

**Ihr Einstieg in SQL  
an der HTBLVA Wien 5  
mit mehr als 100 praktischen Beispielen**

© 12/2011, HTBLVA Wien 5, Andrea Voit und Hans Preissl

## INHALTSVERZEICHNIS

I. ZIELE	2
Zielgruppe	2
Zielsetzung	2
Nicht-Ziele	2
Mitgeltende Dokumente	2
II. UMGEBUNG	2
III. SQL BEFEHLE IM ÜBERBLICK	2
Das Konzept von SQL	2
SQL ist mehr als eine Abfragesprache	2
DDL Befehle	2
Anlegen einer Tabelle (CREATE TABLE)	2
Anlegen eines Index (CREATE INDEX)	2
Anlegen eines Synonyms (CREATE SYNONYM)	2
Anlegen einer View (CREATE VIEW)	2
Verändern von Tabellenstrukturen (ALTER TABLE)	2
Löschen einer Tabelle (DROP TABLE)	2
Löschen eines Index (DROP INDEX)	2
Löschen eines Synonyms (DROP SYNONYM)	2
Löschen einer View (DROP VIEW)	2
DML Befehle	2
Lesen von Datensätzen aus einer/mehreren Tabelle/n (SELECT)	2
Einfügen von Datensätzen in eine Tabelle (INSERT)	2
Das Bearbeiten von Datensätzen in einer Tabelle (UPDATE)	2
Das Löschen von Datensätzen aus einer Tabelle (DELETE)	2
Befehle für die Verwaltung von Zugriffsrechten	2
Die Vergabe von Zugriffsrechten	2
Der Entzug von Zugriffsrechten	2
Befehle zur Transaktionssteuerung	2
Commit einer Transaktion (COMMIT)	2
Rollback einer Transaktion (ROLLBACK)	2
IV. DAS SELECT	2
Auswahl von Spalten	2
Beispiel 1: Ausgabe aller Räume	2
Beispiel 2: Ausgabe aller Langbezeichnungen der Gegenstände	2
Beispiel 3: Ausgabe aller Prüflinge mit Prüfungsfach und Prüfungsdatum	2
Arbeiten mit Bedingungen	2
Beispiel 4: Schüler mit einer Schülernummer kleiner 20	2
Beispiel 5: Schüler mit einer Schülernummer gleich 2222	2
Beispiel 6: Lehrer mit einem Gehalt < 500 und > 200 (mit AND)	2
Beispiel 7: Lehrer mit einem Gehalt < 500 und > 200 (mit BETWEEN)	2
Beispiel 8: Lehrer Gehalt < 500, > 200 und Geburtsdatum vor 1.1.1955	2

	3
Beispiel 9: Lehrer mit Gehalt größer 250 oder Geburtsdatum vor 1.1.1955	3
Beispiel 10: Schüler mit Wohnort Hernals	3
Beispiel 11: Schüler mit Wohnort Hernals oder Mödling	3
Beispiel 12: Schüler der 03TA mit Wohnort Hernals	3
Beispiel 13: Direktoren	3
Beispiel 14: Untergebene Lehrer von Direktoren	3
Beispiel 15: Untergebene Lehrer von Nicht-Direktoren	3
Beispiel 16: Schüler mit einer 1 auf eine beliebige Prüfung	3
Beispiel 17: Schüler mit einer 1 in Englisch	3
Beispiel 18: Lehrer Gehalt 100 oder 220 und Geburtsdatum vor 1.1.1956 (mit / ohne Klammern)	3
Beispiel 19: Lehrer Gehalt 100 oder 220 und Geburtsdatum vor 1.1.1956 (mit / ohne Klammern)	3
Beispiel 20: Lehrer Gehalt 82, 220, 300 und Geburtsdatum vor 1.1.1970	3
Beispiel 21: Lehrer mit Geburtsdatum ungleich 12.12.1950	3
Beispiel 22: Lehrer Gehalt 82, 220, 300 und Geburtsdatum vor 1.1.1970	3
Qualifizierung	3
Beispiel 23: Schüler und deren Geburtstag (ohne Qualifizierung)	3
Beispiel 24: Schüler und deren Geburtstag (mit Qualifizierung)	3
Sortierung	3
Beispiel 25: Alphabetisch absteigend sortierte Schülerliste	3
Beispiel 26: Alphabetisch absteigend sortierte Schülerliste (etwas kürzer)	3
Beispiel 27: Zweifach sortierte Schülerliste	3
Alias für Spaltennamen	3
Beispiel 28: Gegenstandsausgabe mit einprägsamer Spaltenbezeichnung	3
Wildcards	3
Beispiel 29: Lehrer, deren Vorname Gustav lautet	3
Beispiel 30: Lehrer, deren Vorname mit M beginnt	3
Beispiel 31: Lehrernachnamen mit einem B an 1. und einem R an 3. Stelle	3
Beispiel 32: Klassen mit einer 0 an beliebiger Stelle	3
Duplikate	3
Beispiel 33: Stunden, die von PS unterrichtet werden (mit Duplikaten)	3
Beispiel 34: Stunden, die von PS unterrichtet werden (ohne Duplikate)	3
Beispiel 35: Gehälter der Lehrer (ohne Duplikate)	3
Nullwerte	3
Beispiel 36: Lehrer ohne Gehalt	3
Beispiel 37: Lehrer mit bekanntem Gehalt, aufsteigend sortiert	3
Konstante	3
Beispiel 38: Geniale Schüler	3
Aggregatfunktionen	3
Beispiel 39: Aggregierter Wert, Konstante und Ausdruck für Lehrer	3
Beispiel 40: Aggregationswert, Konstante, Ausdruck für Lehrer mit Chef HI	3
Beispiel 41: Rechnen mit dem Lehrer Gehalt	3
Beispiel 42: Geburtsdatum des jüngsten Schülers	3
Beispiel 43: Geburtsdatum des ältesten Schülers	3
Beispiel 44: Ausgabe des heutigen Datums	3
Beispiel 45: Rechnen mit den Geburtsdaten der Lehrer	3
Verbund (Join / Inner Join)	3
Beispiel 46: Schüler der Klasse 03TA	3
Beispiel 47: Schüler der Klasse 03TA mit Klassenbezeichnung	3
Beispiel 48: Stundenplan für Montag in der 03TA mit Gegenstandsnamen	3
Beispiel 49: Prüfungen, die der Klassenvorstand abhielt	3
Beispiel 50: Schüleranzahl der 03TB	3
Beispiel 51: Schüleranzahl der 03TB mit Klassenlangname	3
Beispiel 52: Sitzplätze der Lehrsäle und Labors der 03TA (mit Duplikaten)	3

Beispiel 53: Sitzplätze der Lehrsäle und Labors der 03TA (ohne Duplikate)	4
Alias für Tabellennamen	4
Beispiel 54: Lehrer mit Gehalt so hoch wie vom direkten Vorgesetzten	4
Beispiel 55: Lehrer mit Gehalt kleiner gleich dem des direkten Vorgesetzten	4
Beispiel 56: Liste aller Vorgesetztenbeziehungen	4
Datumswerte	4
Beispiel 57: Ausgabe der Schüler, die im Dezember Geburtstag haben	4
Beispiel 58: Ausgabe der Schüler, die am 3.12.1980 Geburtstag haben	4
Beispiel 59: Schüler, die jünger als Lehrer HA sind	4
Gruppierung	4
Beispiel 60: Schüleranzahl	4
Beispiel 61: Schüleranzahl je Klasse	4
Gruppeneinschränkung	4
Beispiel 62: Klassen mit mehr als 6 Schülern	4
Beispiel 63: Schüleranzahl je Jahrgang mit mehr als einem Schüler	4
Beispiel 64: Jahrgänge ab 1975 mit mehr als einem Schüler	4
Beispiel 65: Jahrgänge ab 1975	4
Beispiel 66: Vornamenhäufigkeit bei Lehrern (mit Nullwert)	4
Beispiel 67: Vornamenhäufigkeit bei Lehrern (ohne Nullwert)	4
Beispiel 68: Lehrsäle mit mehr als 5 Stunden Unterricht	4
Beispiel 69: Vorgesetzte Lehrer mit Durchschnittsgehalt ihrer Untergebenen	4
Beispiel 70: Vorgesetzte mit Durchschnittsgehalt > 150 der Untergebenen	4
Beispiel 71: Unterrichtsstunden je Lehrer	4
Beispiel 72: Lehrer, die mehr als 10 Stunden unterrichten	4
Beispiel 73: Stunden mit mehr Schülern als Sitzplätzen	4
Subselect	4
Beispiel 74: Geburtsdatum und Name des jüngsten Schülers	4
Beispiel 75: Schüler, die bereits Prüfungen absolviert haben (Subselect)	4
Beispiel 76: Schüler mit absolvierten Prüfungen (Join)	4
Beispiel 77: Klassensprecher	4
Beispiel 78: Klassensprecher der Klasse mit BA als Klassenvorstand	4
Beispiel 79: Klassensprecher (mit ANY)	4
Beispiel 80: Lehrer mit dem höchsten bekannten Gehalt (mit ALL)	4
Beispiel 81: Lehrer mit dem niedrigsten bekannten Gehalt	4
Beispiel 82: Klassensprecher und Klassensprecherstellvertreter (EXISTS)	4
Beispiel 83: Ungeprüfte Gegenstände	4
Beispiel 84: Unterrichtende, aber nicht prüfende Lehrer	4
Beispiel 85: Falsche Fremdschlüssel im Stundenplan	4
Beispiel 86: Lehrer mit nicht existierenden Vorgesetzten	4
Beispiel 87: Lehrer mit gleichem Gehalt	4
Beispiel 88: Lehrer mit dem zweithöchsten Gehalt	4
Outer Join	4
Beispiel 89: Notendurchschnitt aller Gegenstände	4
Beispiel 90: Notendurchschnitt aller Schüler	4
Mengenoperationen	4
Beispiel 91: Anzahl der Unterrichtsstunden aller Lehrer (UNION)	4
Beispiel 92: Vornamen aller Schüler und Lehrer (ohne Duplikate / UNION)	4
Beispiel 94: Alle Klassen vereinigt mit allen Räumen	4
Statistiken in SQL	4
Beispiel 95: Durchschnitt und Summe der Lehrergehältern	4
Beispiel 96: Anzahl unterschiedlicher Lehrergehälter	4
Beispiel 97: Ungewichteter Durchschnitt der Lehrergehälter	4
Beispiel 98: Modus der Lehrergehälter	4
Beispiel 99: Streubreite und arithmetisches Mittel der Lehrergehälter	4

Extraktion von Stringbestandteilen	5
Beispiel 100: Erster bis vierter Buchstabe alle Lehrervornamen	5
Beispiel 101: Letzter und vorletzter Buchstabe alle Lehrervornamen	5
Beispiel 102: Dritter bis vierter Buchstabe alle Lehrervornamen	5
 ANHANG A: INHALT ALLER TABELLEN DER SCHULDATENBANK	5
Tabelle gegenstaende	5
Tabelle klassen	5
Tabelle lehrer	5
Tabelle pruefungen	5
Tabelle raeume	5
Tabelle schueler	5
Tabelle stunden	5
Tabelle vorgesetzte	5
 ANHANG B: ÜBUNGSBEISPIELE	5
 ANHANG C: ANSÄTZE FÜR DIE ERWEITERUNG DES DATENMODELLS	5
 ANHANG D: FEEDBACK	5
 ANHANG E: RAUM FÜR IHRE NOTIZEN	5

# I. ZIELE

## A. Zielgruppe

Dieses Skriptum richtet sich an Schüler/innen der HTBLVA Wien 5, denen im Rahmen des Unterrichts SQL, die standardisierte Sprache zur Abfrage und Manipulation von relationalen Datenbanken, vorgestellt und näher gebracht wird.

Das folgende Wissen wird hierbei vorausgesetzt:

- Regeln für den Aufbau und die Auswertung von Bedingungen<sup>1</sup>

Hilfreich, aber nicht zwingend erforderlich, ist ein Wissen rund um die Datenmodellierung (mit ER<sup>2</sup>-Diagrammen) bzw. um das Datenbankdesign von relationalen Datenbanken.

## B. Zielsetzung

Jeder, der dieses Skriptum durchgearbeitet hat, sollte in der Lage sein,

- die Konzepte von SQL für die Gestaltung eigener Abfragen auf der Basis der im Unterricht zur Verfügung gestellten Schul-Datenbank im MS Access Format anzuwenden
- das SQL Wissen auf jeden anderen, beliebigen Sachverhalt, der in einer MS Access Datenbank abgebildet wird, umzulegen und ebendort anzuwenden

## C. Nicht-Ziele

Als Einführungs-Skriptum wird bewusst von zwar relevanten, aber für den Einstieg zu weit führenden datenbankspezifischen SQL Aspekten wie z.B.

- Transaktionen
- Zugriffsrechte
- Stored Procedures
- Trigger
- Embedded SQL

abstrahiert. Der Schwerpunkt wird auf das SELECT gelegt werden.

---

<sup>1</sup> unter Nutzung von Vergleichsoperatoren, logischen Operatoren et.al.

<sup>2</sup> Entity Relationship Diagramme / Modelle

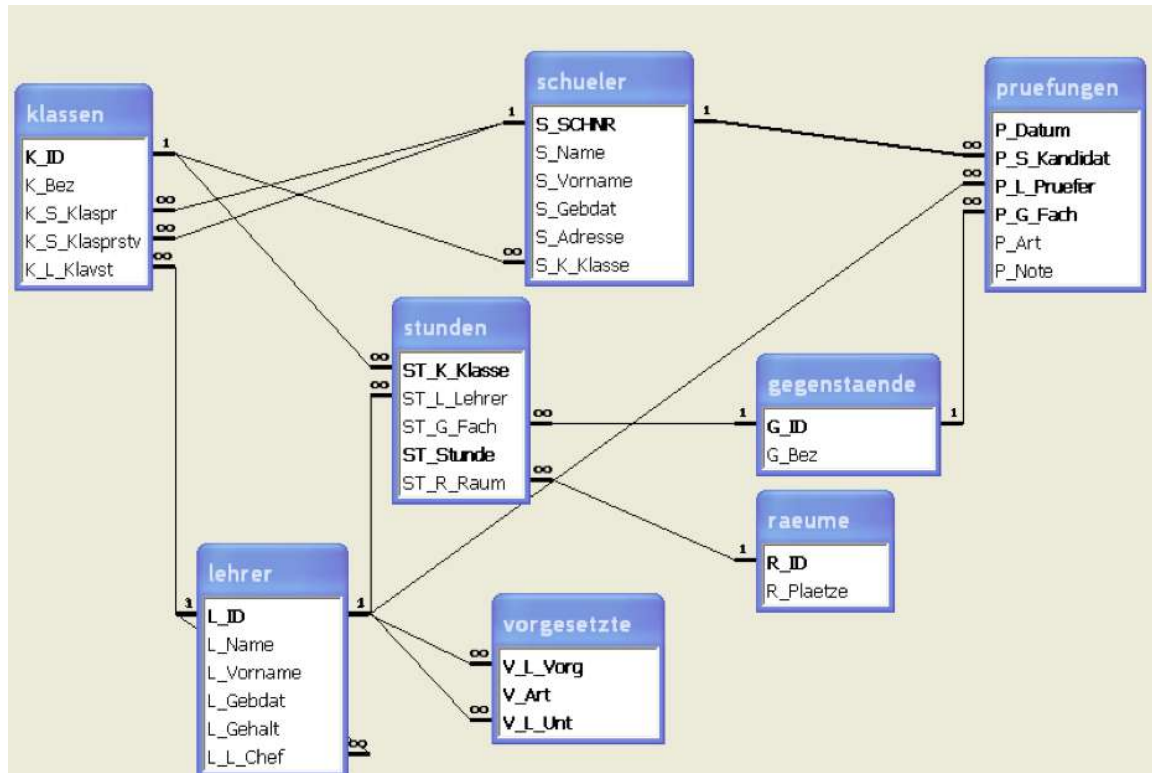
## **D. Mitgeltende Dokumente**

Die Handhabung von Access (z.B. Tabellen anlegen und Tabellen modifizieren, Daten importieren, Abfragen erstellen, et.al.) sind ebenso wie die wesentlichen Aspekte von SQL in den (im Zuge des Unterrichts) ausgehändigten Präsentationen beschrieben.

## II. UMGEBUNG

Die Schuldatenbank (Tabellen und deren Inhalt: siehe Anhang B) bildet den Ausgangspunkt für die SQL Statements.

Die Beziehungen zwischen den Tabellen dieser Datenbank sind wie folgt:





## I. III. SQL BEFEHLE IM ÜBERBLICK

### A. Das Konzept von SQL

SQL –

das S steht für Structured,  
das Q für Query und  
das L für Language,

ergibt gesamt Structured Query Language (dt: strukturierte Abfragesprache).

SQL ist eine normierte, mengenorientiert arbeitende, nicht-prozedurale Abfragesprache. Genauer gesagt, sie ist die einzige normierte Sprache zur Abfrage und Manipulation von relationalen Datenbanken.

- Die Normen und Standards stammen von ANSI<sup>3</sup> und ISO<sup>4</sup>. Die einzelnen Datenbank-Anbieter (wie Oracle, Microsoft, et.al.) orientieren sich an den durch die Normen vorgegebenen Rahmenbedingungen. Gleichzeitig grenzen sie zumeist ihr eigenes Produkt durch zusätzliche, proprietäre Komponenten von den Konkurrenzprodukten ab.
- Die Mengenorientierung von SQL spiegelt sich darin wider, dass die Befehle so ausgelegt sind, dass Daten in Mengen verarbeitet werden (und nicht als einzelne Datensätze).
- Die Nicht-Prozeduralität ist Ausdruck des Konzepts, nicht das ‚Wie‘, sondern das ‚Was‘ in den Vordergrund zu stellen.  
Ein SQL Statement beschreibt, welche Daten einzufügen, zu modifizieren, abzufragen oder zu löschen sind, abstrahiert jedoch von der Art und Weise, wie dies vorgenommen werden soll.

---

<sup>3</sup> American National Standards Institute

<sup>4</sup> International Organization for Standardization

## **B. SQL ist mehr als eine Abfragesprache**

Die Aussage 'Sprache zur Abfrage und Manipulation von relationalen Datenbanken' lässt bereits erahnen, dass SQL in mehr als einem Gebiet zum Einsatz kommen kann, nämlich

- DDL Befehle zur Festlegung des Tabellenaufbaus
- DML Befehle zur Bearbeitung der in den Tabellen abgelegten Daten
- Befehle zur Verwaltung von Rechten an einer Datenbank und deren Elemente
- Befehle zum Handling von Transaktionen

## **c. DDL Befehle**

DDL steht für DATA DEFINITION LANGUAGE und umfasst alle Befehle, die erforderlich sind, um Tabellen und deren Elemente in ihrer Struktur zu definieren, zu verändern und zu löschen.

Zu den DDL Befehlen zählen

- CREATE zum Anlegen einer Tabelle / eines Index / eines Synonyms / einer View
- ALTER zum Modifizieren
- DROP zum Entfernen

## 1. Anlegen einer Tabelle (CREATE TABLE)

### Eine Tabelle

- ist die Struktur, die den Rahmen für die Speicherung von Daten darstellt
- zeichnet sich durch einen innerhalb der Datenbank eindeutigen Namen aus
- besteht aus 1 bis n Spalten (Attributen / Feldern) und aus 0 bis n Zeilen (Datensätzen / Records)
- hat einen natürlichen oder einen künstlichen, einen einfachen oder einen zusammengesetzten Primärschlüssel (PK)
- kann über 0 bis n Fremdschlüssel (FK) verfügen

Die 'einfache' Syntax lautet:

```
CREATE TABLE tabellenname (spaltenname datentyp [NOT NULL])
```

tabellenname ist der frei zu vergebene Name für die zu erstellende Tabelle.

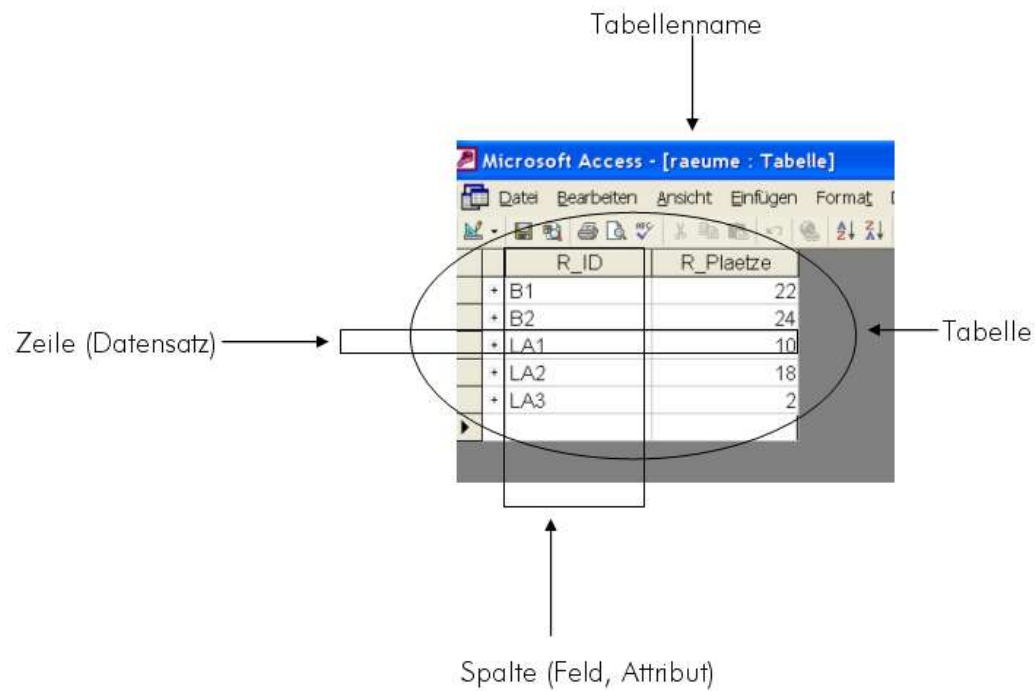
spaltenname ist der frei zu vergebene Name für die zu erstellende Spalte.

datentyp variiert in Name und Ausprägung abhängig vom Datenbankanbieter und kann lauten

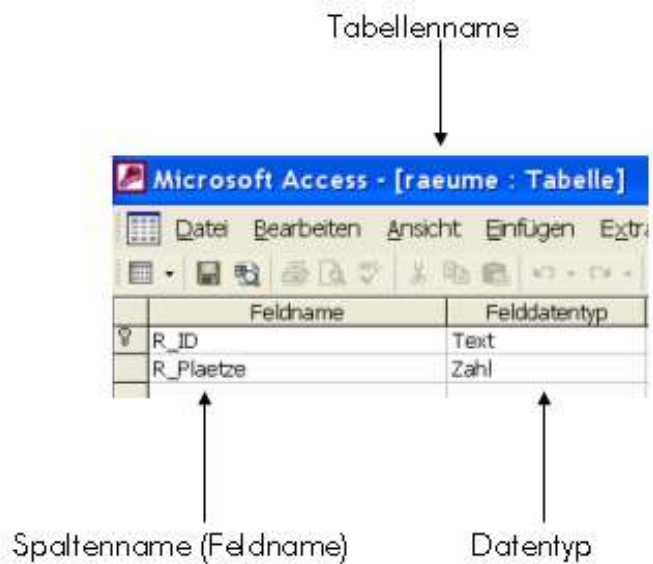
- für Zeichen: CHAR(n), VARCHAR(n), VARCHAR2(n), ....
- für Zahlen: NUMBER(n), NUMBER(n,d), INTEGER, FLOAT, LONG, DECIMAL(n,d), ...
- für Datumswerte: DATE, DATETIME, INTERVAL, .....
- für Binardaten: BLOB, RAW(n), LONG RAW ...

NOT NULL legt fest, dass bei diesem Feld Nullwerte nicht erlaubt sind und somit ein leerer (unbestimmter, unbekannter) Feldinhalt unzulässig ist.

Eine Tabelle der Schuldatenbank in der Datenblattansicht:



Eine Tabelle der Schuldatenbank in der Entwurfs/Design-Ansicht:



## Exkurs: Built-In Datentypen im Vergleich

Oracle:<sup>5</sup>

Datentyp	Spezifika
VARCHAR2(size [BYTE   CHAR])	Variable-length character string having maximum length size bytes or characters. Maximum size is 4000 bytes or characters, and minimum is 1 byte or 1 character. You must specify size for VARCHAR2. BYTE indicates that the column will have byte length semantics. CHAR indicates that the column will have character semantics.
NUMBER [ (p [, s]) ]	Number having precision p and scale s. The precision p can range from 1 to 38. The scale s can range from -84 to 127. Both precision and scale are in decimal digits. A NUMBER value requires from 1 to 22 bytes.
DATE	Valid date range from January 1, 4712 BC, to December 31, 9999 AD. The default format is determined explicitly by the NLS_DATE_FORMAT parameter or implicitly by the NLS_TERRITORY parameter. The size is fixed at 7 bytes. This datatype contains the datetime fields YEAR, MONTH, DAY, HOUR, MINUTE, and SECOND. It does not have fractional seconds or a time zone.
BLOB	A binary large object. Maximum size is (4 gigabytes - 1) * (database block size).

Microsoft SQL Server:<sup>6</sup>

Datentyp	Spezifika
numeric	Numerische Datentypen mit fester Genauigkeit und fester Anzahl von Dezimalstellen.
date	Definiert ein Datum.
nvarchar	Unicode-Zeichendaten variabler Länge. n kann ein Wert zwischen 1 und 4.000 sein. max gibt an, dass die maximale Speichergröße $2^{31}-1$ Bytes beträgt. Die Speichergröße in Bytes ist doppelt so groß wie die Anzahl eingegebener Zeichen + 2 Bytes. Die eingegebenen Daten können 0 Zeichen lang sein. Die ISO-Synonyme für nvarchar sind national char varying und national character varying.

<sup>5</sup> Die vollständige Auflistung der Oracle Built-In Datentypen für Oracle 11g findet sich u.a. unter [http://docs.oracle.com/cd/B28359\\_01/server.111/b28286/sql\\_elements001.htm#i54330](http://docs.oracle.com/cd/B28359_01/server.111/b28286/sql_elements001.htm#i54330) ; dieser Seite wurde auch der obige Extrakt der wichtigsten Datentypen entnommen.

<sup>6</sup> Die vollständige Auflistung der SQL Server Built-In Datentypen für MS SQL Server 2008 findet sich u.a. unter <http://msdn.microsoft.com/de-de/library/ms187752.aspx> ; dieser Seite wurde auch der obige Extrakt der wichtigsten Datentypen entnommen.

Microsoft Access:<sup>7</sup>

Datentyp	Spezifika
Autowert	Sie verwenden ein AutoWert-Feld, um einen eindeutigen Wert bereitzustellen, der ausschließlich den Zweck erfüllt, dass jeder Datensatz eindeutig ist. Am häufigsten wird ein AutoWert-Feld als Primärschlüssel verwendet, und zwar insbesondere dann, wenn kein geeigneter natürlicher Schlüssel (ein auf einem Datenfeld basierender Schlüssel) verfügbar ist. Für den Wert eines AutoWert-Felds sind 4 oder 16 Bytes erforderlich.
Datum/Uhrzeit	Dient zum Speichern zeitbasierter Daten.

<sup>7</sup> Die vollständige Auflistung der Access Built-In Datentypen für MS Access 2010 findet sich u.a. unter <http://office.microsoft.com/de-at/access-help/einfuehrung-in-datentypen-und-feldeigenschaften-HA010341783.aspx> ; dieser Seite wurde auch der obige Extrakt der wichtigsten Datentypen entnommen.

Datentyp	Spezifika
Zahl	<p>Verwenden Sie diesen Datentyp, um einen numerischen Wert zu speichern, bei dem es sich nicht um einen monetären Wert handelt. Wenn Sie die Werte im Feld zum Durchführen einer Berechnung verwenden möchten, verwenden Sie dazu den Datentyp "Zahl".</p> <p>Feldgrößen:</p> <ul style="list-style-type: none"> <li>• Byte : Verwenden Sie diese Option für ganze Zahlen von 0 bis 255. Die Speicherplatzanforderung beträgt 1 Byte.</li> <li>• Integer : Verwenden Sie diese Option für ganze Zahlen von -32.768 bis 32.767. Die Speicherplatzanforderung beträgt 2 Bytes.</li> <li>• Long Integer : Verwenden Sie diese Option für ganze Zahlen von -2.147.483.648 bis 2.147.483.647. Die Speicherplatzanforderung beträgt 4 Bytes.</li> <li>• Single Verwenden Sie diese Option für numerische Gleitkommawerte, die von <math>-3,4 \times 10^{38}</math> bis <math>3,4 \times 10^{38}</math> reichen und bis zu sieben Nachkommastellen umfassen. Die Speicherplatzanforderung beträgt 4 Bytes.</li> <li>• Double Verwenden Sie diese Option für numerische Gleitkommawerte, die von <math>-1,797 \times 10^{38}</math> bis <math>1,797 \times 10^{38}</math> reichen und bis zu 15 Nachkommastellen umfassen. Die Speicherplatzanforderung beträgt 8 Bytes.</li> <li>• Replikations-ID Verwenden Sie diese Option zum Speichern einer für die Replikation erforderlichen GUID (Globally Unique Identifier). Die Speicherplatzanforderung beträgt 16 Bytes. Die Replikation wird nicht für das ACCDB-Dateiformat unterstützt.</li> <li>• Dezimal Verwenden Sie diese Option für numerische Werte von -9,999... <math>\times 10^{27}</math> bis 9,999... <math>\times 10^{27}</math>. Die Speicherplatzanforderung beträgt 12 Bytes.</li> </ul>
Text	Mit diesem Datentyp können Sie bis zu 255 Zeichen (Text) speichern.
Ja/Nein	Verwenden Sie diesen Datentyp, um einen booleschen Wert zu speichern.

### MySQL:<sup>8</sup>

Datentyp	Spezifika
INT[(M)] [UNSIGNED] [ZEROFILL]	Integer normaler Größe. Der vorzeichenbehaftete Bereich liegt zwischen -2147483648 und 2147483647. Der vorzeichenlose Bereich liegt zwischen 0 und 4294967295.

<sup>8</sup> Die vollständige Auflistung der MySQL Built-In Datentypen findet sich u.a. unter <http://dev.mysql.com/doc/refman/5.1/de/data-type-overview.html> ; dieser Seite wurde auch der obige Extrakt der wichtigsten Datentypen entnommen.

Datentyp	Spezifika
DECIMAL[(M[,D])] [UNSIGNED] [ZEROFILL]	Gepackte „exakte“ Festkommazahl. M ist die Gesamtzahl von Dezimalstellen (Genauigkeit), D die Anzahl der Stellen hinter dem Dezimalpunkt. Der Dezimalpunkt sowie das Zeichen '-' (für negative Zahlen) werden bei der Zählung für M nicht berücksichtigt. Wenn D 0 ist, haben die Werte keinen Dezimalpunkt und keine Nachkommastellen. Die maximale Anzahl der Stellen (M) beträgt bei DECIMAL 65, die maximale Anzahl unterstützter Dezimalstellen (D) 30. Wird D weggelassen, dann wird als Vorgabe 0 verwendet; fehlt die Angabe M, dann ist 10 der Standardwert.
DATE	Datum. Der unterstützte Bereich liegt zwischen '1000-01-01' und '9999-12-31'. MySQL zeigt DATE-Werte im Format 'YYYY-MM-DD' an, gestattet Ihnen aber, wahlweise Strings oder Zahlen in DATE-Spalten einzugeben.
[NATIONAL] VARCHAR(M) [BINARY]	Ein String variabler Länge. M gibt die maximale Spaltenlänge an. Für M liegt der zulässige Bereich zwischen 0 und 65.535. (Die tatsächliche Maximallänge einer VARCHAR-Spalte wird durch die maximale Datensatzlänge und den von Ihnen verwendeten Zeichensatz bestimmt. Die effektive Maximallänge beträgt 65.532 Byte.)
BLOB[(M)]	Eine BLOB-Spalte mit einer Maximallänge von 65.535 ( $2^{16} - 1$ ) Byte.



## 2. Anlegen eines Index (CREATE INDEX)

### Ein Index

- beschleunigt die Suche<sup>9</sup>
- wird über ein oder mehrere Felder (Attribute, Spalten) definiert
- wird für eindeutige Schlüssel mit dem Schlüsselwort UNIQUE eingeleitet

Die 'einfache' Syntax lautet:

```
CREATE [UNIQUE] INDEX indexname ON tabellenname (spaltenname datentyp)
```

indexname ist der frei zu vergebene Name für den zu erstellenden Index.

tabellenname ist der Name einer existierenden Tabelle.

spaltenname ist der Name einer existierenden Spalte der gewählten Tabelle.

## 3. Anlegen eines Synonyms (CREATE SYNONYM)

### Ein Synonym

- ist ein alternativer Name und/oder ein abgekürzter/sprechenderer Name
- kann für bestehende Tabellen vergeben werden

Die 'einfache' Syntax lautet:

```
CREATE SYNONYM synonymname FOR tabellenname
```

synonymname ist der frei zu vergebene Name für das zu erstellende Synonym.

tabellenname ist der Name einer existierenden Tabelle.

---

<sup>9</sup> .. und verlangsamt im Gegenzug ein Insert/Update. Ein Index sollte daher immer mit Bedacht gewählt werden.

#### 4. Anlegen einer View (CREATE VIEW)

##### Eine View

- ist eine logische Sicht auf eine Tabelle bzw. auf mehrere Tabellen
- kann als virtuelle, tatsächlich nicht physisch gespeicherte Tabelle betrachtet werden
- ist bei der Bereitstellung von Daten für unterschiedliche Benutzergruppen relevant
- verkörpert das Ergebnis eines SQL Befehls
- unterliegt (im Vergleich zu einer Tabelle) Einschränkungen bezüglich der Änderungsoperationen

Die 'einfache' Syntax lautet:

```
CREATE VIEW viewname AS selectbefehl
```

viewname ist der frei zu vergebene Name für die zu erstellende Sicht.

selectbefehl ist die Regel, nach der die View auf der Grundlage bestehender Tabellen aufzubauen ist.

## 5. Verändern von Tabellenstrukturen (ALTER TABLE)

### Eine Tabelle

- kann wachsen durch die Einführung von zusätzlichen Feldern (Spalten)
- kann schrumpfen durch die Entfernung von bestehenden Feldern (Spalten)
- ist offen gegenüber Änderungen der Datentypen der einzelnen Felder

Die 'einfache' Syntax zum Hinzufügen von Spalten lautet:

```
ALTER TABLE tabellenname ADD (neuerspaltenname datentyp)
```

tabellenname ist der Name einer existierenden Tabelle.

neuerspaltenname ist der frei zu vergebene Name für das hinzukommende Feld.

datentyp ist der frei zu vergebende Datentyp des hinzukommenden Feldes.

Die 'einfache' Syntax zum Löschen von Spalten lautet:

```
ALTER TABLE tabellenname DROP (alterspaltenname)
```

tabellenname ist der Name einer existierenden Tabelle.

alterspaltenname ist Name des zu entfernenden Feldes der angegebenen Tabelle.

Die 'einfache' Syntax zum Verändern des Datentyps lautet:

```
ALTER TABLE tabellenname MODIFY (alterspaltenname neuerdatentyp)
```

tabellenname ist der Name einer existierenden Tabelle.

alterspaltenname ist Name eines existierenden Feldes der angegebenen Tabelle.

neuerdatentyp ist der Datentyp, der für das angegebene Feld zum Einsatz kommen soll.

## 6. Löschen einer Tabelle (DROP TABLE)

### Das Löschen einer Tabelle

- soll mit äußerster Vorsicht vorgenommen werden
- bewirkt das Löschen der Tabellenstruktur plus der darauf definierten Indizes und Synonyme
- macht auch nicht vor den in der Tabelle enthaltenen Daten halt
- obliegt demjenigen, der Eigentümer- oder Administratorrechte an dieser Tabelle hat

Die Syntax zum Löschen einer Tabelle lautet:

`DROP TABLE tabellenname`

tabellenname ist der Name der zu löschenden Tabelle.

## 7. Löschen eines Index (DROP INDEX)

Die Syntax zum Löschen eines Index lautet:

`DROP INDEX indexname`

indexname ist der Name des zu löschenden Index

## 8. Löschen eines Synonyms (DROP SYNONYM)

Die Syntax zum Löschen eines Synonym lautet:

**DROP SYNONYM synonymname**

synonymname ist der Name des zu löschenden Synonyms.

## 9. Löschen einer View (DROP VIEW)

Die Syntax zum Löschen einer View lautet:

**DROP VIEW viewname**

viewname ist der Name der zu löschenden View.

## D. DML Befehle

DML steht für DATA MANIPULATION LANGUAGE und umfasst alle Befehle, die erforderlich sind, um die in den Tabellen gespeicherten Informationen zu manipulieren.

Zu den DML Befehlen zählen

- SELECT zum Lesen
- INSERT zum Einfügen
- UPDATE zum Ändern
- DELETE zum Löschen

### 1. Lesen von Datensätzen aus einer/mehreren Tabelle/n (SELECT)

Das Lesen von Daten

- kann sich auf alle oder ausgewählte Spalten einer Tabelle oder mehrerer Tabellen beziehen
- führt zu keiner Modifikation der in der Tabelle / den Tabellen gespeicherten Daten

Die ‚einfache‘ Syntax zum Lesen lautet:

SELECT spaltenname FROM tabellenname WHERE bedingung

spaltenname ist der Name einer zu selektierenden Spalte der im weiteren genannten Tabelle.

tabellenname ist der Name einer existierenden Tabelle.

bedingung definiert die Kriterien, nach denen Zeilen aus der Tabelle auszuwählen sind.

**Das SELECT wird DEN Schwerpunkt dieses Skriptums bilden!**

## 2. Einfügen von Datensätzen in eine Tabelle (INSERT)

### Das Einfügen von Daten

- kann datensatzweise erfolgen
- kann sich auf einen Extrakt aus dem Inhalt einer anderen Tabelle beziehen

Die 'einfache' Syntax zum Eintragen von Datensätzen lautet:

```
INSERT INTO tabellenname (spaltennamenliste) VALUES (werteliste)
```

tabellenname ist der Name der Tabelle, in der ein Datensatz eingetragen werden soll.

spaltennamenliste ist die Liste der Spalten der angegebenen Tabelle.

werteliste beinhaltet die für die einzelnen Spalten vorgesehenen konkreten Werte.

## 3. Das Bearbeiten von Datensätzen in einer Tabelle (UPDATE)

### Das Bearbeiten von Daten

- kann datensatzweise erfolgen
- kann für eine Gruppe von Datensätze durchgeführt werden

Die 'einfache' Syntax zum Bearbeiten von Datensätzen lautet:

```
UPDATE tabellenname SET (spaltenname = ausdruck) WHERE bedingung
```

tabellenname ist der Name einer existierenden Tabelle.

spaltenname ist das Feld der angegebenen Tabelle, dessen Wert verändert werden soll.

ausdruck ist der Wert, auf den der Inhalt des genannten Feldes geändert werden soll.

bedingung dient zur Einschränkung der zu korrigierenden Datensätze.

#### 4. Das Löschen von Datensätzen aus einer Tabelle (DELETE)

Die 'einfache' Syntax zum Löschen von Datensätzen lautet:

DELETE FROM tabellenname WHERE bedingung

tabellenname ist der Name einer existierenden Tabelle.

bedingung dient zur Einschränkung der zu löschenden Datensätze.



## E. Befehle für die Verwaltung von Zugriffsrechten

Zum Aufgabenbereich eines Datenbankadministrators (DBA) zählt unter anderem die Benutzerverwaltung, d.h. das Management der Rechte, die einzelnen Benutzern oder Benutzergruppen in einer Mehrbenutzerumgebung bezüglich der Verwendung der Datenbank zugestanden werden.

Die beiden Schlüsselwörter hierzu lauten:

- GRANT zum Gewähren von Rechten
- REVOKE zum Entziehen von Rechten

### 1. Die Vergabe von Zugriffsrechten

Die 'einfache' Syntax zur Vergabe von Zugriffsrechten lautet:

GRANT recht ON tabellenname TO benutzer

recht spezifiziert das zu vergebende Privileg.  
tabellenname ist der Name einer existierenden Tabelle.  
benutzer ist der Name des Benutzers.

Zu den Rechten zählt unter anderem die Erlaubnis

- zur Anmeldung bei einer Datenbank
- zum Anlegen von Tabellen, Views, Indizes, et. al.
- zum Ausführen von Abfrage-Statements

### 2. Der Entzug von Zugriffsrechten

Die 'einfache' Syntax zum Entzug von Zugriffsrechten lautet:

REVOKE recht ON tabellenname FROM benutzer

recht spezifiziert das zu entziehende Privileg.  
tabellenname ist der Name einer existierenden Tabelle.  
benutzer ist der Name des Benutzers.

## F. Befehle zur Transaktionssteuerung

Zunächst zur Erklärung, was unter einer Transaktion zu verstehen ist: Eine Transaktion ist eine Folge von logisch zusammengehörigen Operationen (wie Lesen, Ändern, Einfügen oder Löschen) auf einer Datenbank und führt eine Datenbank von einem konsistenten Zustand in einen anderen konsistenten Zustand über.

Soweit zur Theorie. Nun ein praktisches Beispiel aus dem Bankwesen zur Illustration: im Sinne einer Transaktion ist die Überweisung eines Betrages von 100 € von einem Konto auf ein Sparbuch erst dann abgeschlossen, wenn einerseits vom Konto 100 € abgebucht und andererseits auf dem Sparbuch 100 € gutgeschrieben wurden.

Die Transaktion arbeitet nach dem ‚alles oder nichts Prinzip‘. Nur wenn alle Aktionen erfolgreich durchgeführt werden konnten, ist das Resultat in der Datenbank zu speichern. Es versteht sich von selbst, dass hiermit Transaktionen ihre Ergebnisse nur nach dem Commit sichtbar machen dürfen – da ja jederzeit die Möglichkeit bestünde, die Operationen rückgängig zu machen.

Zu den Befehlen für die Transaktionssteuerung zählen

- COMMIT zum Speichern aller Änderungen einer Transaktion
- ROLLBACK zum Rückgängigmachen von Änderungen

### 1. Commit einer Transaktion (COMMIT)

Die ‘einfache’ Syntax zum Permanent-Machen von Änderungen in der Datenbank lautet:

COMMIT;

### 2. Rollback einer Transaktion (ROLLBACK)

Die ‘einfache’ Syntax zum Zurückrollen der Operationen innerhalb einer Transaktion lautet:

ROLLBACK;

## IV. DAS SELECT

In einem Atemzug mit SQL wird oftmals das Stichwort SELECT genannt: das folgende Kapitel widmet sich nun ausschließlich dem Auslesen von Informationen aus einer relationalen Datenbank.

### G. Auswahl von Spalten

Der Merksatz für das Lesen von Daten lautet:

```
SELECT spaltennamen
FROM tabellenname
WHERE bedingung;
```

In die Umgangssprache übersetzt könnte man auch lesen: SELECT was (die Spalten) FROM wo (den Tabellen) WHERE wann (unter welchen Bedingungen).

#### 1. Beispiel 1: Ausgabe aller Räume

```
SELECT *
FROM raeume;
```

Ausgabe:

R_ID	R_Plaetze
B1	22
B2	24
LA1	10
LA2	18
LA3	2

```
SELECT *
FROM tabellenname;
```

SELECT \* liefert alle Daten, die in der angeführten Tabelle hinterlegt sind. Der \* steht in diesem Sinne für ‚alle Spalten‘.

## 2. Beispiel 2: Ausgabe aller Langbezeichnungen der Gegenstände

```
SELECT    g_bez  
FROM      gegenstaende;
```

Ausgabe:

g_bez
Betriebliche Organisation
Betriebssysteme
Datenbanksysteme
Deutsch
Englisch
Mathematik
Programmieren
Prozessrechnerverbund
Rechnungswesen
System-Einsatzplanung
Technische Datenorganisation

## 3. Beispiel 3: Ausgabe aller Prüflinge mit Prüfungsfach und Prüfungsdatum

```
SELECT    p_s_kandidat, p_g_fach, p_datum
FROM      pruefungen;
```

Ausgabe:

p_s_kandidat	p_g_fach	p_datum
55	TDO	24.12.1997
1	TDO	01.01.1998
1266	BO	04.01.1998
3	TDO	12.01.1998
22	TDO	12.01.1998
111	MA	06.02.1998
19	TDO	15.04.1998
111	TDO	07.06.1998
1	BO	15.03.1999
22	BO	15.03.1999
111	TDO	15.03.1999
1	MA	01.04.1999
1	MA	01.04.1999
4	MA	01.04.1999
2222	DBSYS	01.10.2003
2223	DBSYS	01.10.2003
2224	DBSYS	01.10.2003
3333	RW	01.10.2003
2222	BSYS	10.10.2003
2223	BSYS	10.10.2003
2224	BSYS	10.10.2003
2225	BSYS	10.10.2003
3333	BSYS	10.10.2003
3334	PR	05.11.2003
3335	PR	06.11.2003
3336	PR	07.11.2003
3336	PR	12.11.2003
2222	PR	13.11.2003
2224	TDO	14.11.2003
2225	EN	15.11.2003
2225	MA	15.11.2003

## H. Arbeiten mit Bedingungen

Datenbanken werden immer dann eingesetzt, wenn es gilt, viele Daten / Informationen zu speichern – natürlich mit dem Ziel, auf diese Daten jederzeit leicht wieder zugreifen zu können.

Das WHERE leitet den Bedingungsteil eines SQL Statements ein – die Bedingungen schränken die Suche und damit auch das Resultat der Abfrage ein.

```
SELECT spaltenname
FROM tabellenname
WHERE bedingung;
```

### 1. Beispiel 4: Schüler mit einer Schülernummer kleiner 20

```
SELECT s_schnr, s_vorname, s_name, s_k_klasse
FROM schueler
WHERE s_schnr < 20;
```

Ausgabe:

s_schnr	s_vorname	s_name	s_k_klasse
1	Richard	Adler	03TA
3	Burghard	Bartgeier	03TB
4	Robert	Rotkehlchen	03TA
16	Gustav	Geier	03TA
19	Sebastian	Sitzenbleiber	03TA

### 2. Beispiel 5: Schüler mit einer Schülernummer gleich 2222

```
SELECT s_schnr, s_vorname, s_name, s_k_klasse
FROM schueler
WHERE s_schnr = 2222;
```

Ausgabe:

s_schnr	s_vorname	s_name	s_k_klasse
2222	Anna	Apfelbaum	0M5Q

### 3. Beispiel 6: Lehrer mit einem Gehalt < 500 und > 200 (mit AND)

```
SELECT    l_vorname, l_name, l_gehalt
FROM      lehrer
WHERE     l_gehalt > 200 AND l_gehalt < 500;
```

Ausgabe:

l_vorname	l_name	l_gehalt
Franz	Berger	400,00
Alfred	Beringer	220,00
Hans	Bilek	280,00
Gustav	Hanke	300,00
	Lorenz	300,00
Walter	Pirkner	220,00

Einzelne Bedingungen können mittels der logischen Operatoren AND und OR zu einer größeren, komplexeren Bedingung kombiniert werden.

Groß- und Kleinschreibung wird in SQL nicht unterschieden. Ebenso sind Zeilenumbrüche irrelevant.

Aber machen Sie sich und auch denjenigen, die Ihre SQL Statements lesen dürfen / müssen, das Leben etwas einfacher, indem Sie versuchen, auf leichte Leserlichkeit acht zu geben.

```
SELECT    l_vorname, l_name, l_gehalt
FROM      lehrer
WHERE     l_gehalt > 200 AND l_gehalt < 500;
```

liefert das gleiche Resultat wie

```
Select l_vorname, l_NAME,
l_geHAIt
FROM      lehrer where
l_gehalt > 200
and L_GEHALT < 500;
```

Welches Statement konnten Sie leichter lesen und erfassen?

## 4. Beispiel 7: Lehrer mit einem Gehalt &lt; 500 und &gt; 200 (mit BETWEEN)

```

SELECT    l_vorname, l_name, l_gehalt
FROM      lehrer
WHERE     l_gehalt BETWEEN 201 AND 499;

```

Ausgabe:

l_vorname	l_name	l_gehalt
Franz	Berger	400,00
Alfred	Beringer	220,00
Hans	Bilek	280,00
Gustav	Hanke	300,00
	Lorenz	300,00
Walter	Pirkner	220,00

Das BETWEEN einzusetzen, erspart Schreibarbeit, konkret ersetzt ein ‚spaltenname BETWEEN untergrenze AND obergrenze‘ eine Bedingung der Art ‚spaltenname >= untergrenze AND spaltenname <= obergrenze‘.

```

SELECT spaltenname1, spaltenname2
FROM tabellenname
WHERE spaltenname1 BETWEEN untergrenze AND obergrenze;

```

## 5. Beispiel 8: Lehrergehalt &lt; 500, &gt; 200 und Geburtsdatum vor 1.1.1955

```

SELECT    l_vorname, l_name, l_gehalt, l_gebdat
FROM      lehrer
WHERE     (l_gehalt BETWEEN 201 AND 499) AND l_gebdat < #01/01/1955#;

```

Ausgabe:

l_vorname	l_name	l_gehalt	l_gebdat
Franz	Berger	400,00	02.03.1945
Hans	Bilek	280,00	03.03.1932
Gustav	Hanke	300,00	12.12.1950



## 6. Beispiel 9: Lehrer mit Gehalt größer 250 oder Geburtsdatum vor 1.1.1955

```

SELECT    l_vorname, l_name, l_gehalt, l_gebdat
FROM      lehrer
WHERE     l_gehalt > 250 OR l_gebdat < #01/01/1955#;

```

Ausgabe:

l_vorname	l_name	l_gehalt	l_gebdat
	Aichholzer	150,00	07.09.1946
Franz	Berger	400,00	02.03.1945
Hans	Bilek	280,00	03.03.1932
Gustav	Hanke	300,00	12.12.1950
	Lorenz	300,00	12.12.1958
Nikolaus	Lenau	180,00	22.02.1938
Margit	Makandreou	120,00	01.04.1947
Rudi	Radlbauer	115,00	11.11.1949

## 7. Beispiel 10: Schüler mit Wohnort Hernals

```

SELECT    *
FROM      schueler
WHERE     s_adresse = 'Hernals';

```

Ausgabe:

S_SCHNR	S_Name	S_Vorname	S_Gebdat	S_Adresse	S_K_Klasse
19	Sitzenbleiber	Sebastian	08.01.1983	Hernals	03TA
111	Ente	Eberhard	19.04.1983	Hernals	03TB

Ein Vergleich auf Gleichheit (mit =) erfordert bei numerischen Werten keine Hochkommata, bei Strings jedoch schon.

Zum Vergleich: s\_schnr = 1, aber s\_adresse = 'Hernals'.

## 8. Beispiel 11: Schüler mit Wohnort Hernals oder Mödling

```

SELECT *
FROM schueler
WHERE s_adresse = 'Hernals' OR s_adresse = 'Mödling';

```

Ausgabe:

S_SCHNR	S_Name	S_Vorname	S_Gebdat	S_Adresse	S_K_Klasse
19	Sitzenbleiber	Sebastian	08.01.1983	Hernals	03TA
111	Ente	Eberhard	19.04.1983	Hernals	03TB
2223	Birkenbaum	Berta	12.12.1965	Mödling	0M5Q

## 9. Beispiel 12: Schüler der 03TA mit Wohnort Hernals

```

SELECT *
FROM schueler
WHERE s_adresse = 'Hernals' AND s_k_klasse = '03TA';

```

Ausgabe:

S_SCHNR	S_Name	S_Vorname	S_Gebdat	S_Adresse	S_K_Klasse
19	Sitzenbleiber	Sebastian	08.01.1983	Hernals	03TA

## 10. Beispiel 13: Direktoren

```

SELECT v_l_vorg
FROM vorgesetzte
WHERE v_art = 'Direktor';

```

Ausgabe:

v_l_vorg
HI
HI
HI
HI
HI
HI
HI
HI
HI
HI
HI

## 11. Beispiel 14: Untergebene Lehrer von Direktoren

```

SELECT    v_l_unt, v_l_vorg
FROM      vorgesetzte
WHERE     v_art = 'Direktor';

```

Ausgabe:

v_l_unt	v_l_vorg
BA	HI
BI	HI
HA	HI
LO	HI
PS	HI
AM	HI
UK	HI
MM	HI
BF	HI
SS	HI

## 12. Beispiel 15: Untergebene Lehrer von Nicht-Direktoren

```

SELECT    *
FROM      vorgesetzte
WHERE     v_art <> 'Direktor';

```

Ausgabe:

V_L_Vorg	V_Art	V_L_Unt
BA	Fgrp Prog	PS
BI	Fgrp RW	LO
HA	AV	BA
HA	AV	BI
HA	AV	LO
HA	AV	PS
BF	AV	AM
BF	AV	UK
SS	AV	MM
SS	AV	BA
BA	Fgrp Prog	AM
BA	Fgrp Prog	UK

## 13. Beispiel 16: Schüler mit einer 1 auf eine beliebige Prüfung

```

SELECT    p_s_kandidat, p_g_fach, p_note
FROM      pruefungen
WHERE     p_note = 1;

```

Ausgabe:

p_s_kandidat	p_g_fach	p_note
1	TDO	1
3	TDO	1
111	TDO	1
2222	DBSYS	1
2224	DBSYS	1
3333	RW	1
3335	PR	1
2222	PR	1
3333	BSYS	1

## 14. Beispiel 17: Schüler mit einer 1 in Englisch

```

SELECT    p_s_kandidat, p_g_fach, p_note
FROM      pruefungen
WHERE     p_note = 1 AND p_g_fach = 'EN';

```

Ausgabe:

p_s_kandidat	p_g_fach	p_note

Diese ‚leere‘ Ausgabe bedeutet in unserem Fall, dass es keinen Schüler gibt, der eine Prüfung im Fach Englisch mit einer 1 absolvierte.

15. Beispiel 18: Lehrergehalt 100 oder 220 und Geburtsdatum vor 1.1.1956 (mit / ohne Klammern)

```
SELECT l_vorname, l_name, l_gehalt, l_gebdat
FROM lehrer
WHERE (l_gehalt = 100 OR l_gehalt = 220) AND l_gebdat < #01/01/1956#;
```

Ausgabe:

l_vorname	l_name	l_gehalt	l_gebdat
Walter	Pirkner	220,00	22.06.1955

Klammern dienen einerseits dazu, die Übersichtlichkeit zu erhöhen, andererseits auch zur Festlegung einer – von der Standardpriorisierung (AND geht vor OR) – abweichenden Auswertungsreihenfolge der einzelnen Bedingungssteile.

16. Beispiel 19: Lehrergehalt 100 oder 220 und Geburtsdatum vor 1.1.1956 (mit / ohne Klammern)

```
SELECT l_vorname, l_name, l_gehalt, l_gebdat
FROM lehrer
WHERE l_gehalt = 100 OR l_gehalt = 220 AND l_gebdat < #01/01/1956#;
```

Ausgabe:

l_vorname	l_name	l_gehalt	l_gebdat
Walter	Pirkner	220,00	22.06.1955
Johann	Preißl	100,00	05.07.1956

Vergleichen Sie das Ergebnis aus Beispiel 18 mit dem aus Beispiel 19.

## 17. Beispiel 20: Lehrer Gehalt 82, 220, 300 und Geburtsdatum vor 1.1.1970

```

SELECT  l_vorname, l_name, l_gehalt, l_gebdat
FROM    lehrer
WHERE   (l_gehalt = 82 OR l_gehalt = 220 OR l_gehalt = 300)
        AND (l_gebdat < #01/01/1970#);

```

Ausgabe:

l_vorname	l_name	l_gehalt	l_gebdat
Alfred	Beringer	220,00	15.07.1961
Gustav	Hanke	300,00	12.12.1950
	Lorenz	300,00	12.12.1958
Walter	Pirkner	220,00	22.06.1955

## 18. Beispiel 21: Lehrer mit Geburtsdatum ungleich 12.12.1950

```

SELECT  l_vorname, l_name, l_gehalt, l_gebdat
FROM    lehrer
WHERE   l_gebdat <> #12/12/1950#;

```

Ausgabe:

l_vorname	l_name	l_gehalt	l_gebdat
Max	Renkin	180,00	19.08.1961
	Aichholzer	150,00	07.09.1946
Franz	Berger	400,00	02.03.1945
Alfred	Beringer	220,00	15.07.1961
Hans	Bilek	280,00	03.03.1932
	Lorenz	300,00	12.12.1958
Nikolaus	Lenau	180,00	22.02.1938
Margit	Makandreou	120,00	01.04.1947
Walter	Pirkner	220,00	22.06.1955
Johann	Preißl	100,00	05.07.1956
Rudi	Radlbauer	115,00	11.11.1949

## 19. Beispiel 22: Lehrergehalt 82, 220, 300 und Geburtsdatum vor 1.1.1970

```

SELECT    l_vorname, l_name, l_gehalt, l_gebdat
FROM      lehrer
WHERE     (l_gehalt IN (82, 220, 300)) AND (l_gebdat < #01/01/1970#);

```

Ausgabe:

l_vorname	l_name	l_gehalt	l_gebdat
Alfred	Beringer	220,00	15.07.1961
Gustav	Hanke	300,00	12.12.1950
	Lorenz	300,00	12.12.1958
Walter	Pirkner	220,00	22.06.1955

Das IN einzusetzen, erspart Schreibarbeit, konkret ersetzt ein ‚spaltenname IN (wert1, wert2, wert3)‘ eine Bedingung der Art spaltenname = wert1 OR spaltenname = wert2 OR spaltenname = wert3.

```

SELECT *
FROM tabellenname
WHERE spaltenname IN (werteliste);

```

## I. Qualifizierung

Vor dem Spaltenname kann, mit einem Punkt voneinander getrennt, der Tabellename angegeben werden. In diesen Falle wird von einer Qualifizierung gesprochen.

```
SELECT tabellenname.spaltenname  
FROM tabellenname;
```

Wann ist nun eine Qualifizierung erforderlich?

Die folgende Ausgangssituation muss gegeben sein: Es soll ein Join zwischen zwei oder mehreren Tabellen durchgeführt werden und der Join soll über Spalten gleichen Namens (aus den angeführten Tabellen) erfolgen. Genau dann muss im SQL Statement vor dem Spaltenname der Tabellename angegeben werden.



## 1. Beispiel 23: Schüler und deren Geburtstag (ohne Qualifizierung)

```
SELECT    s_name, s_gebdat
FROM      schueler;
```

Ausgabe:

s_name	s_gebdat
Adler	12.03.1973
Bartgeier	01.01.1980
Rotkehlchen	05.05.1977
Geier	04.05.1978
Sitzenbleiber	08.01.1983
Klassenbester	03.09.1985
Picasso	03.12.1980
Hundertwasser	02.02.1980
Pollak	08.04.1982
Feuerstein	02.07.1993
Geröllheimer	15.03.1984
Ente	19.04.1983
Gans	01.01.1962
Schwan	23.04.1963
Apfelbaum	12.12.1965
Birnenbaum	12.12.1965
Marillenbaum	11.11.1964
Kirschebaum	10.10.1963
Tanne	09.09.1962
Fichte	09.09.1962
Föhre	28.09.1972
Kiefer	17.04.1969
Buche	31.01.1980
Eibe	20.02.1982

## 2. Beispiel 24: Schüler und deren Geburtstag (mit Qualifizierung)

```
SELECT    schueler.s_name, schueler.s_gebdat
FROM      schueler;
```

Ausgabe:

s_name	s_gebdat
Adler	12.03.1973
Bartgeier	01.01.1980
Rotkehlchen	05.05.1977
Geier	04.05.1978
Sitzenbleiber	08.01.1983
Klassenbester	03.09.1985
Picasso	03.12.1980
Hundertwasser	02.02.1980
Pollak	08.04.1982
Feuerstein	02.07.1993
Geröllheimer	15.03.1984
Ente	19.04.1983
Gans	01.01.1962
Schwan	23.04.1963
Apfelbaum	12.12.1965
Birnenbaum	12.12.1965
Marillenbaum	11.11.1964
Kirschebaum	10.10.1963
Tanne	09.09.1962
Fichte	09.09.1962
Föhre	28.09.1972
Kiefer	17.04.1969
Buche	31.01.1980
Eibe	20.02.1982

## J. Sortierung

Es gibt genau zwei Formen der Sortierung

- aufsteigend (A-Z, 0-9)
- absteigend (Z-A, 9-0)

```
SELECT spaltenname  
FROM tabellenname  
WHERE bedingung  
ORDER BY spaltenname;
```

Dem Schlüsselwort ORDER BY folgt der Name der Spalte/n, nach der/denen sortiert werden soll.

ASC definiert als Sortierreihenfolge aufsteigend (Default), DESC absteigend.

## 1. Beispiel 25: Alphabetisch absteigend sortierte Schülerliste

```

SELECT    s_vorname, s_name
FROM      schueler
ORDER BY  s_name DESC;

```

Ausgabe:

	s_vorname	s_name
►	Theodor	Tanne
	Sebastian	Sitzenbleiber
	Susanne	Schwan
	Robert	Rotkehlchen
	Peter	Pollak
	Paul	Picasso
	Martha	Marillenbaum
	Konrad	Klassenbester
	Karoline	Kirschebaum
	Kasimir	Kiefer
	Helga	Hundertwasser
	Pampam	Geröllheimer
	Gustav	Geier
	Gustav	Gans
	Franz	Föhre
	Fridolin	Fichte
	Bebbles	Feuerstein
	Eberhard	Ente
	Engelbert	Eibe
	Balduin	Buche
	Berta	Birnenbaum
	Burghard	Bartgeier
	Anna	Apfelbaum
	Richard	Adler
*		

## 2. Beispiel 26: Alphabetisch absteigend sortierte Schülerliste (etwas kürzer)

```
SELECT    s_vorname, s_name
FROM      schueler
ORDER BY  2 DESC;
```

Die Ziffer nach dem Schlüsselwort ORDER BY bezieht sich allgemein gesprochen auf die Position des angegebenen Attributes in der SELECT Klausel. Im obigen Beispiel verweist 2 auf s\_name, 1 würde sich auf s\_vorname beziehen.

Ausgabe:

	s_vorname	s_name
►	Theodor	Tanne
	Sebastian	Sitzenbleiber
	Susanne	Schwan
	Robert	Rotkehlchen
	Peter	Pollak
	Paul	Picasso
	Martha	Marillenbaum
	Konrad	Klassenbester
	Karoline	Kirschebaum
	Kasimir	Kiefer
	Helga	Hundertwasser
	Pampam	Geröllheimer
	Gustav	Geier
	Gustav	Gans
	Franz	Föhre
	Fridolin	Fichte
	Bebbles	Feuerstein
	Eberhard	Ente
	Engelbert	Eibe
	Balduin	Buche
	Berta	Birnenbaum
	Burghard	Bartgeier
	Anna	Apfelbaum
	Richard	Adler
*		

### 3. Beispiel 27: Zweifach sortierte Schülerliste

```
SELECT    s_gebdat, s_name, s_vorname
FROM      schueler
ORDER BY  s_gebdat ASC, s_name DESC;
```

Ausgabe:

	s_gebdat	s_name	s_vorname
►	01.01.1962	Gans	Gustav
	09.09.1962	Tanne	Theodor
	09.09.1962	Fichte	Fridolin
	23.04.1963	Schwan	Susanne
	10.10.1963	Kirschebaum	Karoline
	11.11.1964	Marillenbaum	Martha
	12.12.1965	Birnenbaum	Berta
	12.12.1965	Apfelbaum	Anna
	17.04.1969	Kiefer	Kasimir
	28.09.1972	Föhre	Franz
	12.03.1973	Adler	Richard
	05.05.1977	Rotkehlchen	Robert
	04.05.1978	Geier	Gustav
	01.01.1980	Bartgeier	Burghard
	31.01.1980	Buche	Balduin
	02.02.1980	Hundertwasser	Helga
	03.12.1980	Picasso	Paul
	20.02.1982	Eibe	Engelbert
	08.04.1982	Pollak	Peter
	08.01.1983	Sitzenbleiber	Sebastian
	19.04.1983	Ente	Eberhard
	15.03.1984	Geröllheimer	Pampam
	03.09.1985	Klassenbester	Konrad
	02.07.1993	Feuerstein	Bebbles
...			

Der erste, nach dem ORDER BY genannte Spaltenname ist das primäre Sortierkriterium, der zweite, nach dem ORDER BY genannte Spaltenname, das sekundäre.

## K. Alias für Spaltennamen

Die Spaltenüberschriften der Ergebnistabelle sind per Default gleich den, nach dem SELECT angegebenen Spaltennamen.

Durch die Angabe eines Aliasnamen bei einer Spalte kann bei der Ausgabe der Default übersteuert und durch den Aliasnamen ersetzt werden.

```
SELECT spaltenname AS alias_spaltenname
FROM tabellenname;
```

Alias-Namen dürfen keine Leerzeichen beinhalten.

(Ist doch ein Leerzeichen gewünscht, ist der gesamte gewählte Alias unter Hochkommata zu stellen.).

### 1. Beispiel 28: Gegenstandsausgabe mit einprägsamer Spaltenbezeichnung

```
SELECT    g_id AS Gegenstandskürzel, g_bez AS Gegenstandslangbezeichnung
FROM      gegenstaende
ORDER BY  g_id;
```

Ausgabe:

Gegenstandskürzel	Gegenstandslangbezeichnung
BO	Betriebliche Organisation
BSYS	Betriebssysteme
DBSYS	Datenbanksysteme
DE	Deutsch
EN	Englisch
MA	Mathematik
PR	Programmieren
PRRV	Prozessrechnerverbung
RW	Rechnungswesen
SEP	System-Einsatzplanung
TDO	Technische Datenorganisation

## L. Wildcards

Wildcards sind Platzhalter für eine bestimmte Anzahl (genau 1 oder 0 bis n) von Zeichen und herstellerspezifisch definiert.

### Exkurs: Wildcards in unterschiedlichen DB-Systemen

	genau 1 Zeichen	0 – n Zeichen
ORACLE	_	%
MS SQL SERVER	_	%
MS Access	?	*
MySQL	_	%

Wildcards kommen bei Ähnlichkeitsvergleichen bei Strings zum Einsatz.

```
SELECT spaltenname
FROM tabellenname
WHERE spaltenname LIKE kriterium;
```

#### 1. Beispiel 29: Lehrer, deren Vorname Gustav lautet

```
SELECT l_name, l_vorname, l_gebdat
FROM lehrer
WHERE l_name = 'Gustav';
```

Ausgabe:

	l_name	l_vorname	l_gebdat
1	Hanke	Gustav	12.12.1950

#### 2. Beispiel 30: Lehrer, deren Vorname mit M beginnt

```
SELECT l_name, l_vorname, l_gebdat
FROM lehrer
WHERE l_vorname LIKE 'M*';
```

Ausgabe:



	l_name	l_vorname	l_gebdat
	Renkin	Max	19.08.1961
	Makandreou	Margit	01.04.1947

### 3. Beispiel 31: Lehrernachnamen mit einem B an 1. und einem R an 3. Stelle

```
SELECT    l_name AS Nachname, l_vorname AS Vorname, l_gebdat AS Geburtsdatum
FROM      lehrer
WHERE     l_name LIKE 'B?r*';
```

Ausgabe:

	Nachname	Vorname	Geburtsdatum
	Berger	Franz	02.03.1945
	Beringer	Alfred	15.07.1961

### 4. Beispiel 32: Klassen mit eine 0 an beliebiger Stelle

```
SELECT    k_id
FROM      klassen
WHERE     k_id LIKE '*0*';
```

Ausgabe:

k_id
1VL0
03TA
03TB
0M5Q
0M5A

## M. Duplikate

In allen Abfragen, in denen nicht das Primärschlüssel-Attribut Teil des Ergebnisses ist, kann es zu identen Resultat-Datensätzen kommen. Um diese herauszufiltern, ist das Schlüsselwort **DISTINCT** dem **SELECT** nachzustellen.

```
SELECT DISTINCT spaltenname
FROM tabellenname;
```

### 1. Beispiel 33: Stunden, die von PS unterrichtet werden (mit Duplikaten)

```
SELECT    st_l_lehrer AS Lehrerkürzel,
          st_k_klasse AS Klasse,
          st_g_fach AS Gegenstand
FROM      stunden
WHERE     st_l_lehrer = 'PS';
```

Ausgabe:

	Lehrerkürzel	Klasse	Gegenstand
	PS	03TA	TDO
	PS	03TA	TDO
	PS	03TA	TDO
	PS	03TA	PR
	PS	03TA	PR
	PS	03TB	PR
	PS	03TB	PR
	PS	03TB	TDO
	PS	03TB	TDO

### 2. Beispiel 34: Stunden, die von PS unterrichtet werden (ohne Duplikate)

```
SELECT    DISTINCT st_l_lehrer AS Lehrerkürzel,
          st_k_klasse AS Klasse,
          st_g_fach AS Gegenstand
FROM      stunden
WHERE     st_l_lehrer = 'PS';
```

Ausgabe:

	Lehrerkürzel	Klasse	Gegenstand
	PS	03TA	PR
	PS	03TA	TDO
	PS	03TB	PR
	PS	03TB	TDO

## 3. Beispiel 35: Gehälter der Lehrer (ohne Duplikate)

```
SELECT DISTINCT l_gehalt  
FROM lehrer;
```

Ausgabe:

l_gehalt
75,00
90,00
100,00
115,00
120,00
150,00
180,00
220,00
280,00
300,00
400,00

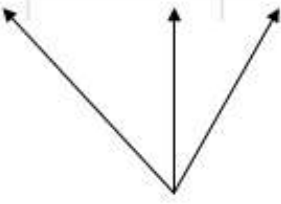
Die erste, scheinbar leere Ergebniszeile ist kein Fehler, sondern die Konsequenz der Tatsache, dass nicht bei allen Lehrern Gehälter vermerkt sind, i.e. Nullwerte vorkommen.

## N. Nullwerte

Ein Nullwert ist ein spezieller, für jeden Datentyp vorhandener Wert, der dann gespeichert wird, wenn ein konkreter Wert nicht bekannt ist.

Null (i.e. unbekannt) darf weder mit der Ziffer 0 noch mit dem Blank (Leerzeichen) verwechselt werden!

K_ID	K_Bez	K_S_Klaspr	K_S_Klasprstv	K_L_Klavst
03TA	Kolleg T 03TA	1	122	BA
03TB	Kolleg T 03TB	1266	111	BI
0M5A	Kolleg A 0M5A	3333	3338	AM
0M5Q	Kolleg A 0M5Q	2222	2223	PS
1VL0	VorbereitungLG			



Nullwert

Nullwerte in Tabellen sind bei der Gestaltung von Abfragen zu berücksichtigen – beispielsweise bei der Einschränkung auf Zeilen, in denen eine gewisse Spalte keine Nullwerte beinhaltet, oder bei der Auswertung von logischen Ausdrücken.

```
SELECT *
FROM tabellenname
WHERE spaltenname IS NULL;
```

## 1. Beispiel 36: Lehrer ohne Gehalt

```

SELECT      *
FROM        lehrer
WHERE       l_gehalt IS NULL
ORDER BY   l_id;

```

Ausgabe:

	L_ID	L_Name	L_Vorname	L_Gebdat	L_Gehalt	L_L_Chef
	HI	Hickel	Wolfgang			
	SG	Siegel	Heinz			HA
	SS	Stappler	Sonja			HI

## 2. Beispiel 37: Lehrer mit bekanntem Gehalt, aufsteigend sortiert

```

SELECT      l_id, l_name, l_vorname, l_gehalt
FROM        lehrer
WHERE       l_gehalt IS NOT NULL
ORDER BY   l_gehalt;

```

Ausgabe:

	l_id	l_name	l_vorname	l_gehalt
	AM	Moitzi	Andrea	75,00
	UK	Unger	Klaus	90,00
	PS	Preißl	Johann	100,00
	RR	Radlbauer	Rudi	115,00
	MM	Makandreou	Margit	120,00
	AI	Aichholzer		150,00
	LN	Lenau	Nikolaus	180,00
	RX	Renkin	Max	180,00
	PI	Pirkner	Walter	220,00
	BA	Beringer	Alfred	220,00
	BI	Bilek	Hans	280,00
	LO	Lorenz		300,00
	HA	Hanke	Gustav	300,00
	BF	Berger	Franz	400,00

Die Verneinung von spaltenname IS NULL lautet: spaltenname IS NOT NULL.

Bei Bedingungen, die Spalten mit Nullwerten beinhalten, kommt bei der Bedingungsauswertung die ‚dreiwertige‘ Logik von NOT, AND und OR zum Tragen.

NOT	
True	False
False	True
Unknown	Unknown

AND	True	False	Unknown
True	True	False	Unknown
False	False	False	False
Unknown	Unknown	False	Unknown

OR	True	False	Unknown
True	True	True	True
False	True	False	Unknown
Unknown	True	Unknown	Unknown

## O. Konstante

In der Ergebnistabelle können nicht nur Werte auftreten, die in der Ausgangstabelle vorliegen, sondern auch nach dem SELECT definierte Konstanten.

```
SELECT spaltenname, konstante
FROM tabellenname;
```

### 1. Beispiel 38: Geniale Schüler

```
SELECT    s_name AS Schüler, 'ist genial' AS Ausdruck
FROM      schueler
ORDER BY  s_name;
```

Ausgabe:

Schüler	Ausdruck
Adler	ist genial
Apfelbaum	ist genial
Bartgeier	ist genial
Birnenbaum	ist genial
Buche	ist genial
Eibe	ist genial
Ente	ist genial
Feuerstein	ist genial
Fichte	ist genial
Föhre	ist genial
Gans	ist genial
Geier	ist genial
Geröllheimer	ist genial
Hundertwasser	ist genial
Kiefer	ist genial
Kirschebaum	ist genial
Klassenbester	ist genial
Marillenbaum	ist genial
Picasso	ist genial
Pollak	ist genial
Rotkehlchen	ist genial
Schwan	ist genial
Sitzenbleiber	ist genial
Tanne	ist genial

## P. Aggregatfunktionen

Aggregatfunktionen sind Funktionen, die einen (1 !) Wert auf der Basis der Werte einer Spalte retournieren.

Zu diesen zählen

- MIN
- MAX
- AVG
- SUM
- COUNT

MIN berechnet das Minimum für eine Gruppe, MAX das Maximum, AVG den Mittelwert, SUM die Summe und COUNT die Anzahl der Werte.

### 1. Beispiel 39: Aggregierter Wert, Konstante und Ausdruck für Lehrer

```
SELECT    COUNT(*), 'konstante', 1, COUNT(*) * 10
FROM      lehrer;
```

Ausgabe:

Expr1000	Expr1001	Expr1002	Expr1003
17	konstante	1	170

COUNT(\*) liefert für die Tabelle lehrer 17, weil in der Tabelle lehrer insgesamt 17 Datensätze vorhanden sind. COUNT(\*) \* 10 stellt die Multiplikation des durch COUNT(\*) ermittelten Wertes mal 10 dar.

An dieser Stelle auch eine Zusammenfassung der arithmetischen Operatoren, die – wie auch in jeder anderen Programmiersprache - in SQL ebenso zur Anwendung kommen können.

+	für die Addition
-	für die Subtraktion
*	für die Multiplikation
/ f	für die Division
mod	für die Modulo-Operation <sup>10</sup>

<sup>10</sup> Die Modulooperation  $a \bmod b$  liefert den Rest der ganzzahligen Division zurück.



## 2. Beispiel 40: Aggregationswert, Konstante, Ausdruck für Lehrer mit Chef HI

```

SELECT    COUNT(*), 'konstante', 1, COUNT(*) * 10
FROM      lehrer
WHERE     l_l_chef = 'HI';

```

Ausgabe:

Expr1000	Expr1001	Expr1002	Expr1003
2	konstante	1	20

## 3. Beispiel 41: Rechnen mit dem Lehrergehalt

```

SELECT    COUNT(*) AS Zeilenanzahl,
          MAX(l_gehalt) AS Höchstes_Lehrergehalt,
          MIN(l_gehalt) AS Niedrigstes_Lehrergehalt,
          AVG(l_gehalt) AS Durchschnittliches_Lehrergehalt,
          SUM(l_gehalt) AS Summe_der_Lehrergehälter,
          MAX(l_gehalt) * 12 AS Höchstes_Gehalt_mal_12,
          (MAX(l_gehalt) + MIN(l_gehalt))/2 AS Mittleres_Lehrergehalt
FROM      lehrer;

```

Ausgabe:

Zeilenanzahl	Höchstes_L	Niedrigstes_L	Durchschnittliches_L	Summe	Höchstes_Gehalt_mal_12	Mittleres_Lehrergehalt
17	400	75	195	2730	4800	237,5

## 4. Beispiel 42: Geburtsdatum des jüngsten Schülers

```

SELECT    MAX(s_gebdat) AS Geburtsdatum_des_jüngsten_Schülers
FROM      schueler;

```

Ausgabe:

Geburtsdatum_des_jüngsten_Schülers
22.07.1993

## 5. Beispiel 43: Geburtsdatum des ältesten Schülers

```
SELECT    MIN(s_gebdat) AS Geburtsdatum_des_ältesten_Schülers
FROM      schueler;
```

Ausgabe:

Geburtsdatum_des_ältesten_Schülers
01.01.1962

## 6. Beispiel 44: Ausgabe des heutigen Datums

```
SELECT    DISTINCT Now() AS Heute
FROM      lehrer;
```

Ausgabe:

Heute
29.11.2003

Exkurs: das Systemdatum und einige Datumsfunktionen in unterschiedlichen DB-Systemen

Oracle:

	Kommentar
sysdate()	aktuelles Tagesdatum inkl. Uhrzeit
month_between()	Anzahl der Monate zwischen zwei Daten
to_char(date)	Umwandlung eines Datums in eine Zeichenkette
to_date(string)	Umwandlung einer Zeichenkette in ein Datum

MS SQL Server:

	Kommentar
getdate()	aktuelles Tagesdatum inkl. Uhrzeit
day()	Tagesanteil aus einem Datum
month()	Monatsanteil aus einem Datum
year()	Jahresanteil aus einem Datum



## MS Access:

	Kommentar
date()	aktuelles Tagesdatum ohne Zeitanteil
now()	aktuelles Tagesdatum inkl. Uhrzeit
day()	Tagesanteil aus einem Datum
month()	Monatsanteil aus einem Datum
year()	Jahresanteil aus einem Datum

## MySQL:

	Kommentar
current_date(), curdate()	aktuelles Tagesdatum ohne Zeitanteil
current_time(), curtime()	aktuelle Uhrzeit ohne Datumsanteil
sysdate(), now()	aktuelles Tagesdatum inkl. Uhrzeit
datediff()	Intervall zwischen zwei Datumswerten

## 7. Beispiel 45: Rechnen mit den Geburtsdaten der Lehrer

```

SELECT      l_name, l_gebdat, l_gebdat - 2 AS Geburtsdatum_minus_2,
            (now() - l_gebdat)/365.25 AS Alter_in_Jahren,
            now() - l_gebdat AS Alter_in_Tagen
FROM        lehrer
ORDER BY    4 DESC;

```

Ausgabe:

	l_name	l_gebdat	Geburtsdatum_minus_2	Alter_in_Jahren	Alter_in_Tagen
	Bilek	03.03.1932	01.03.1932	71,7425392298527	26203,96245370
	Lenau	22.02.1938	20.02.1938	65,7685488123305	24021,96245370
	Berger	02.03.1945	28.02.1945	58,7459615433366	21456,96245370
	Aichholzer	07.09.1946	05.09.1946	57,2291922072654	20902,96245370
	Makandreou	01.04.1947	30.03.1947	56,6651949451162	20696,96245370
	Radlbauer	11.11.1949	09.11.1949	54,0505474434051	19741,96245370
	Hanke	12.12.1950	10.12.1950	52,9663585317008	19345,96245370
	Pirkner	22.06.1955	20.06.1955	48,4406911805714	17692,96245370
	Preißl	05.07.1956	03.07.1956	47,4030457322483	17313,96245370
	Lorenz	12.12.1958	10.12.1958	44,9663585317008	16423,96245370
	Beringer	15.07.1961	13.07.1961	42,3763516870738	15477,96245370
	Renkin	19.08.1961	17.08.1961	42,2805269095242	15442,96245370
	Siegel				
	Hickel				
	Unger				
	Moitzi				
►	Stappler				

## Q. Verbund (Join / Inner Join)

Da in der Regel das Interesse besteht, Daten, die in unterschiedlichen Tabellen abgelegt sind, zu selektieren, ist der Verbund, im Englischen Join genannt, von äußerster Wichtigkeit.

Ein Verbund einer Tabelle kann

- mit einer anderen Tabelle bzw. mit mehreren anderen Tabellen gegeben sein
- mit sich selbst gegeben sein.

```

SELECT *
FROM tabellenname1, tabellenname2
WHERE tabellenname1.spaltenname1 = tabellenname2.spaltenname2;

oder

SELECT *
FROM tabellenname1 INNER JOIN, tabellenname2
ON tabellenname1.spaltenname1 = tabellenname2.spaltenname2;

```

Die zweite Darstellungsart wurde im Unterricht verwendet, die erste Darstellungsart werden Sie bei Ihrer weiterführenden Lektüre / Internet-Recherche oftmals wieder finden.

### 1. Beispiel 46: Schüler der Klasse 03TA

```

SELECT    s_schnr, s_vorname, s_name, s_k_klasse
FROM      schueler
WHERE     s_k_klasse = '03TA'
ORDER BY  s_name;

```

Ausgabe:

s_schnr	s_vorname	s_name	s_k_klasse
1	Richard	Adler	03TA
122	Gustav	Gans	03TA
16	Gustav	Geier	03TA
74	Helga	Hundertwasser	03TA
22	Konrad	Klassenbester	03TA
4	Robert	Rotkehlchen	03TA
19	Sebastian	Sitzenbleiber	03TA

## 2. Beispiel 47: Schüler der Klasse 03TA mit Klassenbezeichnung

```

SELECT    s_schnr, s_vorname, s_name, s_k_klasse, k_bez
FROM      schueler, klassen
WHERE     s_k_klasse = '03TA' AND s_k_klasse = k_id
ORDER BY  s_name;

```

oder

```

SELECT    schueler.s_schnr, schueler.s_vorname, schueler.s_name,
          klassen.s_k_klasse, klassen.k_bez
FROM      schueler INNER JOIN klassen
          ON schueler.s_k_klasse = klassen.k_id
WHERE     schueler.s_k_klasse = '03TA'
ORDER BY  s_name;

```

Ausgabe:

	s_schnr	s_vorname	s_name	s_k_klasse	k_bez
	1	Richard	Adler	03TA	Kolleg T 03TA
	122	Gustav	Gans	03TA	Kolleg T 03TA
	16	Gustav	Geier	03TA	Kolleg T 03TA
	74	Helga	Hundertwasser	03TA	Kolleg T 03TA
	22	Konrad	Klassenbester	03TA	Kolleg T 03TA
	4	Robert	Rotkehlchen	03TA	Kolleg T 03TA
	19	Sebastian	Sitzenbleiber	03TA	Kolleg T 03TA

Der Join zwischen der Tabelle schueler und der Tabelle klasse erfolgt auf der Basis der existierenden Fremdschlüsselbeziehung (Sie erinnern sich in diesem Zusammenhang im besten Fall auch an die Bedeutung des Begriffes 'referentielle Integrität') – durch `s_k_klasse = k_id` werden die beiden Tabellen aneinander gebunden.

## 3. Beispiel 48: Stundenplan für Montag in der 03TA mit Gegenstandsnamen

```

SELECT    k_bez AS Klasse,
          l_name AS Lehrername,
          g_bez AS Gegenstand,
          st_stunde AS Stunde,
          st_r_raum AS Raum
FROM      stunden, klassen, lehrer, gegenstaende
WHERE     st_stunde LIKE 'MO*' AND st_k_klasse = '03TA' AND
          st_k_klasse = k_id AND st_l_lehrer = l_id AND st_g_fach = g_id
ORDER BY  k_bez, st_stunde;

```

oder

```

SELECT    klassen.k_bez AS Klasse,
          lehrer.l_name AS Lehrername,
          gegenstaende.g_bez AS Gegenstand,
          stunden.st_stunde AS Stunde,
          stunden.st_r_raum AS Raum
FROM      ((klassen      INNER JOIN stunden
              ON stunden.st_k_klasse = klassen.k_id)
          INNER JOIN lehrer
              ON stunden.st_l_lehrer = lehrer.l_id)
          INNER JOIN gegenstaende
              ON stunden.st_g_fach = gegenstaende.g_id
WHERE     stunden.st_stunde LIKE 'MO*' AND stunden.st_k_klasse = '03TA'
ORDER BY  k_bez, st_stunde;

```

Ausgabe:

	Klasse	Lehrername	Gegenstand	Stunde	Raum
	Kolleg T 03TA	Lorenz	Technische Datenorganisation	MO1	B1
	Kolleg T 03TA	Lorenz	Technische Datenorganisation	MO2	B1
	Kolleg T 03TA	Lorenz	Technische Datenorganisation	MO3	B1
	Kolleg T 03TA	Beringer	System-Einsatzplanung	MO4	B1
	Kolleg T 03TA	Beringer	System-Einsatzplanung	MO5	B1
	Kolleg T 03TA	Preißl	Programmieren	MO7	LA1
►	Kolleg T 03TA	Preißl	Programmieren	MO8	LA1



## 4. Beispiel 49: Prüfungen, die der Klassenvorstand abhielt

```

SELECT  l_name AS Lehrer,
        s_name AS Schüler,
        p_datum AS Prüfungsdatum,
        g_bez AS Prüfungsfach,
        p_note AS Prüfungsnote
FROM    lehrer, schueler, klassen, pruefungen, gegenstaende
WHERE   s_schnr = p_s_kandidat AND
        l_id = p_l_pruefer AND
        k_id = s_k_klasse AND
        p_g_fach = g_id AND
        k_l_klavst = p_l_pruefer;

```

Ausgabe:

Lehrer	Schüler	Prüfungsdatum	Prüfungsfach	Prüfungsnote
Beringer	Adler	15.03.1999	Betriebliche Organisation	2
Beringer	Sitzenbleiber	15.04.1998	Technische Datenorganisation	5
Beringer	Klassenbester	12.01.1998	Technische Datenorganisation	2
Beringer	Klassenbester	15.03.1999	Betriebliche Organisation	4
Bilek	Bartgeier	12.01.1998	Technische Datenorganisation	1
Preißl	Apfelbaum	01.10.2003	Datenbanksysteme	1
Preißl	Birnenbaum	01.10.2003	Datenbanksysteme	2

## 5. Beispiel 50: Schüleranzahl der 03TB

```

SELECT    COUNT(*) AS Schüleranzahl
FROM      schueler
WHERE     s_k_klasse = '03TB';

```

Ausgabe:

Schüleranzahl
7

## 6. Beispiel 51: Schüleranzahl der 03TB mit Klassenlangname

```

SELECT    COUNT(*) AS Schüleranzahl, MAX(k_bez) AS Klassenlangname
FROM      schueler, klassen
WHERE     k_id = s_k_klasse AND s_k_klasse = '03TB';

```

oder

```

SELECT    COUNT(*) AS Schüleranzahl, MAX(klassen.k_bez) AS Klassenlangname
FROM      schueler INNER JOIN klassen
          ON klassen.k_id = schueler.s_k_klasse
WHERE     schueler.s_k_klasse = '03TB';

```

Ausgabe:

Schüleranzahl	Klassenlangname
7	Kolleg T 03TB

## 7. Beispiel 52: Sitzplätze der Lehrsäle und Labors der 03TA (mit Duplikaten)

```

SELECT    r_id AS Raum_ID,
          r_plaetze AS Platzanzahl,
          k_bez AS Langbezeichnung
FROM      raeume, stunden, klassen
WHERE     k_id = st_k_klasse AND r_id = st_r_raum AND k_id = '03TA'
ORDER BY  r_id, r_plaetze DESC;

```

oder

```

SELECT    raeume.r_id AS Raum_ID,
          raeume.r_plaetze AS Platzanzahl,
          klassen.k_bez AS Langbezeichnung
FROM      (klassen      INNER JOIN stunden ON klassen.k_id = stunden.st_k_klasse)
          INNER JOIN raeume ON stunden.st_r_raum = raeume.r_id
WHERE     klassen.k_id = '03TA'
ORDER BY  raeume.r_id, raeume.r_plaetze DESC;

```

Ausgabe:

Raum_ID	Platzanzahl	Langbezeichnung
B1	22	Kolleg T 03TA
B1	22	Kolleg T 03TA
B1	22	Kolleg T 03TA
B1	22	Kolleg T 03TA
B1	22	Kolleg T 03TA
B1	22	Kolleg T 03TA
B1	22	Kolleg T 03TA
B1	22	Kolleg T 03TA
B1	22	Kolleg T 03TA
B1	22	Kolleg T 03TA
B1	22	Kolleg T 03TA
B1	22	Kolleg T 03TA
B1	22	Kolleg T 03TA
B1	22	Kolleg T 03TA
LA1	10	Kolleg T 03TA
LA1	10	Kolleg T 03TA
LA1	10	Kolleg T 03TA
LA1	10	Kolleg T 03TA
LA2	18	Kolleg T 03TA
LA2	18	Kolleg T 03TA
LA3	2	Kolleg T 03TA
LA3	2	Kolleg T 03TA
LA3	2	Kolleg T 03TA

## 8. Beispiel 53: Sitzplätze der Lehrsäle und Labors der 03TA (ohne Duplikate)

```

SELECT    DISTINCT r_id AS Raum_ID,
           r_plaetze AS Platzanzahl,
           k_bez AS Langbezeichnung
FROM      raeume, stunden, klassen
WHERE     k_id = st_k_klasse AND r_id = st_r_raum AND k_id = '03TA'
ORDER BY  r_id, r_plaetze DESC;

```

oder

```

SELECT    DISTINCT raeume.r_id AS Raum_ID,
           raeume.r_plaetze AS Platzanzahl,
           klassen.k_bez AS Langbezeichnung
FROM      (klassen      INNER JOIN stunden ON klassen.k_id = stunden.st_k_klasse)
           INNER JOIN raeume ON stunden.st_r_raum = raeume.r_id
WHERE     klassen.k_id = '03TA'
ORDER BY  raeume.r_id, raeume.r_plaetze DESC;

```

Ausgabe:

	Raum_ID	Platzanzahl	Langbezeichnung
	B1	22	Kolleg T 03TA
	LA1	10	Kolleg T 03TA
	LA2	18	Kolleg T 03TA
	LA3	2	Kolleg T 03TA

## R. Alias für Tabellennamen

Eine Tabelle kann nicht nur mit einer anderen Tabelle verbunden werden, sondern auch mit sich selbst.

Selbstverständlich muss zu jedem Zeitpunkt klar sein, auf welche Tabelle sich ein Teil eines SQL Statements bezieht: aus diesem Grund kommen Alias-Namen zum Einsatz, konkret wird im FROM nach der Bezeichnung der Tabelle der Alias-Name, getrennt durch AS, angegeben.

```
SELECT alias1.spaltenname, alias2.spaltenname  
FROM tabellenname AS alias1, tabellenname AS alias2  
WHERE bedingung;
```

1. Beispiel 54: Lehrer mit Gehalt so hoch wie vom direkten Vorgesetzten

```
SELECT  chef.l_id AS Vorgesetzten_ID,
        chef.l_gehalt AS Vorgesetzten_Gehalt,
        untergebener.l_id AS Untergebenen_ID,
        untergebener.l_gehalt AS Untergebenen_Gehalt
FROM    lehrer AS chef, lehrer AS untergebener
WHERE   untergebener.l_l_chef = chef.l_id AND
        untergebener.l_gehalt = chef.l_gehalt;
```

oder

```
SELECT  chef.l_id AS Vorgesetzten_ID,
        chef.l_gehalt AS Vorgesetzten_Gehalt,
        untergebener.l_id AS Untergebenen_ID,
        untergebener.l_gehalt AS Untergebenen_Gehalt
FROM    lehrer AS chef INNER JOIN lehrer AS untergebener
        ON untergebener.l_l_chef = chef.l_id
WHERE   untergebener.l_gehalt = chef.l_gehalt;
```

Ausgabe:

Vorgesetzten_ID	Vorgesetzten_Gehalt	Untergebenen_ID	Untergebenen_Gehalt
HA	300,00	LO	300,00

## 2. Beispiel 55: Lehrer mit Gehalt kleiner gleich dem des direkten Vorgesetzten

```

SELECT  chef.l_id AS Vorgesetzten_ID,
        chef.l_gehalt AS Vorgesetzten_Gehalt,
        untergebener.l_id AS Untergebenen_ID,
        untergebener.l_gehalt AS Untergebenen_Gehalt
FROM    lehrer AS chef, lehrer AS untergebener
WHERE   untergebener.l_l_chef = chef.l_id AND
        untergebener.l_gehalt <= chef.l_gehalt;

```

oder

```

SELECT  chef.l_id AS Vorgesetzten_ID,
        chef.l_gehalt AS Vorgesetzten_Gehalt,
        untergebener.l_id AS Untergebenen_ID,
        untergebener.l_gehalt AS Untergebenen_Gehalt
FROM    lehrer AS chef INNER JOIN lehrer AS untergebener
        ON untergebener.l_l_chef = chef.l_id
WHERE   untergebener.l_gehalt <= chef.l_gehalt;

```

Ausgabe:

Vorgesetzten_ID	Vorgesetzten_Gehalt	Untergebenen_ID	Untergebenen_Gehalt
PI	220,00	RX	180,00
LN	180,00	AI	150,00
HA	300,00	BA	220,00
HA	300,00	BI	280,00
BF	400,00	HA	300,00
HA	300,00	LO	300,00
HA	300,00	LN	180,00
LN	180,00	MM	120,00
BF	400,00	PI	220,00
HA	300,00	PS	100,00
PI	220,00	RR	115,00
BF	400,00	UK	90,00
BF	400,00	AM	75,00

## 3. Beispiel 56: Liste aller Vorgesetztenbeziehungen

```

SELECT  chef.l_name AS Name_des_Chefs,
        v_art AS Vorgesetztenyp,
        untergebener.l_name AS Name_des_Untergebenen
FROM    lehrer AS chef, vorgesetzte, lehrer AS untergebener
WHERE   chef.l_id = vorgesetzte.v_l_vorg AND untergebener.l_id = vorgesetzte.v_l_unt
ORDER BY chef.l_name ASC, untergebener.l_name;

```

oder

```

SELECT  chef.l_name AS Name_des_Chefs,
        v_art AS Vorgesetztenyp,
        untergebener.l_name AS Name_des_Untergebenen
FROM    (lehrer AS chef INNER JOIN vorgesetzte
        ON chef.l_id = vorgesetzte.v_l_vorg)
        INNER JOIN lehrer AS untergebener
        ON untergebener.l_id = vorgesetzte.v_l_unt
ORDER BY chef.l_name ASC, untergebener.l_name;

```

Ausgabe:

	Name_des_Chefs	Vorgesetztenyp	Name_des_Untergebenen
	Berger	AV	Moitzi
	Berger	AV	Unger
	Beringer	Fgrp Prog	Moitzi
	Beringer	Fgrp Prog	Preißl
	Beringer	Fgrp Prog	Unger
	Bilek	Fgrp RW	Lorenz
	Hanke	AV	Beringer
	Hanke	AV	Bilek
	Hanke	AV	Lorenz
	Hanke	AV	Preißl
	Hickel	Direktor	Berger
	Hickel	Direktor	Beringer
	Hickel	Direktor	Bilek
	Hickel	Direktor	Hanke
	Hickel	Direktor	Lorenz
	Hickel	Direktor	Makandreou
	Hickel	Direktor	Moitzi
	Hickel	Direktor	Preißl
	Hickel	Direktor	Stappler
	Hickel	Direktor	Unger
	Stappler	AV	Beringer
►	Stappler	AV	Makandreou



## S. Datumswerte

1. Beispiel 57: Ausgabe der Schüler, die im Dezember Geburtstag haben

```
SELECT    s_name, s_vorname, s_gebdat
FROM      schueler
WHERE     month(s_gebdat) = '12'
ORDER BY  s_name;
```

Ausgabe:

s_name	s_vorname	s_gebdat
Apfelbaum	Anna	12.12.1965
Birnenbaum	Berta	12.12.1965
Picasso	Paul	03.12.1980

2. Beispiel 58: Ausgabe der Schüler, die am 3.12.1980 Geburtstag haben

```
SELECT    s_name, s_vorname, s_gebdat
FROM      schueler
WHERE     day(s_gebdat) = '3' AND month(s_gebdat) = '12' AND year(s_gebdat) = '1980';
```

Ausgabe:

s_name	s_vorname	s_gebdat
Picasso	Paul	03.12.1980

## 3. Beispiel 59: Schüler, die jünger als Lehrer HA sind

```

SELECT      s_name AS Schüler, year(s_gebdat) AS Schülergeburtsjahr,
            l_name AS Lehrer, year(l_gebdat) AS Lehrergeburtsjahr
FROM        lehrer, schueler
WHERE       year(s_gebdat) > year(l_gebdat) AND l_id = 'HA'
ORDER BY   s_gebdat;

```

Ausgabe:

Schüler	Schülergeburtsjahr	Lehrer	Lehrergeburtsjahr
Gans	1962	Hanke	1950
Tanne	1962	Hanke	1950
Fichte	1962	Hanke	1950
Schwan	1963	Hanke	1950
Kirschebaum	1963	Hanke	1950
Marillenbaum	1964	Hanke	1950
Birnenbaum	1965	Hanke	1950
Apfelbaum	1965	Hanke	1950
Kiefer	1969	Hanke	1950
Föhre	1972	Hanke	1950
Adler	1973	Hanke	1950
Rotkehlchen	1977	Hanke	1950
Geier	1978	Hanke	1950
Bartgeier	1980	Hanke	1950
Buche	1980	Hanke	1950
Hundertwasser	1980	Hanke	1950
Picasso	1980	Hanke	1950
Eibe	1982	Hanke	1950
Pollak	1982	Hanke	1950
Sitzenbleiber	1983	Hanke	1950
Ente	1983	Hanke	1950
Geröllheimer	1984	Hanke	1950
Klassenbester	1985	Hanke	1950
Feuerstein	1993	Hanke	1950

## T. Gruppierung

In den vorangegangenen Beispielen zum Thema ‚Aggregatfunktionen‘ wurde immer ein Wert berechnet.

GROUP BY schafft die Möglichkeit, für jede Gruppe genau einen Ausgabesatz zu ermitteln. In diesem Zusammenhang fällt auch der Begriff ‚Gruppenwechsel‘.

```
SELECT aggregatfunktion(spaltenname1), spaltenname2
      FROM tabellenname
      GROUP BY spaltenname2;
```

### 1. Beispiel 60: Schüleranzahl

```
SELECT COUNT(s_schnr) AS Schüleranzahl
FROM   schueler;
```

Ausgabe:

Schüleranzahl
24

### 2. Beispiel 61: Schüleranzahl je Klasse

```
SELECT COUNT(s_schnr) AS Schüleranzahl, s_k_klasse AS Klasse
FROM   schueler
GROUP BY s_k_klasse;
```

Ausgabe:

Schüleranzahl	Klasse
7	03TA
7	03TB
6	0M5A
4	0M5Q

## U. Gruppeneinschränkung

Die ermittelten Ausgabedatensätze der Gruppierung können mittels HAVING nochmals eingeschränkt werden.

```
SELECT aggregatfunktion(spaltenname1), spaltenname2
      FROM tabellenname
      WHERE bedingung
      GROUP BY spaltenname2
HAVING aggregatfunktion(spaltenname1) vergleichsoperator vergleichswert;
```

### 1. Beispiel 62: Klassen mit mehr als 6 Schülern

```
SELECT    s_k_klasse AS Klasse, COUNT(*) AS Schüleranzahl
FROM      schueler
GROUP BY  s_k_klasse
HAVING    COUNT(*) > 6;
```

Ausgabe:

	Klasse	Schüleranzahl
	03TA	7
▶	03TB	7

### 2. Beispiel 63: Schüleranzahl je Jahrgang mit mehr als einem Schüler

```
SELECT    year(s_gebdat) AS Jahrgang, COUNT(*) AS Schüleranzahl
FROM      schueler
GROUP BY  year(s_gebdat)
HAVING    COUNT(*) > 1;
```

Ausgabe:

	Jahrgang	Schüleranzahl
	1962	3
	1963	2
	1965	2
	1980	4
	1982	2
*	1983	2

### 3. Beispiel 64: Jahrgänge ab 1975 mit mehr als einem Schüler

```

SELECT    year(s_gebdat) AS Jahrgang, COUNT(*) AS Schüleranzahl
FROM      schueler
WHERE     s_gebdat >= #01/01/1975#
GROUP BY  year(s_gebdat)
HAVING    COUNT(*) > 1
ORDER BY  1;

```

Ausgabe:

	Jahrgang	Schüleranzahl
	1980	4
	1982	2
▶	1983	2

WHERE und HAVING dürfen nicht verwechselt werden!

#### 4. Beispiel 65: Jahrgänge ab 1975

```

SELECT    year(s_gebdat) AS Jahrgang, COUNT(*) AS Schüleranzahl
FROM      schueler
WHERE     s_gebdat >= #01/01/1975#
GROUP BY  year(s_gebdat)
ORDER BY  1;

```

Ausgabe:

	Jahrgang	Schüleranzahl
	1977	1
	1978	1
	1980	4
	1982	2
	1983	2
	1984	1
	1985	1
▶	1993	1

## 5. Beispiel 66: Vornamenhäufigkeit bei Lehrern (mit Nullwert)

```

SELECT    l_vorname AS Vorname, COUNT(*) AS Vorkommen
FROM      lehrer
GROUP BY  l_vorname
ORDER BY  COUNT(*);

```

Ausgabe:

Vorname	Vorkommen
Klaus	1
Alfred	1
Andrea	1
Franz	1
Gustav	1
Hans	1
Johann	1
Wolfgang	1
Margit	1
Max	1
Nikolaus	1
Rudi	1
Sonja	1
Walter	1
Heinz	1
	1

## 6. Beispiel 67: Vornamenhäufigkeit bei Lehrern (ohne Nullwert)

```

SELECT    l_vorname AS Vorname, COUNT(*) AS Vorkommen
FROM      lehrer
WHERE     l_vorname IS NOT NULL
GROUP BY  l_vorname
ORDER BY  COUNT(*);

```

Ausgabe:

Vorname	Vorkommen
Wolfgang	1
Walter	1
Sonja	1
Rudi	1
Nikolaus	1
Max	1
Margit	1
Klaus	1
Johann	1
Heinz	1
Hans	1
Gustav	1
Franz	1
Andrea	1
Alfred	1

## 7. Beispiel 68: Lehrsäle mit mehr als 5 Stunden Unterricht

```

SELECT    st_r_raum AS Raum, COUNT(*) AS Stundenanzahl
FROM      stunden
GROUP BY  st_r_raum
HAVING    COUNT(*) > 5
ORDER BY  COUNT(*) DESC;

```

Ausgabe:

Raum	Stundenanzahl
B2	17
B1	16
LA2	8
LA1	8

## 8. Beispiel 69: Vorgesetzte Lehrer mit Durchschnittsgehalt ihrer Untergebenen

```

SELECT    vorgesetzter.l_id AS Vorgesetzter,
          COUNT(*) AS Anzahl_Untergebene,
          AVG(untergebene.l_gehalt) AS Durchschnittsgehalt_der_Untergebenen
FROM      lehrer AS vorgesetzter, lehrer AS untergebene, vorgesetzte
WHERE     vorgesetzter.l_id = v_l_vorg AND v_l_unt = untergebene.l_id
GROUP BY  vorgesetzter.l_id
ORDER BY  2;

```

Ausgabe:

Vorgesetzter	Anzahl_Untergebene	Durchschnittsgehalt_der_Untergebenen
BI	1	300
SS	2	170
BF	2	82,5
BA	3	88,33333333333333
HA	4	225
HI	10	209,44444444444444

## 9. Beispiel 70: Vorgesetzte mit Durchschnittsgehalt &gt; 150 der Untergebenen

```

SELECT    vorgesetzter.l_id AS Vorgesetzter,
          COUNT(*) AS Anzahl_Untergebene,
          AVG(untergebene.l_gehalt) AS Durchschnittsgehalt_der_Untergebenen
FROM      lehrer AS vorgesetzter, lehrer AS untergebene, vorgesetzte
WHERE     vorgesetzter.l_id = v_l_vorg AND v_l_unt = untergebene.l_id
GROUP BY  vorgesetzter.l_id
HAVING    AVG(untergebene.l_gehalt) > 150
ORDER BY  2;

```

Ausgabe:

Vorgesetzter	Anzahl_Untergebene	Durchschnittsgehalt_der_Untergebenen
BI	1	300
SS	2	170
HA	4	225
HI	10	209,44444444444444



## 10. Beispiel 71: Unterrichtsstunden je Lehrer

```

SELECT    MAX(l_name) AS Lehrer, l_id AS Lehrerkürzel, COUNT(*) AS Stundenanzahl
FROM      lehrer, stunden
WHERE     l_id = st_l_lehrer
GROUP BY  l_id
ORDER BY  1;

```

oder

```

SELECT    MAX(lehrer.l_name) AS Lehrer, lehrer.l_id AS Lehrerkürzel,
COUNT(*) AS Stundenanzahl
FROM      lehrer INNER JOIN stunden ON lehrer.l_id = stunden.st_l_lehrer
GROUP BY  lehrer.l_id
ORDER BY  1;

```

Ausgabe:

	Lehrer	Lehrerkürzel	Stundenanzahl
	Berger	BF	9
	Beringer	BA	6
	Bilek	BI	11
	Hanke	HA	7
	Lorenz	LO	5
	Makandreou	MM	2
	Moitzi	AM	1
	Preißl	PS	9
▶	Unger	UK	2

## 11. Beispiel 72: Lehrer, die mehr als 10 Stunden unterrichten

```

SELECT    MAX(l_name) AS Lehrer, l_id AS Lehrerkürzel, COUNT(*) AS Stundenanzahl
FROM      lehrer, stunden
WHERE     l_id = st_l_lehrer
GROUP BY  l_id
HAVING    COUNT(*) > 10
ORDER BY  1;

```

oder

```

SELECT    MAX(lehrer.l_name) AS Lehrer, lehrer.l_id AS Lehrerkürzel,
COUNT(*) AS Stundenanzahl
FROM      lehrer INNER JOIN stunden ON lehrer.l_id = stunden.st_l_lehrer
GROUP BY  lehrer.l_id
HAVING    COUNT(*) > 10
ORDER BY  1;

```

Ausgabe:

	Lehrer	Lehrerkürzel	Stundenanzahl
▶	Bilek	Bl	11

## 12. Beispiel 73: Stunden mit mehr Schülern als Sitzplätzen

```

SELECT  st_r_raum,
        MAX(r_plaetze) AS Platzanzahl,
        k_id AS Klasse,
        st_stunde AS Stunde,
        COUNT(*) AS Schüleranzahl
FROM    raeume, stunden, klassen, schueler
WHERE   st_r_raum = r_id AND st_k_klasse = k_id AND k_id = s_k_klasse
GROUP BY st_r_raum, st_stunde, k_id
HAVING  (MAX(r_plaetze) - COUNT(*)) < 0;

```

oder

```

SELECT  stunden.st_r_raum,
        MAX(raeume.r_plaetze) AS Platzanzahl,
        klassen.k_id AS Klasse,
        stunden.st_stunde AS Stunde,
        COUNT(*) AS Schüleranzahl
FROM    ((stunden      INNER JOIN klassen ON stunden.st_k_klasse = klassen.k_id)
        INNER JOIN schueler ON klassen.k_id = schueler.s_k_klasse)
        INNER JOIN raeume ON stunden.st_r_raum = raeume.r_id
GROUP BY stunden.st_r_raum, stunden.st_stunde, klassen.k_id
HAVING  (MAX(raeume.r_plaetze) - COUNT(*)) < 0;

```

Ausgabe:

	st_r_raum	Platzanzahl	Klasse	Stunde	Schüleranzahl
	LA3	2	03TA	DI3	7
	LA3	2	03TA	DI4	7
▶	LA3	2	03TA	DI5	7

## V. Subselect

Ein Subselect, im Deutschen Unterabfrage, liegt vor, wenn ein gleichsam untergeordnetes SELECT-Statement im Bedingungsteil eines ‚höheren‘ SELECT-Statements auftritt.

```
SELECT spaltenname1
      FROM tabellenname1
WHERE spaltenname1 vergleichsoperator (SELECT spaltenname2 FROM tabellenname2);
```

### 1. Beispiel 74: Geburtsdatum und Name des jüngsten Schülers

```
SELECT  s_vorname AS Vorname, s_name AS Name, s_gebdat AS Geburtsdatum
FROM    schueler
WHERE   s_gebdat = (SELECT MAX(s_gebdat) FROM schueler);
```

Ausgabe:

	Vorname	Name	Geburtsdatum
	Bebbles	Feuerstein	02.07.1993

## 2. Beispiel 75: Schüler, die bereits Prüfungen absolviert haben (Subselect)

```
SELECT  DISTINCT s_vorname AS Vorname, s_name AS Name
FROM    schueler
WHERE   s_schnr IN (SELECT p_s_kandidat FROM pruefungen);
```

Ausgabe:

Vorname	Name
Anna	Apfelbaum
Berta	Birnenbaum
Burghard	Bartgeier
Eberhard	Ente
Franz	Föhre
Fridolin	Fichte
Karoline	Kirschebaum
Kasimir	Kiefer
Konrad	Klassenbester
Martha	Marillenbaum
Paul	Picasso
Richard	Adler
Robert	Rotkehlchen
Sebastian	Sitzenbleiber
Susanne	Schwan
Theodor	Tanne

## 3. Beispiel 76: Schüler mit absolvierten Prüfungen (Join)

```

SELECT    DISTINCT s_vorname AS Vorname,
           s_name AS Name
FROM      schueler, pruefungen
WHERE     s_schnr = p_s_kandidat;

```

oder

```

SELECT    DISTINCT schueler.s_vorname AS Vorname,
           schueler.s_name AS Name
FROM      schueler INNER JOIN pruefungen
           ON schueler.s_schnr = pruefungen.p_s_kandidat;

```

Ausgabe:

	Vorname	Name
	Anna	Apfelbaum
	Berta	Birnenbaum
	Burghard	Bartgeier
	Eberhard	Ente
	Franz	Föhre
	Fridolin	Fichte
	Karoline	Kirschebaum
	Kasimir	Kiefer
	Konrad	Klassenbester
	Martha	Marillenbaum
	Paul	Picasso
	Richard	Adler
	Robert	Rotkehlchen
	Sebastian	Sitzenbleiber
	Susanne	Schwan
►	Theodor	Tanne

## 4. Beispiel 77: Klassensprecher

```

SELECT      *
FROM        schueler
WHERE       s_schnr IN (SELECT k_s_klaspr FROM klassen);

```

Ausgabe:

S_SCHNR	S_Name	S_Vorname	S_Gebdat	S_Adresse	S_K_Klasse
1	Adler	Richard	12.03.1973	Josefstadt	03TA
1266	Schwan	Susanne	23.04.1963	Ottakring	03TB
2222	Apfelbaum	Anna	12.12.1965	Tulln	0M5Q
3333	Tanne	Theodor	09.09.1962	Brunn/Gebirge	0M5A

## 5. Beispiel 78: Klassensprecher der Klasse mit BA als Klassenvorstand

```

SELECT      s_schnr, s_vorname, s_name, s_k_klasse
FROM        schueler
WHERE       s_schnr IN
            (SELECT k_s_klaspr FROM klassen WHERE k_l_klavst = 'BA');

```

Ausgabe:

s_schnr	s_vorname	s_name	s_k_klasse
1	Richard	Adler	03TA

## 6. Beispiel 79: Klassensprecher (mit ANY)

```

SELECT      *
FROM        schueler
WHERE       s_schnr = ANY (SELECT k_s_klaspr FROM klassen);

```

Ausgabe:

S_SCHNR	S_Name	S_Vorname	S_Gebdat	S_Adresse	S_K_Klasse
1	Adler	Richard	12.03.1973	Josefstadt	03TA
1266	Schwan	Susanne	23.04.1963	Ottakring	03TB
2222	Apfelbaum	Anna	12.12.1965	Tulln	0M5Q
3333	Tanne	Theodor	09.09.1962	Brunn/Gebirge	0M5A

```

SELECT *
FROM tabellenname
WHERE spaltenname vergleichsoperator ANY (subselect);

```

Es kann vereinfacht auch gesagt werden: wenn irgendein Einzelvergleich true ist, so ist das Ergebnis von ANY true.

Der Operator IN ist äquivalent zum Operator = ANY, d.h. wird im WHERE des ‚höheren‘ SELECTs mittels = auf ANY verglichen, so entspricht dies einem WHERE spaltenname IN werteliste.



## 7. Beispiel 80: Lehrer mit dem höchsten bekannten Gehalt (mit ALL)

```

SELECT      *
FROM        lehrer
WHERE       l_gehalt >= ALL      (SELECT l_gehalt
                                FROM lehrer
                                WHERE l_gehalt IS NOT NULL);

```

Ausgabe:

L_ID	L_Name	L_Vorname	L_Gebdat	L_Gehalt	L_L_Chef
BF	Berger	Franz	02.03.1945	400,00	HI

```

SELECT *
FROM tabellenname
WHERE spaltenname vergleichsoperator ALL (subselect);

```

ALL bedeutet, dass die innerhalb des Subselects gewonnenen Resultate AND-verknüpft und in den Bedingungsteil des ‚höheren‘ SELECTs eingesetzt werden. Vereinfacht gesagt: nur wenn alle Einzelvergleiche true sind, so ist das Ergebnis von ALL true.

## 8. Beispiel 81: Lehrer mit dem niedrigsten bekannten Gehalt

```

SELECT      *
FROM        lehrer
WHERE       l_gehalt <= ALL      (SELECT      l_gehalt
                                FROM          lehrer
                                WHERE         l_gehalt IS NOT NULL);

```

Ausgabe:

L_ID	L_Name	L_Vorname	L_Gebdat	L_Gehalt	L_L_Chef
AM	Moitzi	Andrea		75,00	BF

## 9. Beispiel 82: Klassensprecher und Klassensprecherstellvertreter (EXISTS)

```

SELECT    s_name, s_vorname, s_k_klasse
FROM      schueler
WHERE     EXISTS (SELECT * FROM klassen WHERE k_s_klaspr = s_schnr) OR
          EXISTS (SELECT * FROM klassen WHERE k_s_klasprstv = s_schnr)
ORDER BY  s_name;

```

Ausgabe:

s_name	s_vorname	s_k_klasse
Adler	Richard	03TA
Apfelbaum	Anna	0M5Q
Birnenbaum	Berta	0M5Q
Eibe	Engelbert	0M5A
Ente	Eberhard	03TB
Gans	Gustav	03TA
Schwan	Susanne	03TB
Tanne	Theodor	0M5A

```

SELECT *
FROM tabellenname
WHERE EXISTS (subselect);

```

Wie das Schlüsselwort EXISTS bereits vermuten lässt, ist das Resultat des Subselects true, wenn das Subselect zumindest einen Wert liefert, bzw. false, wenn die Ergebnismenge leer ist.

## 10. Beispiel 83: Ungeprüfte Gegenstände

```

SELECT      *
FROM        gegenstaende
WHERE       g_id NOT IN (SELECT DISTINCT p_g_fach FROM pruefungen)
ORDER BY   g_id;

```

Ausgabe:

G_ID	G_Bez
DE	Deutsch
PRRV	Prozessrechnerverbung
SEP	System-Einsatzplanung

## 11. Beispiel 84: Unterrichtende, aber nicht prüfende Lehrer

```

SELECT      l_name, l_vorname
FROM        lehrer
WHERE       l_id IN (SELECT st_l_lehrer FROM stunden) AND
            NOT EXISTS (SELECT * FROM pruefungen WHERE l_id = p_l_pruefer)
ORDER BY   l_name;

```

Ausgabe:

Name	Vorname
Lorenz	

## 12. Beispiel 85: Falsche Fremdschlüssel im Stundenplan

```

SELECT *
FROM stunden
WHERE NOT EXISTS (SELECT * FROM lehrer WHERE l_id = st_l_lehrer) OR
       NOT EXISTS (SELECT * FROM klassen WHERE k_id = st_k_klasse) OR
       NOT EXISTS (SELECT * FROM gegenstaende WHERE g_id = st_g_fach) OR
       NOT EXISTS (SELECT * FROM raeume WHERE r_id = st_r_raum);

```

Ausgabe:

ST_K_Klasse	ST_L_Lehrer	ST_G_Fach	ST_Stunde	ST_R_Raum

Das Ergebnis dieser Abfrage ist leer, da in der Schuldatenbank für alle Beziehungen die Forderung nach referentieller Integrität<sup>11</sup> (auch Beziehungsintegrität genannt) gegeben ist. Prüfen Sie ruhig in der Relationship-Ansicht die bei den einzelnen Beziehungen spezifizierten Rahmenbedingungen.



<sup>11</sup> Referentielle Integrität besagt, dass für jeden Fremdschlüsselwert (in der Tabelle A) der entsprechende Primärschlüsselwert (in der Tabelle B) vorhanden sein muss.

## 13. Beispiel 86: Lehrer mit nicht existierenden Vorgesetzten

```

SELECT      *
FROM        lehrer
WHERE       l_l_chef NOT IN (SELECT l_id FROM lehrer);

```

Ausgabe:

L_ID	L_Name	L_Vorname	L_Gebdat	L_Gehalt	L_L_Chef

## 14. Beispiel 87: Lehrer mit gleichem Gehalt

```

SELECT      *
FROM        lehrer
WHERE       l_gehalt = ANY      (SELECT l_gehalt
                                FROM lehrer
                                GROUP BY l_gehalt
                                HAVING COUNT(*) > 1)

ORDER BY l_gehalt, l_id;

```

Ausgabe:

L_ID	L_Name	L_Vorname	L_Gebdat	L_Gehalt	L_L_Chef
LN	Lenau	Nikolaus	22.02.1938	180,00	HA
RX	Renkin	Max	19.08.1961	180,00	PI
BA	Beringer	Alfred	15.07.1961	220,00	HA
PI	Pirkner	Walter	22.06.1955	220,00	BF
HA	Hanke	Gustav	12.12.1950	300,00	BF
LO	Lorenz		12.12.1958	300,00	HA

## 15. Beispiel 88: Lehrer mit dem zweithöchsten Gehalt

```
SELECT  l_id AS Lehrer, l_name AS Nachname, l_gehalt AS Gehalt
FROM    lehrer
WHERE   l_gehalt = (SELECT MAX(l_gehalt)
                   FROM lehrer
                   WHERE l_id NOT IN
                     (SELECT l_id
                      FROM lehrer
                      WHERE l_gehalt = (SELECT MAX(l_gehalt) FROM
lehrer)));
```

Ausgabe:

Lehrer	Nachname	Gehalt
HA	Hanke	300,00
LO	Lorenz	300,00

## W. Outer Join

Neben dem oben beschriebenen ‘normalen’ (inneren) Verbund, ist der äußere Verbund (outer join) nicht minder interessant.

```
SELECT *
FROM tabellenname1 RIGHT JOIN tabellenname2
ON tabellenname1.spaltenname1 = tabellenname2.spaltenname2;
```

Ein RIGHT OUTER Join gibt alle Datensätze aus der rechten Tabelle zurück – selbst wenn keine dazu passenden Datensätze in der linken Tabelle existieren.

Ein LEFT OUTER Join gibt alle Datensätze aus der linken Tabelle zurück – selbst wenn keine dazu passenden Datensätze in der rechten Tabelle existieren.

### 1. Beispiel 89: Notendurchschnitt aller Gegenstände

```
SELECT    gegenstaende.g_id AS Gegenstandskürzel,
          MAX(gegenstaende.g_bez) AS Langbezeichnung,
          AVG(pruefungen.p_note) AS Durchschnittsnote
FROM      gegenstaende LEFT JOIN pruefungen
          ON gegenstaende.g_id = pruefungen.p_g_fach
GROUP BY  gegenstaende.g_id;
```

Ausgabe:

Gegenstandskürzel	Langbezeichnung	Durchschnittsnote
BO	Betriebliche Organisation	3
BSYS	Betriebssysteme	3,4
DBSYS	Datenbanksysteme	1,33333333333333
DE	Deutsch	
EN	Englisch	4
MA	Mathematik	3
PR	Programmieren	2
PRRV	Prozessrechnerverbung	
RW	Rechnungswesen	1
SEP	System-Einsatzplanung	
TD0	Technische Datenorganisation	2,625

Ein einfacher, innerer Verbund (... FROM gegenstaende INNER JOIN pruefungen ON ...) würde nur die Durchschnittsnoten derjenigen Gegenstände auflisten, zu denen auch Prüfungseinträge in der Tabelle pruefungen abgespeichert sind.

Durch den hier dargestellten äußeren Verbund werden auch diejenigen Einträge aus der Tabelle gegenstaende berücksichtigt, zu denen es keine Entsprechung in der Tabelle pruefungen gibt.

## 2. Beispiel 90: Notendurchschnitt aller Schüler

```

SELECT    schueler.s_schnr,
          MAX(schueler.s_name) AS Nachname,
          COUNT(pruefungen.p_s_kandidat) AS Prüfungsanzahl,
          AVG(pruefungen.p_note) AS Durchschnittsnote
FROM      schueler LEFT JOIN pruefungen
          ON schueler.s_schnr = pruefungen.p_s_kandidat
GROUP BY  schueler.s_schnr;

```

Ausgabe:

s_schnr	Nachname	Prüfungsanzahl	Durchschnittsnote
1	Adler	4	2,25
3	Bartgeier	1	1
4	Rotkehlchen	1	3
16	Geier	0	
19	Sitzenbleiber	1	5
22	Klassenbester	2	3
55	Picasso	1	5
74	Hundertwasser	0	
77	Pollak	0	
84	Feuerstein	0	
88	Geröllheimer	0	
111	Ente	3	3
122	Gans	0	
1266	Schwan	1	3
2222	Apfelbaum	3	1,33333333333333
2223	Birnenbaum	2	3,5
2224	Marillenbaum	3	2,33333333333333
2225	Kirschebaum	3	3,66666666666667
3333	Tanne	2	1
3334	Fichte	1	2
3335	Föhre	1	1
3336	Kiefer	2	3
3337	Buche	0	
3338	Eibe	0	



## X. Mengenoperationen

Aus der Mathematik ist Ihnen die Mengenlehre – mit den Operationen der Vereinigung, des Durchschnitts und der Differenz - sicher ein Begriff.

```
SELECT spaltenname1
FROM tabellenname1
UNION
SELECT spaltenname1
FROM tabellenname2;
```

Mit UNION werden die Resultate von zwei oder mehr SELECT Statements verbunden werden.

### 1. Beispiel 91: Anzahl der Unterrichtsstunden aller Lehrer (UNION)

```
SELECT l_name AS Lehrername, 0 AS Anzahl_Unterrichtsstunden
FROM lehrer
WHERE l_id NOT IN (SELECT st_l_lehrer FROM stunden)
UNION
SELECT MAX(l_name) AS Lehrername, COUNT(*)
FROM lehrer, stunden
WHERE l_id = st_l_lehrer
GROUP BY l_id
ORDER BY 2 DESC;
```

Ausgabe:

Lehrername	Anzahl_Unterrichtsstunden
Bilek	11
Berger	9
Preißl	9
Hanke	7
Beringer	6
Lorenz	5
Makandreou	2
Unger	2
Moitzi	1
Aichholzer	0
Hickel	0
Lenau	0
Pirkner	0
Radlbauer	0
Renkin	0
Siegel	0
Stappler	0

## 2. Beispiel 92: Vornamen aller Schüler und Lehrer (ohne Duplikate / UNION)

```

SELECT    l_vorname AS Vornamen
FROM      lehrer
UNION
SELECT    s_vorname
FROM      schueler
ORDER BY  1;

```

Ausgabe:

Vornamen
Alfred
Andrea
Anna
Balduin
Bebbles
Berta
Burghard
Eberhard
Engelbert
Franz
Fridolin
Gustav
Hans
Heinz
Helga
Johann
Karoline
Kasimir
Klaus
Konrad
Margit
Martha
Max
Nikolaus
Pampam
Paul
Peter
Richard
Robert
Rudi
Sebastian
Sonja
Susanne
Theodor
Walter
Wolfgang

### Beispiel 93: Vornamen aller Schüler und Lehrer (mit Duplikaten / UNION ALL)

```

SELECT    l_vorname AS Vornamen
FROM      lehrer
UNION ALL
SELECT    s_vorname
FROM      schueler
ORDER BY  1;

```

Ausgabe:

Vornamen
Alfred
Andrea
Anna
Balduin
Bebbles
Berta
Burghard
Eberhard
Engelbert
Franz
Franz
Fridolin
Gustav
Gustav
Gustav
Hans
Heinz
Helga
Johann
Karoline
Kasimir
Klaus
Konrad
Marqit
Martha
Max
Nikolaus
Pampam
Paul
Peter
Richard
Robert
Rudi
Sebastian
Sonia
Susanne
Theodor
Walter
Wolfgang

```

SELECT spaltenname1
FROM tabellenname1
  UNION ALL
SELECT spaltenname1
FROM tabellenname2;

```

Im Gegensatz zu UNION, das die Ergebnisse der einzelnen SELECT Statements abzüglich allfällig vorkommender, doppelter Datensätze miteinander vereinigt, belässt UNION ALL Duplikate in der Ergebnistabelle.

### 3. Beispiel 94: Alle Klassen vereinigt mit allen Räumen

```

SELECT    k_id AS Nonsens
FROM      klassen
UNION
SELECT    r_id
FROM      raeume;

```

Ausgabe:

Nonsens
03TA
03TB
0M5A
0M5Q
1VL0
B1
B2
LA1
LA2
LA3

Auch wenn diese Abfrage wenig Sinn macht: die einzige Voraussetzung für ein UNION ist, dass die zu vereinigenden Elemente den gleichen Spaltenaufbau und damit auch den gleichen Datentyp haben.

## Y. Statistiken in SQL

### 1. Beispiel 95: Durchschnitt und Summe der Lehrergehältern

```
SELECT    COUNT(*) AS Zeilenanzahl,
          AVG(l_gehalt) AS Durchschnittsgehalt,
          SUM(l_gehalt) AS Gehaltssumme,
          SUM(l_gehalt) / COUNT (*) AS Summe_durch_Anzahl
FROM      lehrer;
```

Ausgabe:

Zeilenanzahl	Durchschnittsgehalt	Gehaltssumme	Summe_durch_Anzahl
17	195	2730	160,588235294118

### 2. Beispiel 96: Anzahl unterschiedlicher Lehrergehälter

```
SELECT    COUNT(*) AS Anzahl_unterschiedliche_Gehälter
FROM      (SELECT DISTINCT l_gehalt FROM lehrer);
```

Ausgabe:

Anzahl_unterschiedliche_Gehälter
12

### 3. Beispiel 97: Ungewichteter Durchschnitt der Lehrergehälter

Die Besonderheit des gewichteten Durchschnitts ist, dass Duplikate nicht als Grundlage der Berechnung herangezogen werden, i.e. doppelte oder mehrfach gegebene Werte kommen trotz allem nur einmal zum Zug.

```
SELECT    AVG(l_gehalt) AS Ungewichteter_Durchschnitt
FROM      (SELECT DISTINCT l_gehalt FROM lehrer);
```

Ausgabe:

Ungewichteter_Durchschnitt
184,545454545455

#### 4. Beispiel 98: Modus der Lehrergehälter

Der innerhalb einer Menge am häufigsten vorkommende Wert wird in der Statistik als Modus bezeichnet. (Der Modus von 1, 1, 1, 2, 2, 3, 4 ist 1).

```
SELECT    l_gehalt AS Modus, COUNT(*) AS Anzahl
FROM      lehrer
GROUP BY  l_gehalt
HAVING    COUNT(*) >= ALL (SELECT COUNT(*) FROM lehrer GROUP BY l_gehalt);
```

Ausgabe:

Modus	Anzahl
	3

In unserem Fall ist bei den Gehältern also am häufigsten der Nullwert eingetragen, konkret betrifft dies die folgenden drei Lehrer:

l_name	l_vorname	l_gehalt
Hickel	Wolfgang	
Siegel	Heinz	
Stappler	Sonja	

#### 5. Beispiel 99: Streubreite und arithmetisches Mittel der Lehrergehälter

```
SELECT    MAX(l_gehalt) - MIN(l_gehalt) AS Streubreite,
          (MAX(l_gehalt) + MIN(l_gehalt) ) / 2 AS Arithmetisches_Mittel
FROM      lehrer;
```

Ausgabe:

Streubreite	Arithmetisches_Mittel
325	237,5

## Z. Extraktion von Stringbestandteilen

Wenn Sie sich zurückerinnern, haben Sie in diesem Skriptum schon Befehle kennengelernt, um aus einem Datum Werte zu extrahieren, nämlich year, month und day. Die Funktionen LEFT, RIGHT und MID werden Ihnen gute Dienste tun, wenn Sie SQL Abfragen erstellen möchten, die die Extraktion eines Teilstrings aus einem String erfordern.

### 1. Beispiel 100: Erster bis vierter Buchstabe alle Lehrervornamen

```
SELECT    DISTINCT LEFT(l_vorname, 4) AS Erste_bis_vierte_Stelle_der_Lehrervornamen
FROM      lehrer
ORDER BY  1;
```

Ausgabe:

Erste_bis_vierte_Stelle_der_Lehrervornamen
Alfr
Andr
Fran
Gust
Hans
Hein
Joha
Klau
Marg
Max
Niko
Rudi
Sonj
Walt
► Wolf

## 2. Beispiel 101: Letzter und vorletzter Buchstabe alle Lehrervornamen

```
SELECT    DISTINCT RIGHT(l_vorname, 2) AS
           Letzte_und_vorletzte_Stelle_der_Lehrervornamen
FROM      lehrer
ORDER BY  1;
```

Ausgabe:

	Letzte_und_vorletzte_Stelle_der_Lehrervornamen
	av
	ax
	di
	ea
	ed
	er
	it
	ja
	ng
	nn
	ns
	nz
►	us



## 3. Beispiel 102: Dritter bis vierter Buchstabe alle Lehrervornamen

```
SELECT    DISTINCT MID(l_vorname, 3, 2) AS Dritte_bis_vierte_Stelle_der_Lehrervornamen
FROM      lehrer
ORDER BY  1;
```

Ausgabe:

Dritte_bis_vierte_Stelle_der_Lehrervornamen
an
au
di
dr
fr
ha
in
ko
lf
lt
nj
ns
rg
st
x

## II. ANHANG A: Inhalt aller Tabellen der Schuldatenbank

### 1. Tabelle gegenstaende

G_ID	G_Bez
BO	Betriebliche Organisation
BSYS	Betriebssysteme
DBSYS	Datenbanksysteme
DE	Deutsch
EN	Englisch
MA	Mathematik
PR	Programmieren
PRRV	Prozessrechnerverbung
RW	Rechnungswesen
SEP	System-Einsatzplanung
TDO	Technische Datenorganisation

### 2. Tabelle klassen

K_ID	K_Bez	K_S_Klaspr	K_S_Klasprstv	K_L_Klavst
03TA	Kolleg T 03TA	1	122	BA
03TB	Kolleg T 03TB	1266	111	BI
0M5A	Kolleg A 0M5A	3333	3338	AM
0M5Q	Kolleg A 0M5Q	2222	2223	PS
1VL0	VorbereitungLG			I

### 3. Tabelle lehrer

L_ID	L_Name	L_Vorname	L_Gebdat	L_Gehalt	L_L_Chef
AI	Aichholzer		07.09.1946	150,00	LN
AM	Moitzi	Andrea		75,00	BF
BA	Beringer	Alfred	15.07.1961	220,00	HA
BF	Berger	Franz	02.03.1945	400,00	HI
BI	Bilek	Hans	03.03.1932	280,00	HA
HA	Hanke	Gustav	12.12.1950	300,00	BF
HI	Hickel	Wolfgang			
LN	Lenau	Nikolaus	22.02.1938	180,00	HA
LO	Lorenz		12.12.1958	300,00	HA
MM	Makandreou	Margit	01.04.1947	120,00	LN
PI	Pirkner	Walter	22.06.1955	220,00	BF
PS	Preißl	Johann	05.07.1956	100,00	HA
RR	Radlbauer	Rudi	11.11.1949	115,00	PI
RX	Renkin	Max	19.08.1961	180,00	PI
SG	Siegel	Heinz			HA
SS	Stappler	Sonja			HI
UK	Unger	Klaus		90,00	BF I

## 4. Tabelle pruefungen

P_Datum	P_S_Kandidat	P_L_Pruefer	P_G_Fach	P_Art	P_Note
24.12.1997	55	PS	TDO	M	5
01.01.1998	1	PS	TDO	M	1
04.01.1998	1266	PS	BO	M	3
12.01.1998	3	BI	TDO	M	1
12.01.1998	22	BA	TDO	M	2
06.02.1998	111	BA	MA	M	4
15.04.1998	19	BA	TDO	M	5
07.06.1998	111	BA	TDO	M	1
15.03.1999	1	BA	BO	S	2
15.03.1999	22	BA	BO	K	4
15.03.1999	111	BA	TDO	F	4
01.04.1999	1	HA	MA	M	4
01.04.1999	1	PS	MA	K	2
01.04.1999	4	HA	MA		3
01.10.2003	2222	PS	DBSYS		1
01.10.2003	2223	PS	DBSYS		2
01.10.2003	2224	AM	DBSYS	M	1
01.10.2003	3333	MM	RW	M	1
10.10.2003	2222	BA	BSYS	M	2
10.10.2003	2223	BA	BSYS	M	5
10.10.2003	2224	BA	BSYS	M	4
10.10.2003	2225	BA	BSYS	M	5
10.10.2003	3333	BA	BSYS	M	1
05.11.2003	3334	UK	PR		2
06.11.2003	3335	UK	PR	S	1
07.11.2003	3336	UK	PR	S	4
12.11.2003	3336	UK	PR	S	2
13.11.2003	2222	BF	PR	M	1
14.11.2003	2224	BF	TDO		2
15.11.2003	2225	BF	EN	M	4
15.11.2003	2225	BF	MA	M	2

## 5. Tabelle raeume

R_ID	R_Plaetze
B1	22
B2	24
LA1	10
LA2	18
LA3	2

## 6. Tabelle schueler

S_SCHNR	S_Name	S_Vorname	S_Gebdat	S_Adresse	S_K_Klasse
1	Adler	Richard	12.03.1973	Josefstadt	03TA
3	Bartgeier	Burghard	01.01.1980	Wieden	03TB
4	Rotkehlchen	Robert	05.05.1977	Fünfhaus	03TA
16	Geier	Gustav	04.05.1978	Simmering	03TA
19	Sitzenbleiber	Sebastian	08.01.1983	Hernals	03TA
22	Klassenbester	Konrad	03.09.1985	Margareten	03TA
55	Picasso	Paul	03.12.1980	Döbling	03TB
74	Hundertwasser	Helga	02.02.1980	Landstrasse	03TA
77	Pollak	Peter	08.04.1982	Favoriten	03TB
84	Feuerstein	Bebbles	02.07.1993	Steintal	03TB
88	Geröllheimer	Pampam	15.03.1984	Ottakring	03TB
111	Ente	Eberhard	19.04.1983	Hernals	03TB
122	Gans	Gustav	01.01.1962	Brigittenau	03TA
1266	Schwan	Susanne	23.04.1963	Ottakring	03TB
2222	Apfelbaum	Anna	12.12.1965	Tulln	0M5Q
2223	Birnenbaum	Berta	12.12.1965	Mödling	0M5Q
2224	Marillenbaum	Martha	11.11.1964	Baden	0M5Q
2225	Kirschebaum	Karoline	10.10.1963	Eisenstadt	0M5Q
3333	Tanne	Theodor	09.09.1962	Brunn/Gebirge	0M5A
3334	Fichte	Fridolin	09.09.1962	Vösendorf	0M5A
3335	Föhre	Franz	28.09.1972	Wöllersdorf	0M5A
3336	Kiefer	Kasimir	17.04.1969	Mistelbach	0M5A
3337	Buche	Balduin	31.01.1980	Gänserndorf	0M5A
3338	Eibe	Engelbert	20.02.1982	Stockerau	0M5A

## 7. Tabelle stunden

ST_K_Klasse	ST_L_Lehrer	ST_G_Fach	ST_Stunde	ST_R_Raum
03TA	BI	BO	DI1	B1
03TA	BI	BO	DI2	B1
03TA	HA	PR	DI3	LA3
03TA	HA	PR	DI4	LA3
03TA	HA	PR	DI5	LA3
03TA	BF	MA	DO1	B1
03TA	BF	MA	DO2	B1
03TA	BI	BO	DO3	B1
03TA	BA	PR	DO4	LA1
03TA	BA	PR	DO5	LA1
03TA	PS	TDO	MI1	LA2
03TA	PS	TDO	MI2	LA2
03TA	PS	TDO	MI3	B1
03TA	BI	RW	MI4	B1
03TA	BI	RW	MI5	B1
03TA	LO	TDO	MO1	B1
03TA	LO	TDO	MO2	B1
03TA	LO	TDO	MO3	B1
03TA	BA	SEP	MO4	B1
03TA	BA	SEP	MO5	B1
03TA	PS	PR	MO7	LA1
03TA	PS	PR	MO8	LA1
03TB	PS	PR	DI1	LA1
03TB	PS	PR	DI2	LA1
03TB	HA	PR	DI3	LA2
03TB	HA	PR	DI4	LA2
03TB	HA	PR	DI5	LA2
03TB	PS	TDO	DO1	B2
03TB	PS	TDO	DO2	B2
03TB	BF	MA	DO3	B2
03TB	BF	MA	DO4	B2
03TB	BI	RW	MI1	B2
03TB	BI	RW	MI2	B2
03TB	BI	RW	MI3	B2
03TB	LO	TDO	MI4	B2
03TB	LO	TDO	MI5	B2
03TB	BI	BO	MO1	B2
03TB	BI	BO	MO2	B2
03TB	BI	BO	MO3	B2
03TB	BA	SEP	MO4	B2
03TB	BA	SEP	MO5	B2
0M5A	BF	TDO	DI2	B2
0M5A	BF	PR	DI6	B2
0M5A	BF	PR	DI9	LA1
0M5A	BF	PR	MI4	LA2
0M5Q	HA	BSYS	MO1	B1
0M5Q	AM	DBSYS	MO12	LA1
0M5Q	UK	PR	MI2	LA2
0M5Q	UK	PR	DO3	LA2
0M5Q	MM	RW	FR2	B2
0M5Q	MM	EN	FR3	B1
0M5Q	BF	DE	MO11	B1

## 8. Tabelle vorgesetzte

V_L_Vorg	V_Art	V_L_Unt
BA	Fgrp Prog	PS
BA	Fgrp Prog	AM
BA	Fgrp Prog	UK
BF	AV	AM
BF	AV	UK
BI	Fgrp RW	LO
HA	AV	BA
HA	AV	BI
HA	AV	LO
HA	AV	PS
HI	Direktor	BA
HI	Direktor	BI
HI	Direktor	HA
HI	Direktor	LO
HI	Direktor	PS
HI	Direktor	AM
HI	Direktor	UK
HI	Direktor	MM
HI	Direktor	BF
HI	Direktor	SS
SS	AV	MM
SS	AV	BA

### III. ANHANG B: Übungsbeispiele

ID	Wir sind interessiert an ...	L, N, H
1	.. dem Gesamtinhalt der Tabelle 'raeume'	L/N/H
2	.. den Nachnamen aller Schüler	L/N/H
3	.. den Nachnamen der Schüler, alphabetisch aufsteigend (A-Z) sortiert	L/N/H
4	.. den Vornamen und den Nachnamen aller Lehrer, primär nach dem Nachnamen sortiert, sekundär nach dem Vornamen	L/N/H
5	.. den Langbezeichnungen der Unterrichtsgegenstände	L/N/H
6	.. den Vornamen aller Lehrer unter Ausschluss von Duplikaten	L/N/H
7	.. Vornamen und Nachnamen aller Lehrenden, die keinen Vorgesetzten haben	L/N/H
8	.. Klassen, deren Identifikation mit einer 0 beginnt	L/N/H
9	.. Klassen, die an einer beliebigen Stellen den Buchstaben A haben	L/N/H
10	.. Schüler, deren Vorname mit T beginnt und mit M endet	L/N/H
11	.. den Namen aller Klassenvorstände unter Angabe der dazugehörigen Klasse	L/N/H
12	.. den Namen und Adressen aller Klassensprecher und Klassensprecherstellvertreter	L/N/H
13	.. dem Geburtsdatum des jüngsten Schülers	L/N/H
14	.. dem Geburtsdatum des ältesten Lehrers	L/N/H
15	.. dem Geburtsdatum des jüngsten Schülers mit der Überschrift 'Unser Jüngster'	L/N/H
16	.. den Vornamen und Nachnamen der Schüler, die in Ottakring wohnen	L/N/H
17	.. den Lehrer, die Programmieren unterrichten	L/N/H
18	.. den Lehrern, die weder Programmieren, TDO noch Rechnungswesen unterrichten	L/N/H
19	.. den Schülern, die in Klassen gehen, die mit einem 0 beginnen	L/N/H
20	.. den Stunden, die im Labor LA1 unterrichtet werden	L/N/H
21	.. den Stunden, die in Labors unterrichtet werden	L/N/H
22	.. der Anzahl der Stunden, die in Labors unterrichtet werden	L/N/H

ID	Wir sind interessiert an ...	L, N, H
23	.. den Schülern, die im Lehrsaal B1 unterrichtet werden	L/N/H
24	.. den Schülern, die im Lehrsaal B1 unterrichtet werden, sortiert nach deren Klasse	L/N/H
25	.. den Jahren, in denen Prüfungen abgehalten wurden	L/N/H
26	.. den Namen der Lehrer, die Abteilungsvorstände sind	L/N/H
27	.. den Namen der Abteilungsvorstände, die weniger als 200 verdienen	L/N/H
28	.. den untergebenen Lehrern mit den gleichen Vornamen wie vorgesetzte Lehrer	L/N/H
29	.. der Anzahl der Schüler je Klasse	L/N/H
30	.. den Klassen, die mehr als 8 Schüler haben	L/N/H
31	.. den Klassen, die von weniger als 5 Schülern besucht werden	L/N/H
32	.. den Klassenvorstände, die bereits Schuler ihrer Klasse geprüft habe	L/N/H
33	.. den Klassenvorstände, die mehr als 2 Gegenstände unterrichten	L/N/H
34	.. dem Vorgesetzten des Klassenvorstandes der Klasse, die der Schüler Adler besucht	L/N/H
35	.. der Durchschnittsnote aller Prüfungen	L/N/H
36	.. der Durchschnittsnote aller Prüfungen je Gegenstand	L/N/H
37	.. dem Minimalgehalt und dem Durchschnittsgehalt der Lehrergehälter	L/N/H
38	.. Lehrer, die keinen Eintrag im Gehaltsfeld haben	L/N/H
39	.. Lehrer, die mehr verdienen als der Direktor der Schule	L/N/H
40	.. der Durchschnittsnote je Gegenstand unter Berücksichtigung auch jener Gegenstände, für die noch keine Prüfung abgehalten wurde	L/N/H
41	.. den Namen des Schülers, der die meisten Prüfungen absolviert hat	L/N/H
42	.. den Gegenständen, die ausschliesslich von Lehrern unterrichtet werden, die nicht der Kategorie 'untergebene Lehrer' zuordnen sind	L/N/H
43	.. Schülern, die versehentlich als Klassensprecher oder Klassensprecherstellvertreter für Klassen eingetragen sind, die sie selbst nicht besuchen	L/N/H
44	.. Klassenvorständen, die Datenbanksysteme unterrichten	L/N/H
45	.. Schülern, die noch keine Prüfung absolviert haben	L/N/H



ID	Wir sind interessiert an ...	L, N, H
46	.. den Vornamen aller Lehrer und aller Schüler mit der Einschränkung, dass Duplikate zulässig sind	L/N/H
47	.. den Vornamen aller Lehrer und aller Schüler mit der Einschränkung, dass Duplikate nicht zulässig sind	L/N/H
48	.. dem Namen des zweitjüngsten Schülers	L/N/H
49	.. der Prüfungsanzahl und dem Notendurchschnitt je Lehrer	L/N/H
50	.. der Liste der noch nie geprüften Gegenstände	L/N/H
51	.. den Lehrern, die in mehr als einer Klasse Klassenvorstand sind	L/N/H
52	.. Lehrer, die älter als das Lehrer-Durchschnittsalter sind	L/N/H
53	.. Schüler, die am 1. April Geburtstag haben	L/N/H
54	.. Schüler und Lehrer, die am 1. April Geburtstag haben	L/N/H
55	.. Durchschnittsnote und Anzahl aller Prüfungen je Schüler mit einer oder mehr Prüfungen	L/N/H
56	.. Durchschnittsnote und Anzahl aller Prüfungen aller Schüler	L/N/H
57	.. der Häufigkeit der einzelnen Schüler-Vornamen	L/N/H
58	.. Gegenstände, die nicht unterrichtet werden	L/N/H
59	.. Schüler, die zwischen 1.1.1980 und 31.12.1985 geboren wurden	L/N/H
60	.. Räume, die doppelbelegt sind (mehr als ein Gegenstand/eine Klasse je Unterrichtsstunde)	L/N/H
61	.. Schüler, die älter als an der Schule angestellte Lehrer sind	L/N/H
62	.. Lehrer, die Prüfungen nur mit 1 oder 2 beurteilen	L/N/H
63	.. Namen des Lehrers, der die meisten 5er ausgeteilt hat	L/N/H
64	.. Lehrer, die aktuell in keiner Klasse unterrichten	L/N/H
65	.. Lehrer, die nur am Vormittag unterrichten	L/N/H
66	.. Schüler gleichen Nachnamens, allerdings in unterschiedlichen Klassen	L/N/H
67	.. Schüler, die im gleichen Bezirk wie ihr Klassenvorstand wohnen	L/N/H
68	.. Anzahl der Untergebenen je Vorgesetzten-Typ	L/N/H
69	.. Anzahl der abgelegten Prüfungen je Note	L/N/H

ID	Wir sind interessiert an ...	L, N, H
70	.. Klassenvorstände, die weniger als 3 Prüfungen abgehalten haben	L/N/H
71	.. Lehrer, die ausschliesslich ‚Untergebene‘ sind, nicht aber ‚Vorgesetzte‘	L/N/H
72	.. die Anzahl der Schüler je Geburtsjahr	L/N/H
73	.. Fächer, die lt. Stundenplan nur von einem Lehrer unterrichtet werden	L/N/H
74	.. Klasse mit der höchsten Wochenstundenanzahl	L/N/H
75	.. Räume, die für mindestens drei unterschiedliche Fächer verwendet werden	L/N/H
	.. etc, etc, ...	

Sie werden für die an dieser Stelle aufgezählten Beispiele vergeblich die Lösungen im Skriptum suchen. Aus einem simplen Grund: das Lesen, Auseinandersetzen und Verstehen der Fragestellungen und Antworten aus den vorangegangenen Kapitel ist der erste Schritt, das eigenständige Üben sowie das Erarbeiten von Lösungen in der Gruppe jedoch der entscheidende zweite.

Die Spalte mit den Werten L, N, H dient der Selbstkontrolle und kann nur von Ihnen selbst beantwortet werden. Klassifizieren Sie jedes SQL Statement, indem Sie festhalten, ob Sie die Lösung für die Fragestellung  
 locker (L),  
 mit ein bisschen Nachdenken und Nachlesen (N) oder  
 nur mit Hilfe anderer Kolleg/innen (H)  
 fanden.

In diesem Sinne: viel Spaß und Erfolg bei der Realisierung der SQL Statements.

## IV. ANHANG C: Ansätze für die Erweiterung des Datenmodells

Das der verwendeten Schul-Datenbank zugrundeliegende Datenmodell wurde bewusst einfach gehalten (siehe Kapitel II), um die Grundkonzepte von SQL vorstellen zu können, ohne Sie der Gefahr auszusetzen, in einer Masse von Tabellen mit Unmengen von Spalten die Übersicht zu verlieren.

Selbstverständlich steht es jedem frei, dieses Datenmodell als Grundlage für ein erweitertes Datenmodell heranzuziehen, um komplexere Sachverhalte abbilden zu können. Hier einige Denkanstöße.

Wie müsste das Datenmodell modifiziert werden, um
.. zwischen pragmatisierten und nicht pragmatisierten Lehrern zu unterscheiden
.. das Eintrittsdatum für jeden Lehrenden zu erfassen
.. für Schüler das Religionsbekenntnis und die Staatsbürgerschaft zu erfassen
.. die Vorbildung (letzte besuchte Schule / Schulzweig) der Schüler/innen abzubilden
.. zwischen männlichen und weiblichen Lehrern und Schülern zu differenzieren
.. Verwandtschaftsverhältnisse zwischen Lehrern und Schülern darzustellen
.. Ehen zwischen Lehrern und Lehrerinnen festzuhalten
.. Unterrichtsstunden von mehr als einem Lehrer abhalten zu lassen
.. in mehreren Jahren denselben Klassennamen zuzulassen
.. Zeugnisdaten zu vermerken
.. Diplomarbeitenprojekte von Abschlussjahrgängen zu verwalten
.. die Kategorie eines Lehrsaales nicht aus seinem Namensbeginn ablesen zu müssen
.. Leereinträge bei den Lehrergehältern nicht zuzulassen
.. Gegenstandsfehleintragungen bei den Stundenplänen zu vermeiden
.. eine Unterrichtsstunde laut Stundenplan in mehr als einem Lehrsaal stattfinden zu lassen
.. Klassen ohne Klassenvorstand zu unterbinden
.. Abendklassen speziell zu kennzeichnen
.. Lehrer mit gleichen Initialen zuzulassen
.. Ausstattungsgegenstände je Lehrsaal/Labor zu erfassen
.. maximal zwei Erziehungsberechtigte je Schüler zu hinterlegen

.. Lehrer mit Sonderfunktionen wie Erste-Hilfe-Befähigter, Schulbibliotheks-Verantwortlicher, Brandschutzbeauftragter kennzuzeichnen
--

.. die Menge der zulässigen Vorgesetztenarten einzuschränken
--

.. bei Räumen neben der Maximalbelegung auch eine Minimalbelegung anzugeben
---

.. etc, etc ...
-----------------

. und unter der Annahme, dass die Datenbank einige / alle der oben skizzierten Erweiterungen erfuhr, wären dann SQL Statements für folgende Fragestellungen interessant:

Gesucht ist/sind
.. die Namen der nicht-pragmatisierten Lehrer
.. die Namen der pragmatisierten Lehrer, die schon mehr als 15 Jahre an der Schule sind
.. die Anzahl der Schülerinnen
.. die Anzahl der Schülerinnen je Klasse
.. die Namen der Lehrer, die einerseits AV sind und andererseits gleichzeitig eine Sonderfunktion wahrnehmen
.. etc, etc ...

## v. ANHANG D: Feedback

Ihr Feedback zu diesem Skriptum ist uns sehr willkommen. Bitte schreiben Sie uns Ihre Anregungen (oder noch besser, Ihre konkreten Vorschläge) an

[preissl@spengergasse.at](mailto:preissl@spengergasse.at)

[voit@spengergasse.at](mailto:voit@spengergasse.at)

Vielen Dank schon vorab für Ihre Unterstützung!

## VI. ANHANG E: Raum für Ihre Notizen











