SQL – Structured Query Language

Hinweis: Einige Funktionen haben bei den verschiedenen Datenbanken eine unterschiedliche Syntax. Nachfolgend wird die Syntax für MS SQL Server verwendet.

SELECT	Wähle die Felder …
FROM	aus Tabelle/n …
WHERE	unter der Bedingung, dass…
GROUP BY	Gruppiere nach den Feldern …
ORDER BY	Sortiere nach den Feldern …

SELECT-Anweisung

Hinter SELECT gibt man die Felder an, die angezeigt werden sollen. Werden mehrere (miteinander verknüpfte) Tabellen abgefragt, sollte den Feldnamen der Tabellenname oder ein Alias (z.B. ein Buchstabe, siehe Hinweise zur FROM-Anweisung unten) vorangestellt werden, die Trennung zwischen Tabellenname/Alias und Feldname ist ein Punkt, zwischen den einzelnen Feldern ein Komma:

```
SELECT n_vorgang.aktenzeichen, n_adressen.nachname,
n_adressen.vorname
```

oder

```
SELECT    v.aktenzeichen, a.nachname, a.vorname
```

Es empfiehlt sich zur späteren Nachvollziehbarkeit, die einzelnen Feldnamen strukturiert in die SQL-Abfrage einzugeben, z.B. untereinander statt hintereinander und eingerückt (ggf. mit der Tabulator-Taste oder Leerzeichen arbeiten):

```
SELECT    v.aktenzeichen,
          a.nachname,
          a.vorname
```

Die Feldnamen können mit eigenen Bezeichnungen (Spaltenüberschriften für das Auswertungsergebnis) versehen werden:

```
SELECT    v.aktenzeichen      as "Az",
          a.nachname          as "Nachname junger Mensch",
          a.vorname           as "Vorname junger Mensch"
```

FROM-Anweisung

Hinter FROM werden die Tabellen angegeben, die man für die Abfrage benötigt (aus denen die im SELECT-Teil aufgeführten Felder kommen).

Wenn mehrere Tabellen abgefragt werden müssen, muss man angeben, wie jeweils zwei Tabellen miteinander verknüpft sind, d.h. welche Felder in beiden Tabellen den gleichen Inhalt haben.

Dabei kommt es darauf an, ob in beiden Tabellen immer ein Datensatz steht oder ggf. nur in einer der beiden Tabellen und in der anderen könnte ein Datensatz stehen.

Wenn immer in beiden Tabellen Datensätze vorhanden sind, wird zur Verknüpfung der beiden Tabellen der INNER JOIN verwendet.

Beispiel:
Liste der Vorgänge mit Bereichsbezeichnung

In der Tabelle n_vorgang steht nur die Bereichsnummer im Feld "bereich". Die Bereichsbezeichnung steht in der Tabelle bere. Dort steht im Feld "brs" ebenfalls die Bereichsnummer und im Feld "bb" die Bereichsbezeicchnung.

Um nun beide Tabellen zu verknüpfen, muss man zunächst wissen, dass in PROSOZ 14plus kein Vorgang ohne eingetragene Bereichsnummer erfasst werden kann. Insofern steht fest, dass es zwischen Datensätzen in diesen beiden Tabellen immer eine Verbindung gibt, nämlich:

```
n_vorgang.bereich = bere.brs
```

Insofern lautet die Verknüpfung in der FROM-Anweisung

```
FROM n_vorgang INNER JOIN bere ON n_vorgang.bereich = bere.brs
```

Kann es hingegen sein, dass in einer der beiden Tabellen Daten vorhanden sind, in der anderen Tabelle aber nicht zwingend, dann muss der LEFT JOIN verwendet werden, wobei die "linke" Tabelle immer Daten enthält, die "rechte" aber nur manchmal.

Beispiel:
Liste aller Personen mit evtl. erfasstem Geburtsdatum

Der Name und Vorname von Personen steht in der Tabelle n_adressen. Das Geburtsdatum steht in der Tabelle n_personendaten. Im Feld "n_adressen.adressnummer" steht eine eindeutige Nummer jede erfassten Person. Im Feld "n_personendaten.zuordnungsnummer" steht die identische Nummer, wenn Personendaten (also z.B. das Geburtsdatum) erfasst wurden. Andernfalls gibt es zu dem Datensatz in n_adressen keinen korrespondierenden Datensatz in n_personendaten.

Da nicht zu jeder Person in PROSOZ 14plus zwingend ein Geburtsdatum erfasst werden muss (dies ist nur bei jungen Menschen und Geschwistern der Fall), gibt es daher zwar immer einen Datensatz in der Tabelle n_adressen, aber nur manchmal einen Datensatz in n_personendaten.

Um trotzdem alle Namen aufzulisten, gilt es folgende Überlegung anzustellen:

Würde man zur Verknüpfung der beiden Tabellen den INNER JOIN verwenden, würde die Abfrage nur die Datensätze ausgeben, bei denen ein Geburtsdatum erfasst ist (also in beiden Tabellen ein korrespondierender Datensatz vorhanden ist). Personen ohne Geburtsdatum würden dann im Ergebnis nicht auftauchen.

Um alle Adressen (mit und ohne Personendaten) zu erhalten, lautet die Verbindung zwischen diesen Tabellen daher

```
FROM n_adressen.adressnummer LEFT JOIN n_personendaten
     ON   n_adressen.adressnummer =
          n_personendaten.zuordnungsnummer
```

Die Tabelle n_adressen ist die "linke" Tabelle (es gibt immer einen Datensatz) und die Tabelle n_personendaten die "rechte" Tabelle (es gibt nur manchmal einen Datensatz).

Generell gilt für die FROM-Anweisung außerdem:

Es müssen auch Tabellen angegeben werden, deren Felder nicht im Ergebnis angezeigt werden, sofern es sich um Verknüpfungstabellen handelt (hier z.B. n_vorgbeteiligte, weil in dieser die Zuordnung der Datensätze aus n_adressen zum Vorgang gespeichert ist):

```
FROM n_vorgang
     INNER JOIN n_vorgbeteiligte
     ON n_vorgang.vorgangsnummer = n_vorgbeteiligte.vorgang
     INNER JOIN n_adressen
```

```
ON n_vorgbeteiligte.adresse = n_adressen.adressnummer
```

Wenn Sie für die Tabellenbezeichnungen Aliase verwenden (z.B. Buchstaben), müssen diese hinter den Tabellennamen getrennt durch ein Leerzeichen definiert werden und können dann im gesamten Script als Buchstabe vor dem jeweiligen Feldnamen (getrennt durch einen Punkt) gesetzt werden, um anzugeben, aus welcher Tabelle das jeweilige Feld stammt.

Insofern lautet die Anweisung insgesamt

```
SELECT    v.aktenzeichen,
          a.nachname,
          a.vorname

FROM      n_vorgang v

          INNER JOIN n_vorgbeteiligte b
          ON v.vorgangsnummer = b.vorgang

          INNER JOIN n_adressen a
          ON b.adresse = a.adressnummer
```

WHERE-Anweisung

Hinter WHERE werden alle inhaltlichen Bedingungen (Kriterien / Filter) angegeben, denen die ausgewerteten Daten entsprechen müssen.

Es ist zu beachten, dass Felder mit Datum und Text bei den Kriterien in einfachen Hochkommata und Felder mit numerischen Werten ohne Hochkomma angegeben werden.

Beispiele:

```
a.ort = 'München'
p.geburtsdatum = '25.10.2008'
year(p.geburtsdatum) > 25
```

Sollen mehrere Bedingungen gleichzeitig zutreffen, werden sie mit AND nacheinander aufgeführt. Vor der ersten Bedingung steht die WHERE-Anweisung.

Beispiel (nur weibliche Personen aus München):

```
WHERE     a.ort = 'München'
AND       p.geschlecht = 'w'
```

Sollen hingegen mindestens eine von mehreren Bedingungen zutreffen, dann werden sie mit OR aufgeführt und die gesamte Bedingung (also das Argument vor dem OR und das Argument nach dem OR) muss eingeklammert werden:

Beispiel (weibliche oder männliche Personen aus München):

```
WHERE     a.ort = 'München'
AND       ( p.geschlecht = 'w' OR p.geschlecht = 'm' )
```

ORDER BY-Anweisung

ORDER BY sortiert die Datensätze nach den dahinter angegebenen Feldern. Diese werden analog der SELECT-Anweisung angegeben, wobei hier keine eigenen Spaltenüberschriften angegeben werden dürfen:

```
ORDER BY v.aktenzeichen, a.nachname, a.vorname
```

Man kann die Felder, nach denen sortiert werden soll, auch mit einer Zahl (Spaltennummer des Auswertungsergebnisses) angeben.

Steht im SELECT-Teil beispielsweise:

```
SELECT    v.aktenzeichen,
          a.nachname,
          a.vorname
```

kann man als Sortierungsanweisung angeben:

```
ORDER BY 2, 3
```

Das Ergebnis wird dann zuerst nach Nachname, dann nach Vorname sortiert.

Standardmäßig wird immer aufsteigend sortiert. Wird hinter den Spaltenname und die Spaltennummer jedoch der Begriff DESC geschrieben, wird stattdessen absteigend sortiert:

```
ORDER BY 2 DESC, 3
```

Sortiert absteigend nach Nachname, danach aufsteigend nach Vorname

GROUP BY

Bei mathematischen Funktionen in der SELECT-Anweisung (count, sum, min, max, avg) müssen Sie nach der WHERE-Anweisung mit dem Befehl GROUP BY nach allen anderen Feldern aus dem SELECT-Teil (die keine mathematische Funktion enthalten) gruppieren.

Individuelle Spaltenüberschriften dürfen in der GROUP BY-Anweisung nicht enthalten sein.

Beispiel:
Anzahl der Vorgänge pro Bereich (bis inkl. Bereich 4000)

```
SELECT    v.bereich as "Bereich",
          br.bb as "Bereichsname",
          count(v.vorgangsnummer) as "Anzahl Vorgänge"

FROM      n_vorgang v inner join bere br
          ON v.bereich = br.brs

WHERE     v.bereich < 4000

GROUP BY  v.bereich,
          br.bb

ORDER BY 1
```

SQL – Structured Query Language

Funktionen

In SQL-Anweisungen gibt es eine Vielzahl von Funktionen, mit denen man z.B. rechnen oder Feldausgaben formatieren kann.
Der Feldname muss hinter bzw. in die Funktion in Klammern eingegeben werden.

Nachfolgend sind einige Beispiele aufgeführt.

Beschreibung	**MS SQL Server**	**ORACLE**
Jahr (als Zahl) aus dem Geburtsdatum	`year(geburtsdatum)`	`to_number(to_char(geburtsdatum,'yyyy'))`
Monat (als Zahl) aus dem Geburtsdatum	`month(geburtsdatum)`	`to_number(to_char(geburtsdatum,'mm'))`
Tag (als Zahl) aus dem Geburtsdatum	`day(geburtsdatum)`	`to_number(to_char(geburtsdatum,'dd'))`
Geburtsdatum (als Text) in deutscher Schreibweise	`convert(char(10),geburtsdatum,104)`	`to_char(geburtsdatum,'dd.mm.yyyy')`
Anzahl (Zählen) der Vorgänge	`count(vorgangsnummer)`	`count(vorgangsnummer)`
Anzahl (Zählen) der verschiedenen jungen Menschen	`count(distinct muendelnr_alt)`	`count(distinct muendelnr_alt)`
Summe des Betrags	`sum(betrag)`	`sum(betrag)`
Kleinster Betrag	`min(betrag)`	`min(betrag)`
Größter Betrag	`max(betrag)`	`max(betrag)`
Durchschnittlicher Betrag	`avg(betrag)`	`avg(betrag)`

Beschreibung	**MS SQL Server**	**ORACLE**
Ganzzahl (Zahl vor dem Komma) des Betrags	`floor(betrag)`	`floor(betrag)`
Nachkommazahl	`ceiling(betrag)`	`ceiling(betrag)`
Runden des Betrags auf 2 Nachkommastellen	`round(betrag,2)`	`round(betrag,2)`
Verketten von Texten	`nachname + ', ' + vorname`	`nachname || ', ' || vorname`
Teil eines Textes (vier Zeichen ab der 2. Stelle)	`substring(nachname,2,4)`	`substr(nachname,2,5)`
Unterscheidung Groß- und Kleinbuchstaben, hier A und a (siehe Tabelle ASCII-Codes)	`ascii(typ) = 65` `ascii(typ) = 97`	`typ = 'A'` `typ = 'a'`
Stelle des Paragrafzeichens im Textfeld ab der 1. Stelle suchen	`charindex('§',textfeld,1)`	`INSTR(textfeld,'§',1);`