cphbusiness

SQL

Nøgler

- Primærnøgler
- Fremmednøgler
- Sammensatte nøgler

Hvad bruger vi nøgler til? Hvilke krav er der til nøgler?

Count

De to queries giver samme resultat. Det er fordi SupplierID ikke er NULL i nogen af rækkerne. Når vi kvalificerer COUNT med en specifik kolonne og ikke *, siger vi, at vi kun vil have rækker med hvor den værdi ikke er NULL.

```
SELECT count(*)
FROM Products;
```

```
SELECT count(SupplierID)
FROM Products;
```

Result:

Number of Records: 77

ProductID	ProductName	SupplierID
1	Chais	1
2	Chang	1
3	Aniseed Syrup	1

Concat as

- Hvis vi har flere kolonner, vi gerne vil slå sammen til én i resultatet (fx fornavn og efternavn eller navn og adresse), kan vi bruge CONCAT AS

```
SELECT CONCAT (Customername, Address) FROM Customers
```

Result:

Number of Records: 91

CONCAT (Customername, Address)
Alfreds FutterkisteObere Str. 57
Ana Trujillo Emparedados y heladosAvda. de la Constitución 2222
Antonio Moreno TaqueríaMataderos 2312

```
SELECT CONCAT (Customername, Address) AS info FROM Customers
```

Result:

Number of Records: 91

info
Alfreds FutterkisteObere Str. 57
Ana Trujillo Emparedados y heladosAvda. de la Constitución 2222
Antonio Moreno TaqueríaMataderos 2312

Joins

Vores datamodellering med flere tabeller gør, at vi ofte har brug for at læse fra mere end én tabel af gangen. Det kaldes en join-forespørgsel:

```
SELECT Order.OrderID, Customer.CustomerName, Order.OrderDate
FROM Order
INNER JOIN Customer ON Order.CustomerID=Customer.CustomerID
WHERE Order.CustomerID = '11'
```

Her står, at vi gerne vil have felterne OrderID (fra tabellen Order), CustomerName (fra tabellen Customer) og OrderDate (fra tabellen Order) og at det skal gælde om de udvalgte rækker, at CustomerID på Order og Customer er ens og begge felter har værdien '11'.

Mere overskueligt (måske)

```sql
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
FROM Orders
INNER JOIN Customers
ON Orders.CustomerID=Customers.CustomerID;
```

```sql
SELECT o.OrderID, c.CustomerName, o.OrderDate
FROM Orders o
INNER JOIN Customers c
ON o.CustomerID=c.CustomerID;
```

USING clause

Hvis kolonnerne, vi bruger, hedder det samme, kan vi anvende USING i stedet for at skrive kolonnens navn to gange:

```
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
FROM Orders
INNER JOIN Customers
ON Orders.CustomerID=Customers.CustomerID;
```

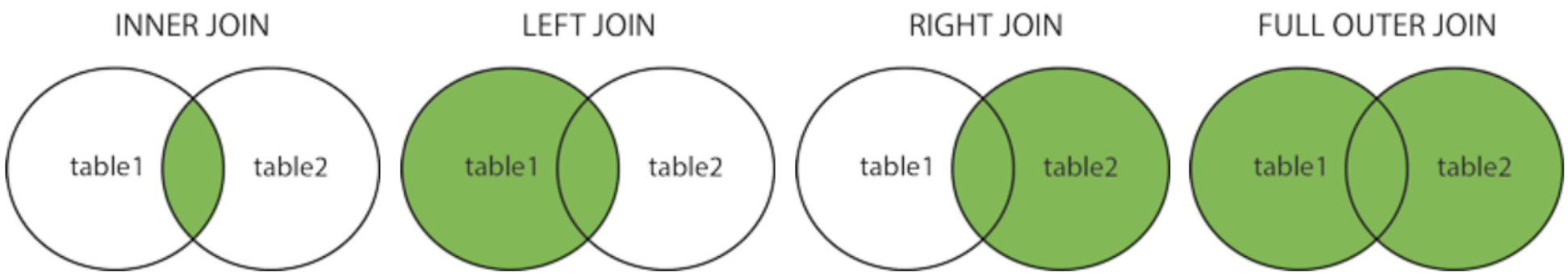
```
SELECT Orders.OrderID, Customers.CustomerName
FROM Orders
INNER JOIN Customers USING (CustomerID)
```

Typer af joins

Different Types of SQL JOINs

Here are the different types of the JOINs in SQL:

- `(INNER) JOIN`: Returns records that have matching values in both tables
- `LEFT (OUTER) JOIN`: Returns all records from the left table, and the matched records from the right table
- `RIGHT (OUTER) JOIN`: Returns all records from the right table, and the matched records from the left table
- `FULL (OUTER) JOIN`: Returns all records when there is a match in either left or right table



cphbusiness

Views – et udsnit af en database

```
CREATE VIEW italian_customers AS SELECT customernumber, customername, phone FROM customers
```



```
SELECT customername FROM italian_customers;
```