommen.

Scenario 1: Änderung des Buchpreises

Verwendete Technik:
Type 2 – Slowly Changing Dimension (Historisierung)

Motivation:
Ein Buchpreis kann sich im Laufe der Zeit ändern. Um die Preisentwicklung nachvollziehen zu können, wird bei jeder Preisänderung ein neuer Datensatz in Dim_Buch angelegt. Der alte Datensatz bleibt zur Historie erhalten.

SQL:

```
-- Prüfen, ob sich der Preis geändert hat
IF EXISTS (
  SELECT 1 FROM Dim_Buch
  WHERE ISBN = @ISBN AND Preis <> @NeuerPreis AND aktiv = 1
)
BEGIN
  -- Alten Datensatz schließen
  UPDATE Dim_Buch
  SET aktiv = 0,
     gueltig_bis = GETDATE()
  WHERE ISBN = @ISBN AND aktiv = 1;

  -- Neuen Datensatz anlegen
  INSERT INTO Dim_Buch (ISBN, Titel, Seitenanzahl, Preis, gueltig_von, aktiv)
  VALUES (@ISBN, @Titel, @Seitenanzahl, @NeuerPreis, GETDATE(), 1);
END;
```

Scenario 2: Änderung der Kundenadresse

Verwendete Technik:
Type 2 – Slowly Changing Dimension (Historisierung)

Motivation:
Wenn ein Kunde umzieht, sollen sowohl die alte als auch die neue Adresse im DWH verfügbar bleiben. Dadurch können Analysen nach Standort und Zeitraum korrekt durchgeführt werden.

SQL:

```
-- Prüfen, ob Adresse oder PLZ geändert wurde
IF EXISTS (
  SELECT 1 FROM Dim_Kunde
  WHERE KundeID = @KundeID
    AND (Straße <> @NeueStrasse OR PLZ <> @NeuePLZ)
    AND aktiv = 1
)
BEGIN
  -- Alten Datensatz schließen
  UPDATE Dim_Kunde
  SET aktiv = 0,
     gueltig_bis = GETDATE()
  WHERE KundeID = @KundeID AND aktiv = 1;

  -- Neuen Datensatz anlegen
  INSERT INTO Dim_Kunde (KundeID, Vorname, Nachname, Email, Straße, PLZ, gueltig_von, aktiv)
  VALUES (@KundeID, @Vorname, @Nachname, @Email, @NeueStrasse, @NeuePLZ, GETDATE(), 1);
END;
```

Scenario 3: Änderung der Kunden-E-Mail

Verwendete Technik:
Type 1 – Overwrite

Motivation:
E-Mail-Änderungen müssen nicht historisiert werden, da sie nur für die aktuelle Kontaktaufnahme relevant sind. Alte Werte werden überschrieben.

SQL:

```
UPDATE Dim_Kunde
SET Email = @NeueEmail
WHERE KundeID = @KundeID;
```

Scenario 4: Korrektur eines Buchtitels

Verwendete Technik:
Type 1 – Overwrite

Motivation:
Ein Schreibfehler oder eine Formatkorrektur im Titel ist kein historisch relevanter Vorgang. Der Wert wird einfach überschrieben.

SQL:

```
UPDATE Dim_Buch
SET Titel = @NeuerTitel
WHERE ISBN = @ISBN;
```

Scenario 5: Änderung der Zahlungsartbeschreibung

Verwendete Technik:
Type 1 – Overwrite

Motivation:
Bei Textkorrekturen oder Vereinheitlichungen der Zahlungsart (z.B. „Kreditkarte“ → „Credit Card“) wird der alte Wert überschrieben, da die Historie nicht relevant ist.

SQL:

```
UPDATE Dim_Zahlungsmodus
SET Zahlungsart = @NeueBezeichnung
WHERE ZahlungsmodusID = @ZahlungsmodusID;
```

Scenario 6: Rezension wird aktualisiert

Verwendete Technik:
Type 1 – Overwrite

Motivation:
Kleine Textänderungen oder Korrekturen in Rezensionen müssen nicht historisch nachverfolgt werden. Nur der aktuelle Inhalt ist relevant.

SQL:

```
UPDATE Dim_Rezension
SET Buchbeschreibung = @NeueBeschreibung,
  Erstelldatum = GETDATE()
WHERE RezensionID = @RezensionID;
```

Scenario 7: Einführung von Gültigkeitsfeldern (SCD 2 Vorbereitung)

Verwendete Technik:
Schema-Erweiterung zur Unterstützung von Type 2

Motivation:
Um Historisierung zu ermöglichen, werden die Felder gueltig_von, gueltig_bis und aktiv zu den betroffenen Tabellen hinzugefügt.

SQL:

```
ALTER TABLE Dim_Buch
ADD gueltig_von DATE,
  gueltig_bis DATE,
  aktiv BIT DEFAULT 1;

ALTER TABLE Dim_Kunde
ADD gueltig_von DATE,
  gueltig_bis DATE,
  aktiv BIT DEFAULT 1;
```

Ergebnis

Nach Umsetzung dieser Szenarien kann das DWH:

- Adress- und Preisänderungen historisch nachverfolgen (SCD 2),
- Textänderungen direkt überschreiben (SCD 1),
- stabile IDs unverändert beibehalten (SCD 0).
- Damit ist die Datenhistorisierung vollständig funktionsfähig.

4.1 Skripte

Im Folgenden werden alle SQL-Skripte für die Erstellung, Befüllung und Integration der Business-Datenbank (Business DB), des Data Warehouse (DWH), der ETL-Prozesse sowie der SCD-Anpassungen dargestellt.

4.1.1 Business DB – Erstellung der Tabellen

```
-- ============================================
-- 1 Datenbank erstellen und aktivieren
-- ============================================
CREATE DATABASE InternetBuchhandlung;
GO

USE InternetBuchhandlung;
GO

-- ============================================
-- 2 Tabellen erstellen
-- ============================================

-- Tabelle: Buch
CREATE TABLE Buch (
ISBN CHAR(13) PRIMARY KEY,
Titel NVARCHAR(50),
Seitenanzahl INT,
Preis DECIMAL(6,2)
);

-- Tabelle: Kunde
CREATE TABLE Kunde (
KundeID INT PRIMARY KEY IDENTITY(1,1),
Vorname NVARCHAR(50),
Nachname NVARCHAR(50),
Email NVARCHAR(100),
Straße NVARCHAR(100),
PLZ NVARCHAR(10),
Ort NVARCHAR(50)
);

-- Tabelle: Zahlungsmodus
CREATE TABLE Zahlungsmodus (
ZahlungsmodusID INT PRIMARY KEY IDENTITY(1,1),
Rechnung BIT,
Bankeinzug BIT,
Kreditkarte BIT
```

```
);

-- Tabelle: Bestellung
CREATE TABLE Bestellung (
BestellID INT PRIMARY KEY IDENTITY(1,1),
Bestelldatum DATE,
KundeID INT,
ZahlungsmodusID INT,
FOREIGN KEY (KundeID) REFERENCES Kunde(KundeID),
FOREIGN KEY (ZahlungsmodusID) REFERENCES Zahlungsmodus(ZahlungsmodusID)
);

-- Tabelle: Bestellposition
CREATE TABLE Bestellposition (
BestellID INT,
ISBN CHAR(13),
Menge INT,
Aktuellerpreis DECIMAL(6,2),
PRIMARY KEY (BestellID, ISBN),
FOREIGN KEY (BestellID) REFERENCES Bestellung(BestellID),
FOREIGN KEY (ISBN) REFERENCES Buch(ISBN)
);

-- Tabelle: Rezension
CREATE TABLE Rezension (
RezensionID INT PRIMARY KEY IDENTITY(1,1),
ISBN CHAR(13),
KundeID INT,
Buchbeschreibung NVARCHAR(2000),
Erstelldatum DATE,
FOREIGN KEY (ISBN) REFERENCES Buch(ISBN),
FOREIGN KEY (KundeID) REFERENCES Kunde(KundeID)
);

-- ==========================================
-- 3 Beispiel-Datensätze einfügen
-- ==========================================

-- Bücher
INSERT INTO Buch VALUES
('9780140449112', 'Die Odyssee', 480, 14.99),
('9783832180577', 'Der Steppenwolf', 320, 12.50),
('9783453315125', 'Der Vorleser', 210, 9.90),
('9783596294316', 'Faust I', 180, 8.50),
('9783551551672', 'Harry Potter und der Stein der Weisen', 336, 19.99);

-- Kunden
INSERT INTO Kunde (Vorname, Nachname, Email, Straße, PLZ, Ort) VALUES
('Anna', 'Schmidt', 'anna.schmidt@mail.de', 'Hauptstraße 12', '50667', 'Köln'),
('Markus', 'Weber', 'markus.weber@mail.de', 'Bahnhofstr. 5', '10115', 'Berlin'),
('Laura', 'Fischer', 'laura.fischer@mail.de', 'Lindenweg 3', '80331', 'München'),
```

```
('Tim', 'Keller', 'tim.keller@mail.de', 'Am See 8', '20095', 'Hamburg');

-- Zahlungsmodi
INSERT INTO Zahlungsmodus (Rechnung, Bankeinzug, Kreditkarte) VALUES
(0, 0, 1), -- Kreditkarte
(1, 0, 0), -- Rechnung
(0, 1, 0); -- Bankeinzug

-- Bestellungen
INSERT INTO Bestellung (Bestelldatum, KundeID, ZahlungsmodusID) VALUES
('2024-09-01', 1, 1),
('2024-09-03', 2, 2),
('2024-09-05', 3, 3),
('2024-09-07', 4, 1),
('2024-09-08', 1, 2);

-- Bestellpositionen
INSERT INTO Bestellposition VALUES
(1, '9780140449112', 1, 14.99),
(1, '9783551551672', 1, 19.99),
(2, '9783832180577', 2, 12.50),
(3, '9783596294316', 1, 8.50),
(4, '9783453315125', 3, 9.90),
(5, '9780140449112', 1, 14.99);

-- Rezensionen
INSERT INTO Rezension (ISBN, KundeID, Buchbeschreibung, Erstelldatum) VALUES
('9780140449112', 1, 'Ein großartiges Epos, spannend bis zum Schluss.', '2024-09-10'),
('9783551551672', 3, 'Sehr fantasievoll, gut für junge Leser.', '2024-09-12'),
('9783832180577', 2, 'Tiefgründig, aber manchmal schwer verständlich.', '2024-09-15'),
('9783596294316', 4, 'Klassiker, den man gelesen haben muss.', '2024-09-17'),
('9783453315125', 1, 'Emotional und bewegend, sehr empfehlenswert.', '2024-09-20');
```

4.1.2 DWH – Erstellung der Dimensionen und Faktentabelle

```
-- ============================================
-- 1. DWH-Datenbank erstellen
-- ============================================
CREATE DATABASE DWH_InternetBuchhandlung;
GO
USE DWH_InternetBuchhandlung;
GO

-- ============================================
-- 2. Dimensionstabellen erstellen
-- ============================================
-- Dimension: Buch
CREATE TABLE Dim_Buch (
```

```
  ISBN CHAR(13) PRIMARY KEY,
  Titel NVARCHAR(50),
  Seitenanzahl INT,
  Preis DECIMAL(6,2),
  gueltig_von DATE,
  gueltig_bis DATE,
  aktiv BIT DEFAULT 1
);

-- Dimension: Kunde
CREATE TABLE Dim_Kunde (
  KundeID INT PRIMARY KEY,
  Vorname NVARCHAR(50),
  Nachname NVARCHAR(50),
  Email NVARCHAR(100),
  Straße NVARCHAR(100),
  PLZ NVARCHAR(10),
  gueltig_von DATE,
  gueltig_bis DATE,
  aktiv BIT DEFAULT 1
);

-- Dimension: Zahlungsmodus
CREATE TABLE Dim_Zahlungsmodus (
  ZahlungsmodusID INT PRIMARY KEY,
  Zahlungsart NVARCHAR(20)
);

-- Dimension: Zeit
CREATE TABLE Dim_Zeit (
  DatumID INT IDENTITY(1,1) PRIMARY KEY,
  Datum DATE,
  Tag INT,
  Monat INT,
  Quartal INT,
  Jahr INT
);

-- Dimension: Rezension
CREATE TABLE Dim_Rezension (
  RezensionID INT PRIMARY KEY,
  ISBN CHAR(13),
  KundeID INT,
  Buchbeschreibung NVARCHAR(2000),
  Erstelldatum DATE,
  FOREIGN KEY (ISBN) REFERENCES Dim_Buch(ISBN),
  FOREIGN KEY (KundeID) REFERENCES Dim_Kunde(KundeID)
);

-- ============================================
-- 3. Faktentabelle erstellen
```

```
-- ============================================
CREATE TABLE Fakt_Bestellung (
  BestellID INT PRIMARY KEY,
  KundeID INT,
  ISBN CHAR(13),
  ZahlungsmodusID INT,
  Menge INT,
  Aktuellerpreis DECIMAL(6,2),
  Umsatz DECIMAL(8,2),
  Bestelldatum DATE,
  FOREIGN KEY (KundeID) REFERENCES Dim_Kunde(KundeID),
  FOREIGN KEY (ISBN) REFERENCES Dim_Buch(ISBN),
  FOREIGN KEY (ZahlungsmodusID) REFERENCES Dim_Zahlungsmodus(ZahlungsmodusID)
);
```

4.1.3 ETL – Umsetzung mit Stored Procedure

```
USE DWH_InternetBuchhandlung;
GO

-------------------------------------------------------------
-- Ensure Schema and Control Table Exist Before ETL Starts
-------------------------------------------------------------
IF NOT EXISTS (SELECT * FROM sys.schemas WHERE name = 'dwh')
BEGIN
  PRINT 'Creating schema [dwh]...';
  EXEC('CREATE SCHEMA dwh AUTHORIZATION dbo;');
END
GO

IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'ETL_Control' AND SCHEMA_NAME(schema_id) = 'dwh')
BEGIN
  PRINT 'Creating control table [dwh.ETL_Control]...';
  CREATE TABLE dwh.ETL_Control (
     ETL_ID INT IDENTITY(1,1) PRIMARY KEY,
     ProcedureName VARCHAR(100),
     StartTime DATETIME,
     EndTime DATETIME,
     Status VARCHAR(20),
     RowsInserted INT,
     ErrorMessage VARCHAR(4000)
  );
END
GO

-------------------------------------------------------------
```

```
-- Main ETL Procedure
-----------------------------------------------------------------

CREATE OR ALTER PROCEDURE dwh.usp_ETL_InternetBuchhandlung
AS
BEGIN
  SET NOCOUNT ON;
  DECLARE @StartTime DATETIME = GETDATE();
  DECLARE @Status VARCHAR(20) = 'Started';
  DECLARE @RowsInserted INT = 0;
  DECLARE @ErrorMessage VARCHAR(4000) = NULL;

  BEGIN TRY
    BEGIN TRANSACTION;

    PRINT 'ETL started at ' + CONVERT(VARCHAR(30), @StartTime, 120);

    ------------------------------------------------------
    -- 1 Extract Data from Source DB
    ------------------------------------------------------
    IF OBJECT_ID('temp_Buch', 'U') IS NOT NULL DROP TABLE temp_Buch;
    IF OBJECT_ID('temp_Kunde', 'U') IS NOT NULL DROP TABLE temp_Kunde;
    IF OBJECT_ID('temp_Bestellung', 'U') IS NOT NULL DROP TABLE temp_Bestellung;
    IF OBJECT_ID('temp_Rezension', 'U') IS NOT NULL DROP TABLE temp_Rezension;
    IF OBJECT_ID('temp_Zahlungsmodus', 'U') IS NOT NULL DROP TABLE temp_Zahlungsmodus;

    SELECT * INTO temp_Buch FROM InternetBuchhandlung.dbo.Buch;
    SELECT * INTO temp_Kunde FROM InternetBuchhandlung.dbo.Kunde;
    SELECT b.BestellID, b.KundeID, p.ISBN, p.Menge, p.Aktuellerpreis, b.Bestelldatum, b.ZahlungsmodusID
    INTO temp_Bestellung
    FROM InternetBuchhandlung.dbo.Bestellung b
    JOIN InternetBuchhandlung.dbo.Bestellposition p ON b.BestellID = p.BestellID;
    SELECT * INTO temp_Rezension FROM InternetBuchhandlung.dbo.Rezension;
    SELECT ZahlungsmodusID,
        CASE WHEN Rechnung=1 THEN 'Rechnung'
           WHEN Bankeinzug=1 THEN 'Bankeinzug'
           WHEN Kreditkarte=1 THEN 'Kreditkarte'
        END AS Zahlungsart
    INTO temp_Zahlungsmodus
    FROM InternetBuchhandlung.dbo.Zahlungsmodus;

    ------------------------------------------------------
    -- 2 Transform Data
    ------------------------------------------------------
    UPDATE temp_Buch
    SET Titel = LTRIM(RTRIM(Titel)),
      Preis = ROUND(Preis, 2);

    UPDATE temp_Kunde
    SET PLZ = RIGHT('00000' + PLZ, 5),
      Vorname = UPPER(LEFT(Vorname, 1)) + LOWER(SUBSTRING(Vorname, 2, LEN(Vorname))),
```

```
    Nachname = UPPER(LEFT(Nachname, 1)) + LOWER(SUBSTRING(Nachname, 2, LEN(Nachname)));

  UPDATE temp_Rezension
  SET Buchbeschreibung = LEFT(Buchbeschreibung, 2000),
    Erstelldatum = CONVERT(DATE, Erstelldatum, 104);

  ------------------------------------------------------
  -- 3 Load Data into DWH
  ------------------------------------------------------
  PRINT 'Loading data into DWH...';

  -- Load Dim_Buch
  DELETE FROM dwh.Dim_Buch;
  INSERT INTO dwh.Dim_Buch (ISBN, Titel, Seitenanzahl, Preis, gueltig_von, aktiv)
  SELECT DISTINCT ISBN, Titel, Seitenanzahl, Preis, GETDATE(), 1 FROM temp_Buch;

  -- Load Dim_Kunde
  DELETE FROM dwh.Dim_Kunde;
  INSERT INTO dwh.Dim_Kunde (KundeID, Vorname, Nachname, Email, Straße, PLZ, gueltig_von, aktiv)
  SELECT DISTINCT KundeID, Vorname, Nachname, Email, Straße, PLZ, GETDATE(), 1 FROM temp_Kunde;

  -- Load Dim_Zahlungsmodus
  DELETE FROM dwh.Dim_Zahlungsmodus;
  INSERT INTO dwh.Dim_Zahlungsmodus (ZahlungsmodusID, Zahlungsart)
  SELECT DISTINCT ZahlungsmodusID, Zahlungsart FROM temp_Zahlungsmodus;

  -- Load Dim_Zeit
  DELETE FROM dwh.Dim_Zeit;
  INSERT INTO dwh.Dim_Zeit (Datum, Tag, Monat, Jahr, Quartal)
  SELECT DISTINCT Bestelldatum, DAY(Bestelldatum), MONTH(Bestelldatum),
      YEAR(Bestelldatum), DATEPART(QUARTER, Bestelldatum)
  FROM temp_Bestellung;

 -- Load Dim_Rezension
  DELETE FROM dwh.Dim_Rezension;
  INSERT INTO dwh.Dim_Rezension (RezensionID, ISBN, KundeID, Buchbeschreibung, Erstelldatum)
  SELECT RezensionID, ISBN, KundeID, Buchbeschreibung, Erstelldatum FROM temp_Rezension;

-- Load Fakt_Bestellung
  DELETE FROM dwh.Fakt_Bestellung;
  INSERT INTO dwh.Fakt_Bestellung (BestellID, KundeID, ISBN, ZahlungsmodusID, Menge, Aktuellerpreis, Umsatz,
Bestelldatum)
  SELECT b.BestellID, b.KundeID, b.ISBN, b.ZahlungsmodusID,
      b.Menge, ROUND(b.Aktuellerpreis, 2), (b.Menge * b.Aktuellerpreis), b.Bestelldatum
  FROM temp_Bestellung b;

  SET @RowsInserted = @@ROWCOUNT;

  ------------------------------------------------------
  -- 4 Log Success
  ------------------------------------------------------
```

```
    COMMIT TRANSACTION;
    SET @Status = 'Success';
    INSERT INTO dwh.ETL_Control (ProcedureName, StartTime, EndTime, Status, RowsInserted)
    VALUES ('usp_ETL_InternetBuchhandlung', @StartTime, GETDATE(), @Status, @RowsInserted);

  ------------------------------------------------------
  -- 5 Validation and Completion
  ------------------------------------------------------

    PRINT 'ETL completed successfully.';

  END TRY
  BEGIN CATCH
    ROLLBACK TRANSACTION;
    SET @Status = 'Failed';
    SET @ErrorMessage = ERROR_MESSAGE();

    INSERT INTO dwh.ETL_Control (ProcedureName, StartTime, EndTime, Status, ErrorMessage)
    VALUES ('usp_ETL_InternetBuchhandlung', @StartTime, GETDATE(), @Status, @ErrorMessage);

    PRINT 'ETL failed: ' + @ErrorMessage;
  END CATCH
END
GO
```

Test ETL:

```
USE DWH_InternetBuchhandlung;
GO

-------------------------------------------------------------
--1 Create Test Results Table (if not exists)
-------------------------------------------------------------
IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'ETL_TestResults' AND SCHEMA_NAME(schema_id) =
'dwh')
BEGIN
  CREATE TABLE dwh.ETL_TestResults (
     TestID INT IDENTITY(1,1) PRIMARY KEY,
     TestName NVARCHAR(100),
     TestCategory NVARCHAR(50),
     Result NVARCHAR(10),
     ExpectedValue NVARCHAR(100),
     ActualValue NVARCHAR(100),
     TestTimestamp DATETIME DEFAULT GETDATE()
  );
END
GO

-------------------------------------------------------------
--2 Test Procedure Definition
-------------------------------------------------------------
CREATE OR ALTER PROCEDURE dwh.usp_Test_ETL_InternetBuchhandlung
AS
BEGIN
  SET NOCOUNT ON;
  DECLARE @TestName NVARCHAR(100),
        @Category NVARCHAR(50),
        @Expected NVARCHAR(100),
        @Actual NVARCHAR(100),
        @Result NVARCHAR(10);

  ---------------------------------------------------------
  -- Test 1: Verify Last ETL Run Success
  ---------------------------------------------------------
  SET @TestName = 'ETL run status check';
  SET @Category = 'Control';

  SELECT TOP 1 @Actual = Status
  FROM dwh.ETL_Control
  ORDER BY ETL_ID DESC;

  SET @Expected = 'Success';
  SET @Result = CASE WHEN @Actual = @Expected THEN 'PASS' ELSE 'FAIL' END;

  INSERT INTO dwh.ETL_TestResults (TestName, TestCategory, Result, ExpectedValue, ActualValue)
```

```
VALUES (@TestName, @Category, @Result, @Expected, @Actual);


-------------------------------------------------------------
-- Test 2: Record count consistency (Fact table)
-------------------------------------------------------------
SET @TestName = 'Row count match (Fakt_Bestellung)';
SET @Category = 'Consistency';

DECLARE @SrcCount INT, @DwhCount INT;
SELECT @SrcCount = COUNT(*) FROM InternetBuchhandlung.dbo.Bestellposition;
SELECT @DwhCount = COUNT(*) FROM dwh.Fakt_Bestellung;

SET @Expected = CAST(@SrcCount AS NVARCHAR(100));
SET @Actual   = CAST(@DwhCount AS NVARCHAR(100));
SET @Result   = CASE WHEN @SrcCount = @DwhCount THEN 'PASS' ELSE 'FAIL' END;

INSERT INTO dwh.ETL_TestResults (TestName, TestCategory, Result, ExpectedValue, ActualValue)
VALUES (@TestName, @Category, @Result, @Expected, @Actual);


-------------------------------------------------------------
-- Test 3: Umsatz consistency between Source and DWH
-------------------------------------------------------------
SET @TestName = 'Total revenue consistency';
SET @Category = 'Financial';

DECLARE @SrcUmsatz DECIMAL(18,2), @DwhUmsatz DECIMAL(18,2);
SELECT @SrcUmsatz = SUM(p.Menge * p.Aktuellerpreis)
FROM InternetBuchhandlung.dbo.Bestellposition p;

SELECT @DwhUmsatz = SUM(Umsatz) FROM dwh.Fakt_Bestellung;

SET @Expected = CAST(@SrcUmsatz AS NVARCHAR(100));
SET @Actual   = CAST(@DwhUmsatz AS NVARCHAR(100));
SET @Result   = CASE WHEN ABS(@SrcUmsatz - @DwhUmsatz) < 0.01 THEN 'PASS' ELSE 'FAIL' END;

INSERT INTO dwh.ETL_TestResults (TestName, TestCategory, Result, ExpectedValue, ActualValue)
VALUES (@TestName, @Category, @Result, @Expected, @Actual);


-------------------------------------------------------------
-- Test 4: Customer Name Transformation (Capitalization)
-------------------------------------------------------------
SET @TestName = 'Customer name formatting';
SET @Category = 'Transformation';

IF EXISTS (
  SELECT 1 FROM dwh.Dim_Kunde
  WHERE Vorname COLLATE SQL_Latin1_General_CP1_CS_AS NOT LIKE '[A-Z]%'
)
```

```
  SET @Result = 'FAIL';
ELSE
  SET @Result = 'PASS';

INSERT INTO dwh.ETL_TestResults (TestName, TestCategory, Result, ExpectedValue, ActualValue)
VALUES (@TestName, @Category, @Result, 'Capitalized Names', 'Checked');


-------------------------------------------------------
-- Test 5: Postal Code Padding
-------------------------------------------------------
SET @TestName = 'Postal code formatting';
SET @Category = 'Transformation';

IF EXISTS (
  SELECT 1 FROM dwh.Dim_Kunde WHERE LEN(PLZ) <> 5
)
  SET @Result = 'FAIL';
ELSE
  SET @Result = 'PASS';

INSERT INTO dwh.ETL_TestResults (TestName, TestCategory, Result, ExpectedValue, ActualValue)
VALUES (@TestName, @Category, @Result, '5 digits', 'Checked');


-------------------------------------------------------
-- Test 6: Date Dimension Coverage
-------------------------------------------------------
SET @TestName = 'Date dimension completeness';
SET @Category = 'Dimension';

DECLARE @DateCount INT;
SELECT @DateCount = COUNT(DISTINCT Bestelldatum)
FROM InternetBuchhandlung.dbo.Bestellung;

DECLARE @DimDateCount INT;
SELECT @DimDateCount = COUNT(*) FROM dwh.Dim_Zeit;

SET @Expected = CAST(@DateCount AS NVARCHAR(100));
SET @Actual   = CAST(@DimDateCount AS NVARCHAR(100));
SET @Result   = CASE WHEN @DateCount = @DimDateCount THEN 'PASS' ELSE 'FAIL' END;

INSERT INTO dwh.ETL_TestResults (TestName, TestCategory, Result, ExpectedValue, ActualValue)
VALUES (@TestName, @Category, @Result, @Expected, @Actual);


PRINT 'ETL validation completed. See results in [dwh.ETL_TestResults] table.';
END
GO
```

4.1.4 SCD – Umsetzung und Skripte

Zur Umsetzung der *Slowly Changing Dimensions (SCD)* wurden in der DWH-Datenbank verschiedene Stored Procedures entwickelt.

Für **SCD Typ 2** (Historisierung) wurden in den Tabellen *Dim_Buch* und *Dim_Kunde* zusätzliche Felder (*gueltig_von*, *gueltig_bis*, *aktiv*) ergänzt, um Preis- und Adressänderungen zeitlich nachvollziehen zu können. Bei jeder Änderung wird der alte Datensatz deaktiviert und ein neuer mit aktuellem Wert eingefügt.

Für **SCD Typ 1** (Überschreiben) werden Felder ohne historische Relevanz – wie *Email* oder *Buchbeschreibung* – direkt aktualisiert.
Alle Änderungen werden in der Tabelle *dwh.SCD_Log* protokolliert, um eine vollständige Nachverfolgung der Datenänderungen zu gewährleisten.
Zur Validierung sind Testaufrufe integriert, mit denen Preis-, Adress- und Textänderungen überprüft werden können. Damit ist das DWH um eine konsistente und nachvollziehbare Historisierungslogik erweitert.

```
USE DWH_InternetBuchhandlung;
GO

-------------------------------------------------------------
-- 1. Add SCD Columns if Not Yet Added
-------------------------------------------------------------
IF COL_LENGTH('Dim_Buch', 'gueltig_von') IS NULL
BEGIN
  ALTER TABLE Dim_Buch
  ADD gueltig_von DATE,
     gueltig_bis DATE,
     aktiv BIT DEFAULT 1;
END;

IF COL_LENGTH('Dim_Kunde', 'gueltig_von') IS NULL
BEGIN
  ALTER TABLE Dim_Kunde
  ADD gueltig_von DATE,
     gueltig_bis DATE,
     aktiv BIT DEFAULT 1;
END;
GO

-------------------------------------------------------------
-- 2. Create SCD Log Table
-------------------------------------------------------------
IF OBJECT_ID('dwh.SCD_Log', 'U') IS NULL
BEGIN
```

```
  CREATE TABLE dwh.SCD_Log (
    LogID INT IDENTITY(1,1) PRIMARY KEY,
    Tabelle NVARCHAR(50),
    Schluesselwert NVARCHAR(50),
    Aktion NVARCHAR(100),
    Änderungsdatum DATETIME DEFAULT GETDATE(),
    Benutzer NVARCHAR(50)
  );
END;
GO

-------------------------------------------------------------
-- 3. Procedure: SCD2 – Buchpreis ändern (Historisierung)
-------------------------------------------------------------
CREATE OR ALTER PROCEDURE dwh.usp_Update_Buchpreis
  @ISBN CHAR(13),
  @NeuerPreis DECIMAL(6,2)
AS
BEGIN
  SET NOCOUNT ON;
  BEGIN TRANSACTION;

  UPDATE Dim_Buch
  SET aktiv = 0,
    gueltig_bis = GETDATE()
  WHERE ISBN = @ISBN AND aktiv = 1;

  INSERT INTO Dim_Buch (ISBN, Titel, Seitenanzahl, Preis, gueltig_von, aktiv)
  SELECT ISBN, Titel, Seitenanzahl, @NeuerPreis, GETDATE(), 1
  FROM Dim_Buch
  WHERE ISBN = @ISBN;

  INSERT INTO dwh.SCD_Log (Tabelle, Schluesselwert, Aktion, Benutzer)
  VALUES ('Dim_Buch', @ISBN, 'Preisänderung - SCD2', SYSTEM_USER);

  COMMIT TRANSACTION;
END;
GO

-------------------------------------------------------------
-- 4. Procedure: SCD2 – Kundenadresse ändern (Historisierung)
-------------------------------------------------------------
CREATE OR ALTER PROCEDURE dwh.usp_Update_Kundenadresse
  @KundeID INT,
  @NeueStrasse NVARCHAR(100),
  @NeuePLZ NVARCHAR(10)
AS
BEGIN
  SET NOCOUNT ON;
  BEGIN TRANSACTION;
```

```
  UPDATE Dim_Kunde
  SET aktiv = 0,
    gueltig_bis = GETDATE()
  WHERE KundeID = @KundeID AND aktiv = 1;

  INSERT INTO Dim_Kunde (KundeID, Vorname, Nachname, Email, Straße, PLZ, gueltig_von, aktiv)
  SELECT KundeID, Vorname, Nachname, Email, @NeueStrasse, @NeuePLZ, GETDATE(), 1
  FROM Dim_Kunde
  WHERE KundeID = @KundeID;

  INSERT INTO dwh.SCD_Log (Tabelle, Schluesselwert, Aktion, Benutzer)
  VALUES ('Dim_Kunde', CAST(@KundeID AS NVARCHAR(50)), 'Adressänderung - SCD2', SYSTEM_USER);

  COMMIT TRANSACTION;
END;
GO

-------------------------------------------------------------
-- 5. Procedure: SCD1 – Kunden-E-Mail überschreiben
-------------------------------------------------------------
CREATE OR ALTER PROCEDURE dwh.usp_Update_KundenEmail
  @KundeID INT,
  @NeueEmail NVARCHAR(100)
AS
BEGIN
  UPDATE Dim_Kunde
  SET Email = @NeueEmail
  WHERE KundeID = @KundeID;

  INSERT INTO dwh.SCD_Log (Tabelle, Schluesselwert, Aktion, Benutzer)
  VALUES ('Dim_Kunde', CAST(@KundeID AS NVARCHAR(50)), 'Email-Update - SCD1', SYSTEM_USER);
END;
GO

-------------------------------------------------------------
-- 6. Procedure: SCD1 – Rezensionstext überschreiben
-------------------------------------------------------------
CREATE OR ALTER PROCEDURE dwh.usp_Update_Rezension
  @RezensionID INT,
  @NeueBeschreibung NVARCHAR(2000)
AS
BEGIN
  UPDATE Dim_Rezension
  SET Buchbeschreibung = @NeueBeschreibung
  WHERE RezensionID = @RezensionID;

  INSERT INTO dwh.SCD_Log (Tabelle, Schluesselwert, Aktion, Benutzer)
  VALUES ('Dim_Rezension', CAST(@RezensionID AS NVARCHAR(50)), 'Rezension-Update - SCD1', SYSTEM_USER);
END;
GO
```

```
-------------------------------------------------------------
-- 7. Beispielhafte Tests (Demonstration)
-------------------------------------------------------------
-- Preisänderung eines Buchs (SCD2)
EXEC dwh.usp_Update_Buchpreis @ISBN = '9780140449112', @NeuerPreis = 15.99;

-- Adressänderung eines Kunden (SCD2)
EXEC dwh.usp_Update_Kundenadresse @KundeID = 1, @NeueStrasse = 'Musterstraße 99', @NeuePLZ = '50670';

-- E-Mail-Änderung (SCD1)
EXEC dwh.usp_Update_KundenEmail @KundeID = 2, @NeueEmail = 'neue.mail@example.de';

-- Rezensionstext aktualisieren (SCD1)
EXEC dwh.usp_Update_Rezension @RezensionID = 3, @NeueBeschreibung = 'Überarbeitete Rezension mit mehr
Details.';

-- Prüfung der Logeinträge
SELECT * FROM dwh.SCD_Log ORDER BY Änderungsdatum DESC;
GO
```