# 2.1.1

Das relationale Datenbankmodell stellt eine Datenbank schriftlich ohne Grafik dar. Dabei wird jede Tabelle mit ihren Attributen angegeben. Primärschlüssel werden unterstrichen und Fremdschlüssel mit einem # gekennzeichnet.

Ein Vorteil des Relationalen Datenbankmodelles ist es das man in ihm die Fremdschlüssel erkennen kann. Ein Nachteil ist das man nicht bestimmen kann von wie vielen zu wie vielen die Beziehung ausgeht.

# 2.1.2

1. Überführung aus dem ER-Modell in das Relationale Modell:

Hierbei implementiere ich die Auflösungstabellen direkt mit. Dabei wird die Beziehung Impfen trotz, dass sie nur eine n zu 1 Beziehung ist zu einer Auflösungstabelle da sie eigene Attribute hat. Genauso wird liefert eine Auflösungstabelle. Die Fremdschlüssel werden immer von der 1 Seite bei einer 1 zu n Beziehung genommen. Eine Auflösungstabelle integriert die Fremdschlüssel ihrer Nachbartabellen automatisch.

Person{personenID, risiko, name, adresse, gebDatum}

Impfen{#personenID, #bezeichnung, termin1, termin2}

Standort{ortID, temperatur, istAnszahlImpfdosen, kapazitaetImpfdosen, #bezeichnung}

Impfstoff{bezeichnung, lagerTemperatur, gruppe, entwickler}

Liefert{#bezeichnung, #prodID, anzahlImpfdosen, termin}

Produzent{prodID, land, name, plz, ort}

2. Regeln zur Normalisierung:

* Die **Nullte Normalform** ist dann gegeben, wenn alle **Informationen in einer Tabelle vorhanden** sind und noch **unnormalisiert vorliegen**.
* Die **Erste Normalform (1NF)** ist dann gegeben, wenn **alle Informationen** in einer Tabelle **atomar vorliegen**.
* Ein Relationstyp (Tabelle) befindet sich genau dann in der **zweiten Normalform (2NF)**, wenn er sich in der [ersten Normalform (1NF)](https://www.datenbanken-verstehen.de/datenmodellierung/normalisierung/erste-normalform/) befindet und jedes Nichtschlüsselattribut von jedem Schlüsselkandidaten [voll funktional](https://www.datenbanken-verstehen.de/datenmodellierung/normalisierung/abhaengigkeiten-normalisierung/) abhängig ist.  Eine **vollständig funktionale Abhängigkeit** liegt dann vor, wenn dass Nicht-Schlüsselattribut nicht nur von einem Teil der Attribute eines zusammengesetzten Schlüsselkandidaten funktional abhängig ist, sondern von allen Teilen eines Relationstyps.
* Ein Relationstyp befindet sich genau dann in der **dritten Normalform (3NF)**, wenn er sich in der [zweiten Normalform (2NF)](https://www.datenbanken-verstehen.de/datenmodellierung/normalisierung/zweite-normalform/) befindet und kein Nichtschlüsselattribut [transitiv](https://www.datenbanken-verstehen.de/datenmodellierung/normalisierung/abhaengigkeiten-normalisierung/) von einem Kandidatenschlüssel abhängt.

3. Normalisierung umsetzen:

1. Die Informationen liegen schon vollständig for.
2. Die Attribute sind noch nicht alle Atomar.

Person{personenID, risiko, name, adresse, gebDatum} =>

Person{personenID, risiko, vorname, nachname, adresse, gebDatum} =>

Person{personenID, risiko, vorname, nachname, ort, plz, Straße, Hausnummer, gebDatum}

1. Die vollfunktionalle Abhängigkeit liegt schon for.
2. Die Attribute sind noch teils von anderen nicht Schlüsselattributen Abhängig

Person{personenID, risiko, vorname, nachname, #plz, Straße, Hausnummer, gebDatum}

Impfen{#personenID, #bezeichnung, termin1, termin2}

Standort{ortID, temperatur, istAnszahlImpfdosen, kapazitaetImpfdosen, #bezeichnung}

Impfstoff{bezeichnung, lagerTemperatur, gruppe, entwickler}

Liefert{#bezeichnung, #prodID, anzahlImpfdosen, termin}

Produzent{prodID, land, name, plz}

Platz{plz, #ort}

Ort{ort, land}

Möglicherweise liegt nun eine Übermäßige Kleinligkeit for die nicht Sinnvoll zu nutzen währe.

# 2.2.1

Aufgabe: Formulieren Sie eine SQL-Anweisung, die alle Personen mit Personen-ID, Vor- und Nachnamen sowie dem Termin der ersten Impfung auflistet, die bisher nur einmal mit dem Impfstoff mit der Bezeichnungen BT223a2 geimpft wurden.

SELECT p.PersonenID, p.nachname, p.vorname, i.termin1

FROM Person p

INNER JOIN Impfen i ON p.PersonenID = i.PersonenID

INNER JOIN Impfstopff imp ON i.bezeichnung = imp.bezeichung

WHERE imp.bezeichnung = 'BT223a2'

AND i.termin2 IS NULL;

# 2.2.2

Aufgabe: Implementieren Sie eine SQL-Anweisung, die alle Impfstoffgruppen mit der Anzahl der im 1. Quartal 2021 zum zweiten Mal erfolgten Impfungen, absteigend sortiert nach der Anzahl, ausgibt.

SELECT imp.gruppe, COUNT(\*) AS anzahl\_zweiter\_impfungen

FROM Impfstoff imp

INNER JOIN Impfen i ON imp.bezeichnung = i.bezeichnung

WHERE i.termin2 IS NOT NULL

AND i.termin2 BETWEEN '2021-01-01' AND '2021-03-31'

GROUP BY imp.gruppe

ORDER BY anzahl\_zweiter\_impfungen DESC;

# 2.2.3

Aufgabe: Der kanadische Impfstoffproduzent PharmaCorn (ID 96) in Vancouver (Postleitzzahl BC V6T 1Z3) liefert am 26.04.2021 erstmalig 2500 Dosen des Impfstoffs BT223a2.

Entwickeln Sie die SQL-Anweisungen, um die Datenbank zu ergänzen und die Lagermenge zu aktualisieren.

Hinweise:

1. Es existieren bereits mehrere Lagerstandorte für den Impfstoff BT223a2.

2. Die 2500 neune Dosen können nur eingelagert werden, wenn an einem Standort genügend Platz vorhanden ist.

3. Sie können davon ausgehen, dass es mindestens einen Standort gibt, der die gelieferte Menge aufnehmen kann.

4. Der Standort mit der gerinsten Anzahl an BT223a2-Impfdosen ist als Lagerstandort zu wählen.

1. Porduzent erstellen

INSERT INTO Produzent (prodID, name, plz, ort, land)

VALUES (96, “ PharmaCorn”, “BC V6T 1Z32”, “Vancouver”, “Canada”);

2. Liefert erstellen

INSERT INTO Liefert (prodID, bezeichnung, termin, anzahlImpfdosen)

VALUES (96, „BT223a2“, „26.04.2021“, 2500)

3. Impfstoff

Zu schließen aus Kontext und letzter Aufgabe existiert das schon.

4. Standort