# Databasbeskrivning – Orhan & Patric

Här nedan kommer vi gå igenom varje tabell var för sig. Utgångspunkten blir den centrala knytpunkten (Product), där vi börjar med vänster benet, sedan mitten benet och avslutningsvis följt av höger ben.

## Tabeller skrivs i följande format:

| *Namn på kolumn* | *Datatyp på kolumn* | *Kommentarer* |
|---|---|---|

## Product (Agerar centrallager)

| ID | int | |
|---|---|---|
| Name | nvarchar(50) | |
| Manufacturer | nvarchar(50) | |
| Description | nvarchar(255) | För möjlighet till längre beskrivning. Är Nullable. |
| Price | decimal(7, 2) | Anser att ingenting kostar mer än 99999,00 kr |
| Weight(g) | int | Använder int eftersom vi räknar i gram |
| Height(cm) | tinyint | Inga produkter är längre än 255cm (tinyint tar bara 1 byte) – prestandasyfte, tillåter NULL då dessa inte är avgörande för försäljning. |
| Width(cm) | tinyint | Samma som ovan |
| Depth(cm) | tinyint | Samma som ovan |
| DiscountID | int | För koppling till Discount tabellen |

**Constraints**

- **Check:** Price >= 1 – Man ska inte kunna sätta priset på mindre än 1kr av misstag.
- **Trigger:** En rabatt kan inte sänka en produkts pris till lägre än 1 kr. Minimumpriset för en produkt blir därmed 1 kr. Triggern kollar så att rabatten inte är större än (Price – 1). Därav varför priset, efter en fixed rabatt, inte kan bli lägre än 1 kr. Vi kollar även så att fixed rabatt inte är mindre eller lika med 0 kr.

```
    IF EXISTS(
        SELECT Product.[Name], Discount.DiscountFixed, Product.[Price]
        FROM Product
        JOIN Discount ON Discount.ID = Product.DiscountID
        GROUP BY Product.[Name], Discount.DiscountFixed, Product.[Price]
        HAVING Discount.DiscountFixed > SUM(Product.[Price] - 1) OR Discount.DiscountFixed <= 0
    )
    BEGIN
        THROW 50000, 'Discount can''t exceed or be equal to the product price. Price after discount must atleast be (1kr)', 1;
    END
END
```

## StoreStock

| | | |
|---|---|---|
| ProductID | int | Koppling till Product tabellen |
| StoreID | int | Koppling till Store tabellen |
| Quantity | int | Lagersaldo till butik |

**Övrig kommentar:**

- Vi använder StoreStock som en kopplingstabell (Join-tabell) mellan Product och Store.

## Store

| | | |
|---|---|---|
| ID | int | |
| Name | nvarchar(50) | |
| City | nvarchar(50) | |
| ZipCode | int | Int av anledning ex: 955 32, smallint räcker inte. |
| StreetName | nvarchar(50) | |
| BuildingNumber | smallint | Anser inte att ett husnummer är större än 32.767 |
| Size | int | Med tanke på att Gekås är drygt 40.000 kvm stort så valde vi int då smallint inte räcker till. |

## OpeningHours

| | | |
|---|---|---|
| ID | int | |
| DayOfWeek | nvarchar(9) | Längsta dagen i veckan är 9 tecken långt (Wednesday) |
| Opening | time(0) | För att slippa milisekunder osv. Vi vill ha formatet - hh:mm:ss |
| Closing | time(0) | Samma som ovan |
| StoreID | int | Koppling till Store (Butik) |

**Constraints**

- **Indexes/Keys:** DayOfWeek och StoreID är Unique och kan inte dupliceras. Innebär att en och samma butik inte kan ha olika öppettider samma dag, Exempelvis kan inte butik 1 ha öppet från 08.00 – 22.00 en måndag och 07.00 – 23.00 nästa måndag.
- **Check:** Går enbart mata in engelska veckodagar (Monday – Sunday).

## Schedule

| | | |
|---|---|---|
| StartTime | time(0) | För att slippa milisekunder osv. Vi vill ha formatet - hh:mm:ss |
| EndTime | time(0) | Samma som ovan |
| DayOfWeek | nvarchar(9) | Längsta dagen i veckan är 9 tecken långt (Wednesday) |
| EmployeeID | int | Koppling till Employee tabellen |
| StoreID | int | Koppling till Store tabellen |

**Constraints**

- **Indexes/Keys:** DayOfWeek och EmployeeID är Unique och kan inte dupliceras. Innebär att en och samma anställd inte kan jobba i två olika butiker samma dag, Exempelvis kan inte anställd 1 arbeta på två olika platser under en och samma dag. Om anställd 1 är schemalagd en måndag så blir just denna schemaläggning UNIQUE.
- **Check:** Går enbart mata in engelska veckodagar (Monday – Sunday).

## StoreManager

| | | |
|---|---|---|
| StoreID | int | Koppling till Store tabellen |
| EmployeeID | int | Koppling till Employee tabellen |

**Constraints**

- **Indexes/Keys:** StoreID och EmployeeID är Unique. Detta eftersom att varje butik endast kan ha en butikschef. Butikschefen kan är en anställd. En butikschef kan dock jobba i flera olika butiker
- **Övrig kommentar:** Tabellen agerar kopplingstabell (Join-tabell) mellan Employee och Store tabellen.

## Employee

| | | |
|---|---|---|
| ID | int | |
| FirstName | nvarchar(50) | |
| LastName | nvarchar(50) | |

## Discount

| ID | Int | |
|---|---|---|
| DiscountFixed | decimal(7, 2) | Eftersom Price i Product tabellen är satt till samma datatyp så har vi tänkt exempelvis att om en produkt kostar 99000,00 kr så ska vi kunna sätta en rabatt på 98000,00 kr. Är Nullable då det inte alltid finns en rabatt. |
| DiscountPercent | decimal(3, 2) | Matas in i detta format: 0,01 – 0,99. Vilket ska motsvara 1% - 99%. Denna beräkning görs senare i GUI eller direkt i Query. Är Nullable då det inte alltid finns en rabatt. |

**Constraints**

- **Check:** DiscountPercent > 0.00 AND DiscountPercent < 1.00. Vi ska inte kunna sätta en rabatt på mindre än 1% eller större än 99%.

## OrderProduct

| ProductID | int | Koppling till Product tabellen |
|---|---|---|
| OrderID | int | Koppling till Order tabellen |
| Quantity | tinyint | Tinyint eftersom vi anser att en slutkund inte handlar mer än 255 exemplar av samma vara. |

**Övrig kommentar:**

- Agerar kopplingstabell (Join-tabell) mellan Order och Product tabellen. Detta så att vi kan koppla en order med produkt och antal beställda produkter. Vi har inte valt att ha en tabell som heter Webshop just av den anledning att det endast finns en enda Webshop per butikskedja. Vilket innebär att vi inte kommer arbeta med mer än en enda Webshop i det här fallet. Därav låter vi OrderProduct agera som en ”Webshop” tabell. Vi anser även att enkelhet är smidigt i form av att vi inte behöver lägga till ”onödiga” tabeller då primärsyftet / funktionen upprätthålls.

## Order

| | | |
|---|---|---|
| ID | int | |
| CustomerID | int | Koppling till Customer tabellen |
| DiscountOrderID | int | Koppling till DiscountOrder tabellen. Denna är Nullable då det inte alltid används en rabattkod. |
| OrderDate | date | Vi valde inte datetime för enkelhetens skull. Vi tycker inte att tidpunkt är väsentligt. Det räcker med datum. |

**Constraints**

- **Indexes/Keys:** CustomerID och DiscountOrderID är Unique. Vi har valt att göra dessa unika så att en rabattkod inte kan användas flera gånger av samma kund. Det vill säga att varje kund endast kan använda en rabattkod en gång.

## DiscountOrder

| | | |
|---|---|---|
| ID | int | |
| DiscountCode | nvarchar(20) | |
| DiscountPercent | decimal(3, 2) | Matas in i detta format: 0,01 – 0,99. Vilket ska motsvara 1% - 99%. Denna beräkning görs senare i GUI eller direkt i Query. |

**Constraints**

- **Check:** DiscountPercent > 0.00 AND DiscountPercent < 1.00. Vi ska inte kunna sätta en rabatt på mindre än 1% eller större än 99%.
- **Check:** LEN([DiscountCode]) >= 6. En rabattkod måste vara minst 6 tecken långt. I och med datatypen på nvarchar(20) så har vi max 20 tecken långt

# Customer

| | | |
|---|---|---|
| ID | int | |
| FirstName | nvarchar(50) | |
| LastName | nvarchar(50) | |
| PersonalNumber | bigint | Matas in i detta format: yymmddxxxx (991231 0000). Därav varför vi valt bigint eftersom en vanlig int inte räcker till. |
| RegistrationDate | date | Vi valde inte datetime för enkelhetens skull. Vi tycker inte att tidpunkt är väsentligt. Det räcker med datum. |
| MembershipLevel | nvarchar(6) | Max 6 tecken långt eftersom vi använder bronze – gold. En ny kund börjar på Bronze. |
| FavoriteStoreID | int | Koppling till Store tabell för att välja favoritbutik. Är Nullable. Kund kanske inte har någon favoritbutik. |

**Constraints**

- **Indexes/Keys:** PersonalNumber kolumnen är Unique. Detta eftersom vi inte vill att en användare ska kunna skapa flera olika konton för att exempelvis komma förbi rabattkodsregeln där endast en rabattkod får användas en gång per användare.
- **Check:** MembershipLevel tar endast emot strängarna: ’Bronze’, ’silver’ och ’gold’.

# QUERY 1

```sql
--FIXED DISCOUNT ON CHOSEN PRODUCT (QUERY 1)
SELECT
    Product.[Name], Product.[Price],
    Discount.DiscountFixed AS [Discount (kr)],
    Product.[Price] - Discount.DiscountFixed AS [Price after discount]
FROM Product
JOIN Discount ON Discount.ID = Product.DiscountID
WHERE Discount.DiscountFixed IS NOT NULL

--DISCOUNT % ON CHOSEN PRODUCT
SELECT
    Product.[Name], Product.[Price],
    Discount.DiscountPercent * 100 AS [Discount (%)],
    SUM(Product.Price - (Product.Price * Discount.DiscountPercent)) AS [Price after discount]
FROM Product
JOIN Discount ON Discount.ID = Product.DiscountID
WHERE Discount.DiscountPercent IS NOT NULL
GROUP BY Product.[Name], Product.[Price], Discount.DiscountPercent
```

100 %

Results | Messages

| | Name | Price | Discount (kr) | Price after discount |
|---|---|---|---|---|
| 1 | Lätta Original | 15.50 | 14.50 | 1.00 |
| 2 | Lättmajonäs | 21.50 | 10.00 | 11.50 |
| 3 | Mellanmjölk | 9.25 | 8.00 | 1.25 |

| | Name | Price | Discount (%) | Price after discount |
|---|---|---|---|---|
| 1 | Hamburgerbröd | 31.50 | 30.00 | 22.0500 |
| 2 | Lätta Extrasaltat | 16.95 | 50.00 | 8.4750 |
| 3 | Standardmjölk | 13.50 | 20.00 | 10.8000 |

# QUERY 2

```sql
--Store and Opening Hours, Query 2
SELECT
    Store.[Name] AS [Store], Store.StreetName,
    OpeningHours.[DayOfWeek] AS [Day of Week],
    OpeningHours.Opening AS Opens,
    OpeningHours.Closing AS Closes
FROM Store
JOIN OpeningHours ON OpeningHours.StoreID = Store.ID
ORDER BY Store.StreetName,
CASE
    WHEN OpeningHours.[DayOfWeek] = 'Monday' THEN 0
    WHEN OpeningHours.[DayOfWeek] = 'Tuesday' THEN 1
    WHEN OpeningHours.[DayOfWeek] = 'Wednesday' THEN 2
    WHEN OpeningHours.[DayOfWeek] = 'Thursday' THEN 3
    WHEN OpeningHours.[DayOfWeek] = 'Friday' THEN 4
    WHEN OpeningHours.[DayOfWeek] = 'Saturday' THEN 5
    WHEN OpeningHours.[DayOfWeek] = 'Sunday' THEN 6
END
```

100 %

Results | Messages

| | Store | StreetName | Day of Week | Opens | Closes |
|---|---|---|---|---|---|
| 1 | Lilla Coop | Ekgatan | Monday | 09:00:00 | 20:00:00 |
| 2 | Lilla Coop | Ekgatan | Tuesday | 09:00:00 | 20:00:00 |
| 3 | Lilla Coop | Ekgatan | Wednesday | 10:00:00 | 18:00:00 |
| 4 | Lilla Coop | Ekgatan | Thursday | 11:00:00 | 19:00:00 |
| 5 | Lilla Coop | Ekgatan | Friday | 09:00:00 | 18:00:00 |
| 6 | Lilla Coop | Ekgatan | Saturday | 12:00:00 | 18:00:00 |
| 7 | Lilla Coop | Ekgatan | Sunday | 14:00:00 | 17:00:00 |
| 8 | Stora Coop | Munkvägen | Monday | 09:00:00 | 21:00:00 |
| 9 | Stora Coop | Munkvägen | Tuesday | 09:00:00 | 21:00:00 |
| 10 | Stora Coop | Munkvägen | Wednesday | 09:00:00 | 21:00:00 |
| 11 | Stora Coop | Munkvägen | Thursday | 09:00:00 | 21:00:00 |
| 12 | Stora Coop | Munkvägen | Friday | 09:00:00 | 21:00:00 |
| 13 | Stora Coop | Munkvägen | Saturday | 09:00:00 | 22:00:00 |
| 14 | Stora Coop | Munkvägen | Sunday | 06:00:00 | 23:00:00 |

# QUERY 3

```sql
SELECT
    [Order].OrderDate, Customer.FirstName + ' ' + Customer.LastName AS [Customer],
    SUM(OrderProduct.Quantity) AS ProductCount,
    SUM(Product.Price * OrderProduct.Quantity) AS [Price before discount],
    OrderProduct.OrderID
FROM Customer
    JOIN [Order] ON [Order].CustomerID = Customer.ID
    JOIN OrderProduct ON OrderProduct.OrderID = [Order].ID
    JOIN Product ON Product.ID = OrderProduct.ProductID
GROUP BY [Order].OrderDate, Customer.FirstName + ' ' + Customer.LastName, OrderProduct.OrderID
ORDER BY [Order].OrderDate DESC
```

100 %

Results | Messages

| | OrderDate | Customer | ProductCount | Price before discount | OrderID |
|---|---|---|---|---|---|
| 1 | 2020-12-15 | Patric Bergkvist | 10 | 152.80 | 1 |
| 2 | 2020-12-05 | Orhan Albayati | 113 | 2302.35 | 2 |

# QUERY 4 (Bonus, Webborder med discount)

```sql
SELECT
    [Order].OrderDate, Customer.[FirstName],
    SUM(OrderProduct.Quantity) AS ProductCount,
    SUM(Product.Price * OrderProduct.Quantity) AS [Price before discount],
    SUM((Product.Price * OrderProduct.Quantity) -
    (Product.Price * OrderProduct.Quantity * DiscountOrder.DiscountPercent)) AS [Price after discount],
    DiscountOrder.DiscountPercent * 100 AS [Discount Percentage],
    OrderProduct.OrderID
FROM Customer
JOIN [Order] ON [Order].CustomerID = Customer.ID
JOIN OrderProduct ON OrderProduct.OrderID = [Order].ID
JOIN DiscountOrder ON DiscountOrder.ID = [Order].ID
JOIN Product ON Product.ID = OrderProduct.ProductID
GROUP BY [Order].OrderDate, Customer.[FirstName], OrderProduct.OrderID, DiscountOrder.DiscountPercent
ORDER BY [Order].OrderDate DESC
```

100 %

Results | Messages

| | OrderDate | FirstName | ProductCount | Price before discount | Price after discount | Discount Percentage | OrderID |
|---|---|---|---|---|---|---|---|
| 1 | 2020-12-15 | Patric | 10 | 152.80 | 76.4000 | 50.00 | 1 |
| 2 | 2020-12-05 | Orhan | 113 | 2302.35 | 1726.7625 | 25.00 | 2 |