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

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

I. ZIELE 6

Zielgruppe 6

Zielsetzung 6

Nicht-Ziele 6

Mitgeltende Dokumente 7

II. UMGEBUNG 8

III. SQL BEFEHLE IM ÜBERBLICK 9

Das Konzept von SQL 9

SQL ist mehr als eine Abfragesprache 10

DDL Befehle 10

Anlegen einer Tabelle (CREATE TABLE) 11

Anlegen eines Index (CREATE INDEX) 16

Anlegen eines Synonyms (CREATE SYNONYM) 16

Anlegen einer View (CREATE VIEW) 17

Verändern von Tabellenstrukturen (ALTER TABLE) 18

Löschen einer Tabelle (DROP TABLE) 19

Löschen eines Index (DROP INDEX) 19

Löschen eines Synonyms (DROP SYNONYM) 20

Löschen einer View (DROP VIEW) 20

DML Befehle 21

Lesen von Datensätzen aus einer/mehreren Tabelle/n (SELECT) 21

Einfügen von Datensätzen in eine Tabelle (INSERT) 22

Das Bearbeiten von Datensätzen in einer Tabelle (UPDATE) 22

Das Löschen von Datensätzen aus einer Tabelle (DELETE) 23

Befehle für die Verwaltung von Zugriffsrechten 24

Die Vergabe von Zugriffsrechten 24

Der Entzug von Zugriffsrechten 24

Befehle zur Transaktionssteuerung 25

Commit einer Transaktion (COMMIT) 25

Rollback einer Transaktion (ROLLBACK) 25

IV. DAS SELECT 26

Auswahl von Spalten 26

Beispiel 1: Ausgabe aller Räume 26

Beispiel 2: Ausgabe aller Langbezeichnungen der Gegenstände 27

Beispiel 3: Ausgabe aller Prüflinge mit Prüfungsfach und Prüfungsdatum 28

Arbeiten mit Bedingungen 29

Beispiel 4: Schüler mit einer Schülernummer kleiner 20 29

Beispiel 5: Schüler mit einer Schülernummer gleich 2222 29

Beispiel 6: Lehrer mit einem Gehalt < 500 und > 200 (mit AND) 30

Beispiel 7: Lehrer mit einem Gehalt < 500 und > 200 (mit BETWEEN) 31

Beispiel 8: Lehrergehalt < 500, > 200 und Geburtsdatum vor 1.1.1955 31

Beispiel 9: Lehrer mit Gehalt größer 250 oder Geburtsdatum vor 1.1.1955 32

Beispiel 10: Schüler mit Wohnort Hernals 32

Beispiel 11: Schüler mit Wohnort Hernals oder Mödling 33

Beispiel 12: Schüler der 03TA mit Wohnort Hernals 33

Beispiel 13: Direktoren 33

Beispiel 14: Untergebene Lehrer von Direktoren 34

Beispiel 15: Untergebene Lehrer von Nicht-Direktoren 34

Beispiel 16: Schüler mit einer 1 auf eine beliebige Prüfung 35

Beispiel 17: Schüler mit einer 1 in Englisch 35

Beispiel 18: Lehrergehalt 100 oder 220 und Geburtsdatum vor 1.1.1956 (mit / ohne Klammern) 36

Beispiel 19: Lehrergehalt 100 oder 220 und Geburtsdatum vor 1.1.1956 (mit / ohne Klammern) 36

Beispiel 20: Lehrergehalt 82, 220, 300 und Geburtsdatum vor 1.1.1970 37

Beispiel 21: Lehrer mit Geburtsdatum ungleich 12.12.1950 37

Beispiel 22: Lehrergehalt 82, 220, 300 und Geburtsdatum vor 1.1.1970 38

Qualifizierung 39

Beispiel 23: Schüler und deren Geburtstag (ohne Qualifizierung) 40

Beispiel 24: Schüler und deren Geburtstag (mit Qualifizierung) 41

Sortierung 42

Beispiel 25: Alphabetisch absteigend sortierte Schülerliste 43

Beispiel 26: Alphabetisch absteigend sortierte Schülerliste (etwas kürzer) 44

Beispiel 27: Zweifach sortierte Schülerliste 45

Alias für Spaltennamen 46

Beispiel 28: Gegenstandsausgabe mit einprägsamer Spaltenbezeichnung 46

Wildcards 47

Beispiel 29: Lehrer, deren Vorname Gustav lautet 47

Beispiel 30: Lehrer, deren Vorname mit M beginnt 47

Beispiel 31: Lehrernachnamen mit einem B an 1. und einem R an 3. Stelle 48

Beispiel 32: Klassen mit eine 0 an beliebiger Stelle 48

Duplikate 49

Beispiel 33: Stunden, die von PS unterrichtet werden (mit Duplikaten) 49

Beispiel 34: Stunden, die von PS unterrichtet werden (ohne Duplikate) 49

Beispiel 35: Gehälter der Lehrer (ohne Duplikate) 50

Nullwerte 51

Beispiel 36: Lehrer ohne Gehalt 52

Beispiel 37: Lehrer mit bekanntem Gehalt, aufsteigend sortiert 52

Konstante 54

Beispiel 38: Geniale Schüler 54

Aggregatfunktionen 55

Beispiel 39: Aggregierter Wert, Konstante und Ausdruck für Lehrer 55

Beispiel 40: Aggregationswert, Konstante, Ausdruck für Lehrer mit Chef HI 56

Beispiel 41: Rechnen mit dem Lehrergehalt 56

Beispiel 42: Geburtsdatum des jüngsten Schülers 56

Beispiel 43: Geburtsdatum des ältesten Schülers 57

Beispiel 44: Ausgabe des heutigen Datums 57

Beispiel 45: Rechnen mit den Geburtsdaten der Lehrer 59

Verbund (Join / Inner Join) 60

Beispiel 46: Schüler der Klasse 03TA 60

Beispiel 47: Schüler der Klasse 03TA mit Klassenbezeichnung 61

Beispiel 48: Stundenplan für Montag in der 03TA mit Gegenstandsnamen 62

Beispiel 49: Prüfungen, die der Klassenvorstand abhielt 63

Beispiel 50: Schüleranzahl der 03TB 64

Beispiel 51: Schüleranzahl der 03TB mit Klassenlangname 64

Beispiel 52: Sitzplätze der Lehrsäle und Labors der 03TA (mit Duplikaten) 65

Beispiel 53: Sitzplätze der Lehrsäle und Labors der 03TA (ohne Duplikate) 66

Alias für Tabellennamen 67

Beispiel 54: Lehrer mit Gehalt so hoch wie vom direkten Vorgesetzten 68

Beispiel 55: Lehrer mit Gehalt kleiner gleich dem des direkten Vorgesetzten 69

Beispiel 56: Liste aller Vorgesetztenbeziehungen 70

Datumswerte 71

Beispiel 57: Ausgabe der Schüler, die im Dezember Geburtstag haben 71

Beispiel 58: Ausgabe der Schüler, die am 3.12.1980 Geburtstag haben 71

Beispiel 59: Schüler, die jünger als Lehrer HA sind 72

Gruppierung 73

Beispiel 60: Schüleranzahl 73

Beispiel 61: Schüleranzahl je Klasse 73

Gruppeneinschränkung 74

Beispiel 62: Klassen mit mehr als 6 Schülern 74

Beispiel 63: Schüleranzahl je Jahrgang mit mehr als einem Schüler 74

Beispiel 64: Jahrgänge ab 1975 mit mehr als einem Schüler 75

Beispiel 65: Jahrgänge ab 1975 75

Beispiel 66: Vornamenhäufigkeit bei Lehrern (mit Nullwert) 76

Beispiel 67: Vornamenhäufigkeit bei Lehrern (ohne Nullwert) 77

Beispiel 68: Lehrsäle mit mehr als 5 Stunden Unterricht 77

Beispiel 69: Vorgesetzte Lehrer mit Durchschnittsgehalt ihrer Untergebenen 78

Beispiel 70: Vorgesetzte mit Durchschnittsgehalt > 150 der Untergebenen 78

Beispiel 71: Unterrichtsstunden je Lehrer 79

Beispiel 72: Lehrer, die mehr als 10 Stunden unterrichten 80

Beispiel 73: Stunden mit mehr Schülern als Sitzplätzen 81

Subselect 82

Beispiel 74: Geburtsdatum und Name des jüngsten Schülers 82

Beispiel 75: Schüler, die bereits Prüfungen absolviert haben (Subselect) 83

Beispiel 76: Schüler mit absolvierten Prüfungen (Join) 84

Beispiel 77: Klassensprecher 85

Beispiel 78: Klassensprecher der Klasse mit BA als Klassenvorstand 85

Beispiel 79: Klassensprecher (mit ANY) 86

Beispiel 80: Lehrer mit dem höchsten bekannten Gehalt (mit ALL) 87

Beispiel 81: Lehrer mit dem niedrigsten bekannten Gehalt 87

Beispiel 82: Klassensprecher und Klassensprecherstellvertreter (EXISTS) 88

Beispiel 83: Ungeprüfte Gegenstände 89

Beispiel 84: Unterrichtende, aber nicht prüfende Lehrer 89

Beispiel 85: Falsche Fremdschlüssel im Stundenplan 90

Beispiel 86: Lehrer mit nicht existierenden Vorgesetzten 91

Beispiel 87: Lehrer mit gleichem Gehalt 91

Beispiel 88: Lehrer mit dem zweithöchsten Gehalt 92

Outer Join 93

Beispiel 89: Notendurchschnitt aller Gegenstände 93

Beispiel 90: Notendurchschnitt aller Schüler 94

Mengenoperationen 95

Beispiel 91: Anzahl der Unterrichtsstunden aller Lehrer (UNION) 95

Beispiel 92: Vornamen aller Schüler und Lehrer (ohne Duplikate / UNION) 96

Beispiel 94: Alle Klassen vereinigt mit allen Räumen 98

Statistiken in SQL 99

Beispiel 95: Durchschnitt und Summe der Lehrergehältern 99

Beispiel 96: Anzahl unterschiedlicher Lehrergehälter 99

Beispiel 97: Ungewichteter Durchschnitt der Lehrergehälter 99

Beispiel 98: Modus der Lehrergehälter 100

Beispiel 99: Streubreite und arithmetisches Mittel der Lehrergehälter 100

Extraktion von Stringbestandteilen 101

Beispiel 100: Erster bis vierter Buchstabe alle Lehrervornamen 101

Beispiel 101: Letzter und vorletzter Buchstabe alle Lehrervornamen 102

Beispiel 102: Dritter bis vierter Buchstabe alle Lehrervornamen 103

ANHANG A: Inhalt aller Tabellen der Schuldatenbank 104

Tabelle gegenstaende 104

Tabelle klassen 104

Tabelle lehrer 104

Tabelle pruefungen 105

Tabelle raeume 105

Tabelle schueler 106

Tabelle stunden 107

Tabelle vorgesetzte 108

ANHANG B: Übungsbeispiele 109

ANHANG C: Ansätze für die Erweiterung des Datenmodells 112

ANHANG D: Feedback 114

ANHANG E: Raum für Ihre Notizen 115

I. ZIELE

* 1. 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[[1]](#footnote-1)

Hilfreich, aber nicht zwingend erforderlich, ist ein Wissen rund um die Datenmodellierung (mit ER[[2]](#footnote-2)-Diagrammen) bzw. um das Datenbankdesign von relationalen Datenbanken.

* 1. 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
  1. Nicht-Ziele

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

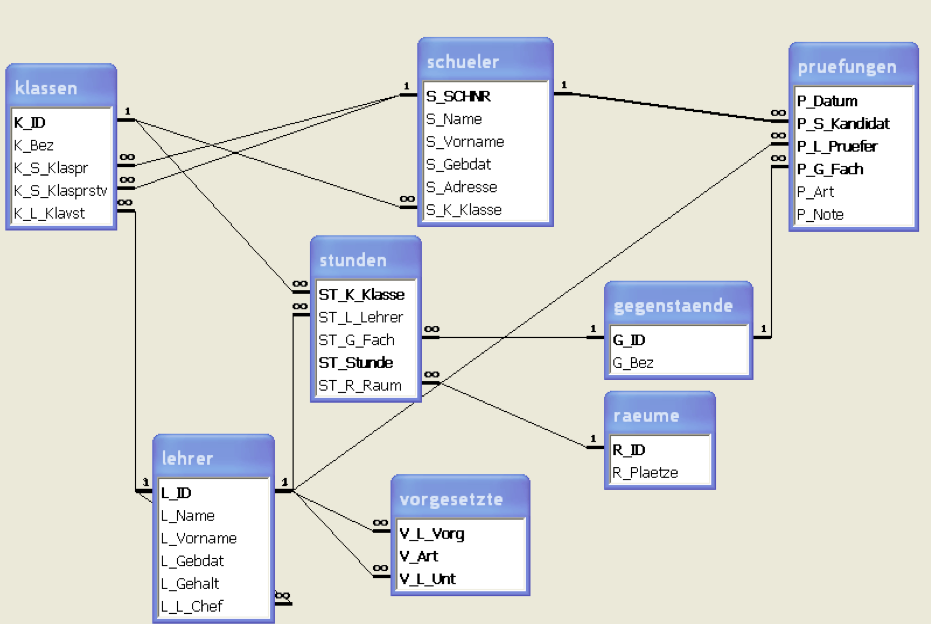
* Transaktionen
* Zugriffsrechte
* Stored Procedures
* Trigger
* Embedded SQL

abstrahiert. Der Schwerpunkt wird auf das SELECT gelegt werden.

* 1. 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:

1. III. SQL BEFEHLE IM ÜBERBLICK
   1. 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[[3]](#footnote-3) und ISO[[4]](#footnote-4). 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.

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
  1. 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 / einen Synonyms / einer View
* ALTER zum Modifizieren
* DROP zum Entfernen
  + 1. Anlegen einer Tabelle (CREATE TABLE)

Eine Tabelle

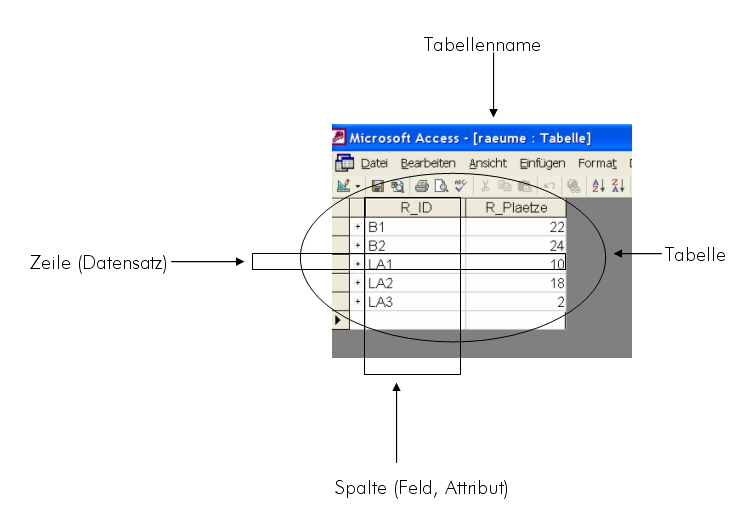
* ist die Struktur, die den Rahmen für die Speicherung von Daten darstellt
* zeichnet sich durch einen innerhalb der Datenbank eindeutigen Namen aus
* besteht aus 1 bis n Spalten (Attributen / Feldern) und aus 0 bis n Zeilen (Datensätzen / Records)
* hat einen natürlichen oder einen künstlichen, einen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

Die ‘einfache’ Syntax lautet:

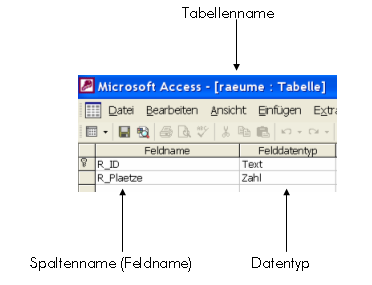
CREATE TABLE tabellenname (spaltenname datentyp [NOT NULL])

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) Feldinhalte 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:[[5]](#footnote-5)

| 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, andSECOND. 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:[[6]](#footnote-6)

| Datentyp | Spezifika |
| --- | --- |
| numeric | Numerische Datentypen mit fester Genauigkeit und fester Anzahl von Dezimalstellen. |
| date | Definiert ein Datum. |
| nvarchar | Unicode-Zeichendaten variabler Länge. nkann 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. |

Microsoft Access:[[7]](#footnote-7)

| 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. |
| Zahl | 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".  Feldgrößen:   * Byte : Verwenden Sie diese Option für ganze Zahlen von 0 bis 255. Die Speicherplatzanforderung beträgt 1 Byte. * Integer : Verwenden Sie diese Option für ganze Zahlen von -32.768 bis 32.767. Die Speicherplatzanforderung beträgt 2 Bytes. * Long Integer : Verwenden Sie diese Option für ganze Zahlen von -2.147.483.648 bis 2.147.483.647. Die Speicherplatzanforderung beträgt 4 Bytes. * Single   Verwenden Sie diese Option für numerische Gleitkommawerte, die von -3,4 x 1038bis 3,4 x 1038 reichen und bis zu sieben Nachkommastellen umfassen. Die Speicherplatzanforderung beträgt 4 Bytes. * Double   Verwenden Sie diese Option für numerische Gleitkommawerte, die von -1,797 x 1038 bis 1,797 x 1038 reichen und bis zu 15 Nachkommastellen umfassen. Die Speicherplatzanforderung beträgt 8 Bytes. * 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. * Dezimal   Verwenden Sie diese Option für numerische Werte von -9,999... x 1027 bis 9,999... x 1027. Die Speicherplatzanforderung beträgt 12 Bytes. |
| 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:[[8]](#footnote-8)

| 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. |
| 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 (216 – 1) Byte. |

* + 1. Anlegen eines Index (CREATE INDEX)

Ein Index

* beschleunigt die Suche[[9]](#footnote-9)
* 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.

* + 1. 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.

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.

* + 1. 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 neuerdatentp)

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.

* + 1. 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.

* + 1. Löschen eines Index (DROP INDEX)

Die Syntax zum Löschen eines Index lautet:

DROP INDEX indexname

indexname ist der Name des zu löschenden Index

* + 1. Löschen eines Synonyms (DROP SYNONYM)

Die Syntax zum Löschen eines Synonym lautet:

DROP SYNONYM synonymname

synonymname ist der Name des zu löschenden Synonyms.

* + 1. Löschen einer View (DROP VIEW)

Die Syntax zum Löschen einer View lautet:

DROP VIEW viewname

viewname ist der Name der zu löschenden View.

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!

* + 1. 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.

* + 1. 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.

* + 1. 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.

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 spezifiert 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
  + 1. Der Entzug von Zugriffsrechten

Die ‘einfache’ Syntax zum Entzug von Zugriffsrechten lautet:

REVOKE recht ON tabellenname FROM benutzer

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

* 1. 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;

* + 1. 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.

* 1. Auswahl von Spalten

Der Merksatz für das Lesen von Daten lautet:

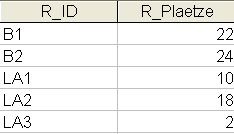
SELECT spaltennamen FROM tabellenname WHERE bedingung;

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:



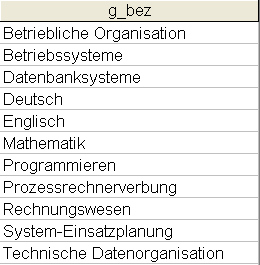
SELECT \* FROM tabellenname;

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

Beispiel 2: Ausgabe aller Langbezeichnungen der Gegenstände

SELECT g\_bezFROM gegenstaende;

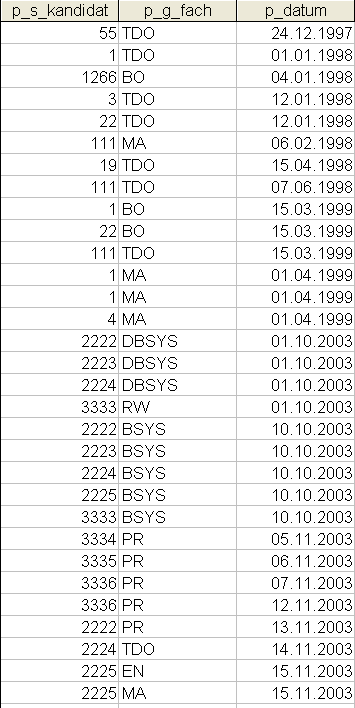
Ausgabe:



Beispiel 3: Ausgabe aller Prüflinge mit Prüfungsfach und Prüfungsdatum

SELECT p\_s\_kandidat, p\_g\_fach, p\_datumFROM pruefungen;

Ausgabe:



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\_klasseFROM schuelerWHERE s\_schnr < 20;

Ausgabe:



* + 1. Beispiel 5: Schüler mit einer Schülernummer gleich 2222

SELECT s\_schnr, s\_vorname, s\_name, s\_k\_klasseFROM schuelerWHERE s\_schnr = 2222;

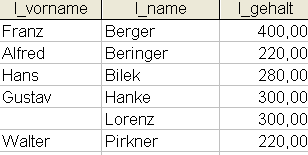
Ausgabe:



* + 1. Beispiel 6: Lehrer mit einem Gehalt < 500 und > 200 (mit AND)

SELECT l\_vorname, l\_name, l\_gehaltFROM lehrerWHERE l\_gehalt > 200 AND l\_gehalt < 500;

Ausgabe:



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\_gehaltFROM lehrerWHERE l\_gehalt > 200 AND l\_gehalt < 500;

liefert das gleiche Resultat wie

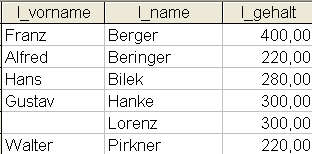
Select l\_vorname, l\_NAME, l\_geHAltFROM lehrer where l\_gehalt > 200 and L\_GEHALT < 500;

Welches Statement konnten Sie leichter lesen und erfassen?

* + 1. Beispiel 7: Lehrer mit einem Gehalt < 500 und > 200 (mit BETWEEN)

SELECT l\_vorname, l\_name, l\_gehaltFROM lehrerWHERE l\_gehalt BETWEEN 201 AND 499;

Ausgabe:



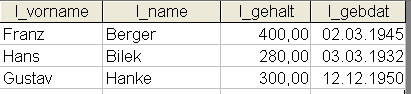
Das BETWEEN einzusetzen, erspart Schreibarbeit, konkret ersetzt ein ‚spaltenname BETWEEN untergrenze AND obergrenze’ eine Bedingung der Art ‚spaltenname >= untergrenze AND spaltenname <= obergrenze’.

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

* + 1. Beispiel 8: Lehrergehalt < 500, > 200 und Geburtsdatum vor 1.1.1955

SELECT l\_vorname, l\_name, l\_gehalt, l\_gebdatFROM lehrerWHERE (l\_gehalt BETWEEN 201 AND 499) AND l\_gebdat < #01/01/1955#;

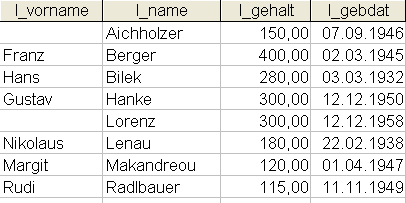
Ausgabe:



* + 1. Beispiel 9: Lehrer mit Gehalt größer 250 oder Geburtsdatum vor 1.1.1955

SELECT l\_vorname, l\_name, l\_gehalt, l\_gebdatFROM lehrerWHERE l\_gehalt > 250 OR l\_gebdat < #01/01/1955#;

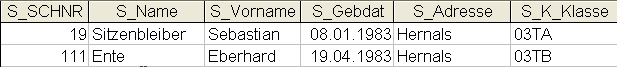
Ausgabe:



* + 1. Beispiel 10: Schüler mit Wohnort Hernals

SELECT \* FROM schuelerWHERE s\_adresse = ‘Hernals’;

Ausgabe:



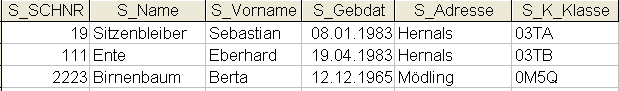
Ein Vergleich auf Gleichheit (mit =) erfordert bei numerischen Werten keine Hochkommata, bei Strings jedoch schon.

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

* + 1. Beispiel 11: Schüler mit Wohnort Hernals oder Mödling

SELECT \* FROM schuelerWHERE s\_adresse = ‘Hernals’ OR s\_adresse = ’Mödling’;

Ausgabe:



* + 1. Beispiel 12: Schüler der 03TA mit Wohnort Hernals

SELECT \* FROM schuelerWHERE s\_adresse = ‘Hernals’ AND s\_k\_klasse = ’03TA’;

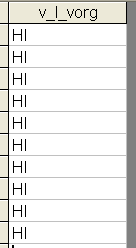
Ausgabe:



* + 1. Beispiel 13: Direktoren

SELECT v\_l\_vorg FROM vorgesetzteWHERE v\_art = ‘Direktor’;

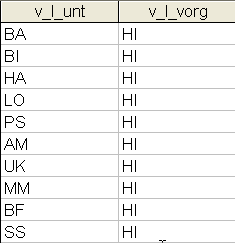
Ausgabe:



* + 1. Beispiel 14: Untergebene Lehrer von Direktoren

SELECT v\_l\_unt, v\_l\_vorg FROM vorgesetzteWHERE v\_art = ‘Direktor’;

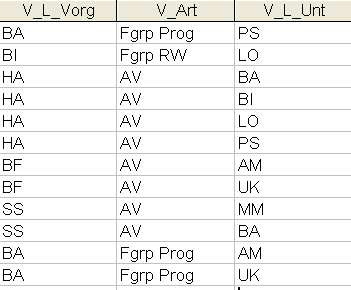
Ausgabe:



* + 1. Beispiel 15: Untergebene Lehrer von Nicht-Direktoren

SELECT \* FROM v orgesetzteWHERE v\_art <> ‘Direktor’;

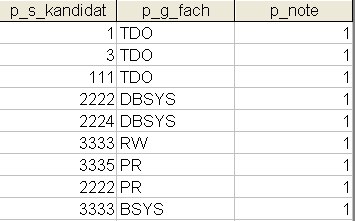
Ausgabe:



* + 1. Beispiel 16: Schüler mit einer 1 auf eine beliebige Prüfung

SELECT p\_s\_kandidat, p\_g\_fach, p\_noteFROM pruefungenWHERE p\_note = 1;

Ausgabe:



* + 1. Beispiel 17: Schüler mit einer 1 in Englisch

SELECT p\_s\_kandidat, p\_g\_fach, p\_noteFROM pruefungenWHERE p\_note = 1 AND p\_g\_fach = ‘EN’;

Ausgabe:



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

* + 1. Beispiel 18: Lehrergehalt 100 oder 220 und Geburtsdatum vor 1.1.1956 (mit / ohne Klammern)

SELECT l\_vorname, l\_name, l\_gehalt, l\_gebdatFROM lehrerWHERE (l\_gehalt = 100 OR l\_gehalt = 220) AND l\_gebdat < #01/01/1956#;

Ausgabe:

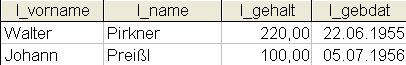


Klammern dienen einerseits dazu, die Übersichtlichkeit zu erhöhen, andererseits auch zur Festlegung einer – von der Standardpriorisierung (AND geht vor OR) – abweichenden Auswertungsreihenfolge der einzelnen Bedingungsteile.

* + 1. Beispiel 19: Lehrergehalt 100 oder 220 und Geburtsdatum vor 1.1.1956 (mit / ohne Klammern)

SELECT l\_vorname, l\_name, l\_gehalt, l\_gebdatFROM lehrerWHERE l\_gehalt = 100 OR l\_gehalt = 220 AND l\_gebdat < #01/01/1956#;

Ausgabe:

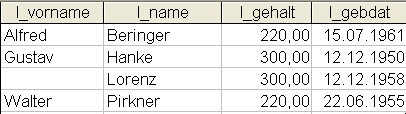


Vergleichen Sie das Ergebnis aus Beispiel 18 mit dem aus Beispiel 19.

* + 1. Beispiel 20: Lehrergehalt 82, 220, 300 und Geburtsdatum vor 1.1.1970

SELECT l\_vorname, l\_name, l\_gehalt, l\_gebdatFROM lehrerWHERE (l\_gehalt = 82 OR l\_gehalt = 220 OR l\_gehalt = 300) AND (l\_gebdat < #01/01/1970#);

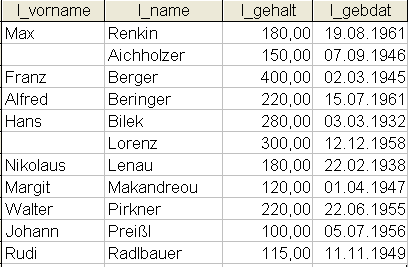
Ausgabe:



* + 1. Beispiel 21: Lehrer mit Geburtsdatum ungleich 12.12.1950

SELECT l\_vorname, l\_name, l\_gehalt, l\_gebdatFROM lehrerWHERE l\_gebdat <> #12/12/1950#;

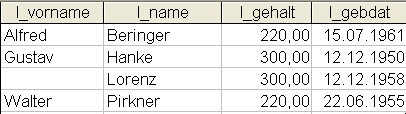
Ausgabe:



* + 1. Beispiel 22: Lehrergehalt 82, 220, 300 und Geburtsdatum vor 1.1.1970

SELECT l\_vorname, l\_name, l\_gehalt, l\_gebdatFROM lehrerWHERE (l\_gehalt IN (82, 220, 300)) AND (l\_gebdat < #01/01/1970#);

Ausgabe:



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);

Qualifizierung

Vor dem Spaltenname kann, mit einem Punkt voneinander getrennt, der Tabellenname 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 Tabellenname angegeben werden.Beispiel 23: Schüler und deren Geburtstag (ohne Qualifizierung)

SELECT s\_name, s\_gebdat FROM schueler;

Ausgabe:



Beispiel 24: Schüler und deren Geburtstag (mit Qualifizierung)

SELECT schueler.s\_name, schueler.s\_gebdat FROM schueler;

Ausgabe:



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;

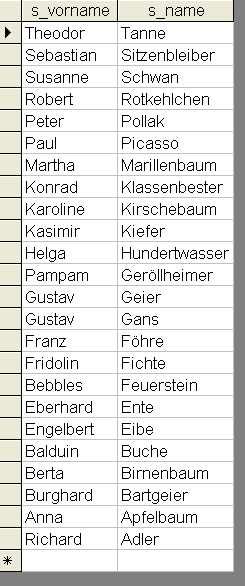
Dem Schlüsselwort ORDER BY folgt der Name der Spalte/n, nach der/denen sortiert werden soll.

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

* + 1. Beispiel 25: Alphabetisch absteigend sortierte Schülerliste

SELECT s\_vorname, s\_nameFROM schuelerORDER BY s\_name DESC;

Ausgabe:

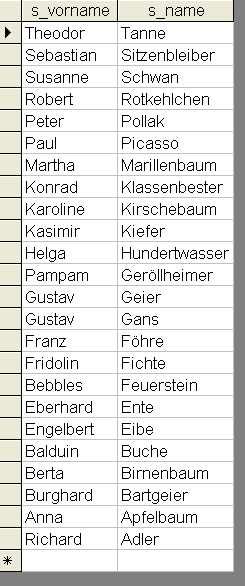


* + 1. Beispiel 26: Alphabetisch absteigend sortierte Schülerliste (etwas kürzer)

SELECT s\_vorname, s\_nameFROM schuelerORDER 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.

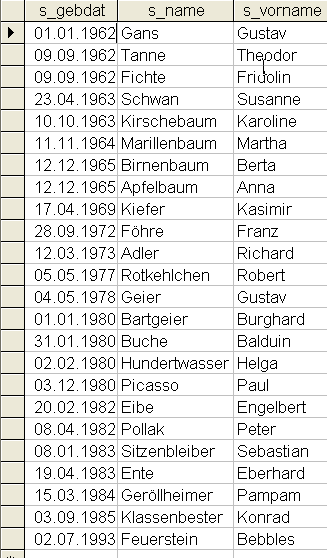
Ausgabe:



* + 1. Beispiel 27: Zweifach sortierte Schülerliste

SELECT s\_gebdat, s\_name, s\_vornameFROM schuelerORDER BY s\_gebdat ASC, s\_name DESC;

Ausgabe:



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

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;

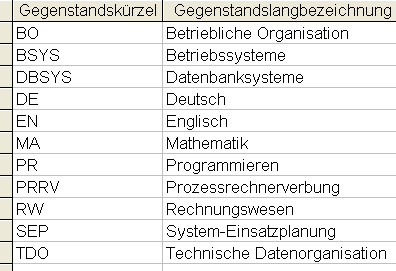
Alias-Namen dürfen keine Leerzeichen beinhalten.

(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 GegenstandslangbezeichnungFROM gegenstaendeORDER BY g\_id;

Ausgabe:



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\_gebdatFROM lehrerWHERE l\_name = ‘Gustav’;

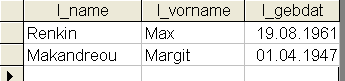
Ausgabe:



* + 1. Beispiel 30: Lehrer, deren Vorname mit M beginnt

SELECT l\_name, l\_vorname, l\_gebdatFROM lehrerWHERE l\_vorname LIKE ‘M\*’;

Ausgabe:



* + 1. 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 GeburtsdatumFROM lehrerWHERE l\_name LIKE ‘B?r\*’;

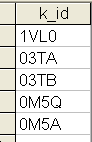
Ausgabe:



* + 1. Beispiel 32: Klassen mit eine 0 an beliebiger Stelle

SELECT k\_idFROM klassenWHERE k\_id LIKE ‘\*0\*’;

Ausgabe:



Duplikate

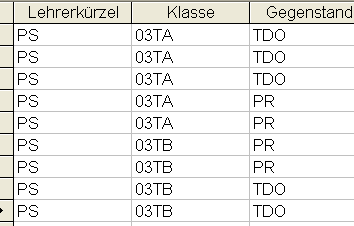
In allen Abfragen, in denen nicht das Primärschlüssel-Attribut Teil des Ergebnisses ist, kann es zu identen Resultat-Datensätzen kommen. Um diese herauszufiltern, ist das Schlüsselwort DISTINCT dem SELECT nachzustellen.

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 GegenstandFROM stundenWHERE st\_l\_lehrer = ’PS’;

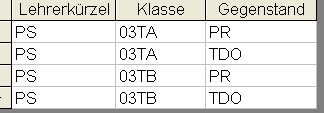
Ausgabe:



* + 1. 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 GegenstandFROM stundenWHERE st\_l\_lehrer = ’PS’;

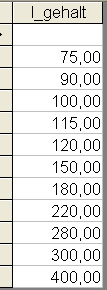
Ausgabe:



* + 1. Beispiel 35: Gehälter der Lehrer (ohne Duplikate)

SELECT DISTINCT l\_gehalt FROM lehrer;

Ausgabe:

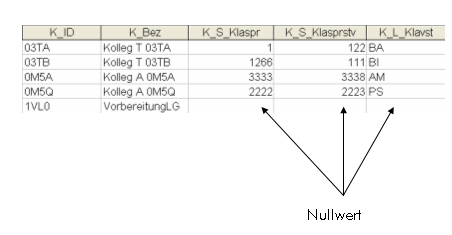


Die erste, scheinbar leere Ergebniszeile ist kein Fehler, sondern die Konsequenz der Tatsache, dass nicht bei allen Lehrern Gehälter vermerkt sind, i.e. Nullwerte vorkommen.

* 1. 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!



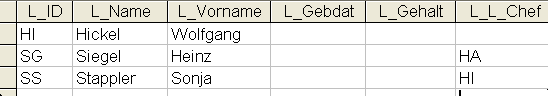
Nullwerte in Tabellen sind bei der Gestaltung von Abfragen zu berücksichtigen – beispielsweise bei der Einschränkung auf Zeilen, in denen eine gewisse Spalte keine Nullwerte beinhaltet, oder bei der Auswertung von logischen Ausdrücken.

SELECT \* FROM tabellenname WHERE spaltenname IS NULL;

* + 1. Beispiel 36: Lehrer ohne Gehalt

SELECT \* FROM lehrerWHERE l\_gehalt IS NULLORDER BY l\_id;

Ausgabe:



* + 1. Beispiel 37: Lehrer mit bekanntem Gehalt, aufsteigend sortiert

SELECT l\_id, l\_name, l\_vorname, l\_gehalt FROM lehrerWHERE l\_gehalt IS NOT NULLORDER BY l\_gehalt;

Ausgabe:



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 |

* 1. Konstante

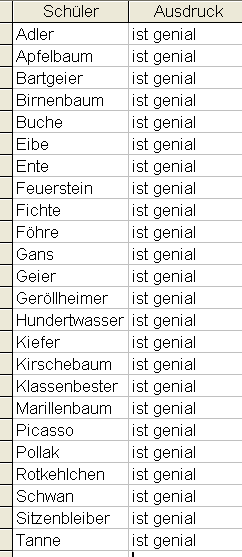
In der Ergebnistabelle können nicht nur Werte auftreten, die in der Ausgangstabelle vorliegen, sondern auch nach dem SELECT definierte Konstanten.

SELECT spaltenname, konstante FROM tabellenname;

* + 1. Beispiel 38: Geniale Schüler

SELECT s\_name AS Schüler, ‘ist genial’ AS AusdruckFROM schuelerORDER BY s\_name;

Ausgabe:



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:



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 ür die Divisionmod für die Modulo-Operation[[10]](#footnote-10)

* + 1. Beispiel 40: Aggregationswert, Konstante, Ausdruck für Lehrer mit Chef HI

SELECT COUNT(\*), ‘konstante’, 1, COUNT(\*) \* 10 FROM lehrerWHERE l\_l\_chef = ‘HI’;

Ausgabe:



* + 1. 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\_LehrergehaltFROM lehrer;

Ausgabe:



* + 1. Beispiel 42: Geburtsdatum des jüngsten Schülers

SELECT MAX(s\_gebdat) AS Geburtsdatum\_des\_jüngsten\_SchülersFROM schueler; Ausgabe:



* + 1. Beispiel 43: Geburtsdatum des ältesten Schülers

SELECT MIN(s\_gebdat) AS Geburtsdatum\_des\_ältesten\_SchülersFROM schueler;

Ausgabe:



* + 1. Beispiel 44: Ausgabe des heutigen Datums

SELECT DISTINCT Now() AS HeuteFROM lehrer;

Ausgabe:



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 |

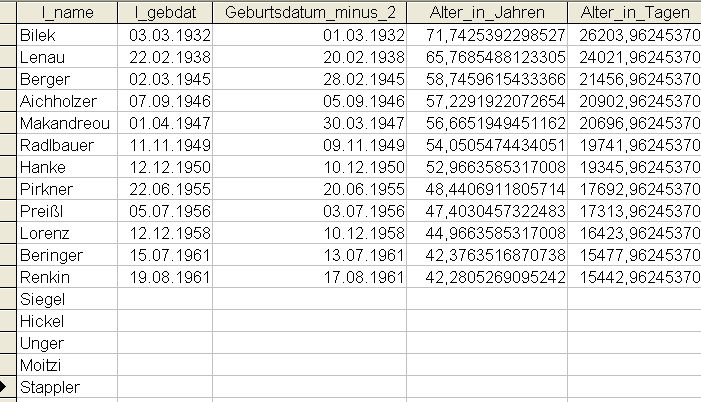
MySQL:

|  |  |
| --- | --- |
|  | 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 |

* + 1. 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\_TagenFROM lehrerORDER BY 4 DESC;

Ausgabe:



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\_klasseFROM schuelerWHERE s\_k\_klasse = ’03TA’ORDER BY s\_name;

Ausgabe:



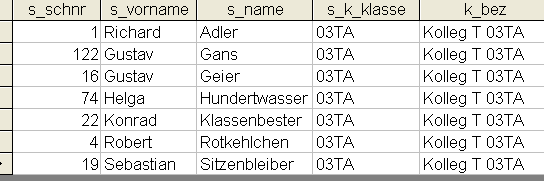
* + 1. Beispiel 47: Schüler der Klasse 03TA mit Klassenbezeichnung

SELECT s\_schnr, s\_vorname, s\_name, s\_k\_klasse, k\_bezFROM schueler, klassenWHERE s\_k\_klasse = ’03TA’ AND s\_k\_klasse = k\_idORDER BY s\_name;

oder

SELECT schueler.s\_schnr, schueler.s\_vorname, schueler.s\_name, klassen.s\_k\_klasse, klassen.k\_bezFROM schueler INNER JOIN klassen ON schueler.s\_k\_klasse = klassen.k\_idWHERE schueler.s\_k\_klasse = ’03TA’ORDER BY s\_name;

Ausgabe:



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.

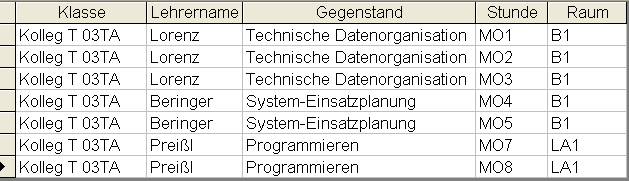
* + 1. Beispiel 48: Stundenplan für Montag in der 03TA mit Gegenstandsnamen

SELECT k\_bez AS Klasse, l\_name AS Lehrername, g\_bez AS Gegenstand, st\_stunde AS Stunde, st\_r\_raum AS RaumFROM stunden, klassen, lehrer, gegenstaendeWHERE 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 RaumFROM ((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\_idWHERE stunden.st\_stunde LIKE ’MO\*’ AND stunden.st\_k\_klasse = ’03TA’ ORDER BY k\_bez, st\_stunde;

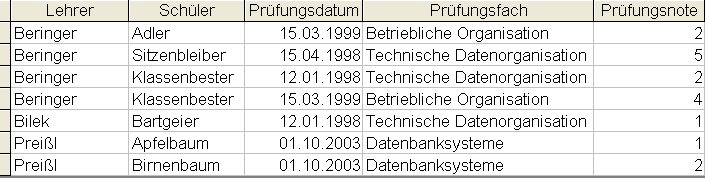
Ausgabe:



* + 1. 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 PrüfungsnoteFROM lehrer, schueler, klassen, pruefungen, gegenstaendeWHERE 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:



* + 1. Beispiel 50: Schüleranzahl der 03TB

SELECT COUNT(\*) AS Schüleranzahl FROM schueler WHERE s\_k\_klasse = ’03TB’;

Ausgabe:



* + 1. Beispiel 51: Schüleranzahl der 03TB mit Klassenlangname

SELECT COUNT(\*) AS Schüleranzahl, MAX(k\_bez) AS Klassenlangname FROM schueler, klassenWHERE 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:



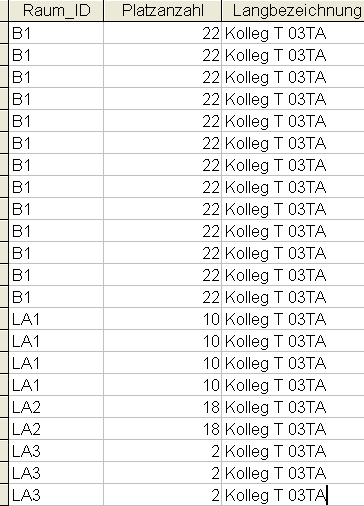
* + 1. Beispiel 52: Sitzplätze der Lehrsäle und Labors der 03TA (mit Duplikaten)

SELECT r\_id AS Raum\_ID, r\_plaetze AS Platzanzahl, k\_bez AS LangbezeichnungFROM raeume, stunden, klassenWHERE 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 LangbezeichnungFROM (klassen INNER JOIN stunden ON klassen.k\_id = stunden.st\_k\_klasse) INNER JOIN raeume ON stunden.st\_r\_raum = raeume.r\_idWHERE klassen.k\_id = ’03TA’ORDER BY raeume.r\_id, raeume.r\_plaetze DESC;

Ausgabe:



* + 1. 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 LangbezeichnungFROM raeume, stunden, klassenWHERE 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 LangbezeichnungFROM (klassen INNER JOIN stunden ON klassen.k\_id = stunden.st\_k\_klasse) INNER JOIN raeume ON stunden.st\_r\_raum = raeume.r\_idWHERE klassen.k\_id = ’03TA’ORDER BY raeume.r\_id, raeume.r\_plaetze DESC;

Ausgabe:



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\_GehaltFROM lehrer AS chef, lehrer AS untergebenerWHERE 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\_GehaltFROM lehrer AS chef INNER JOIN lehrer AS untergebener ON untergebener.l\_l\_chef = chef.l\_idWHERE untergebener.l\_gehalt = chef.l\_gehalt;

Ausgabe:



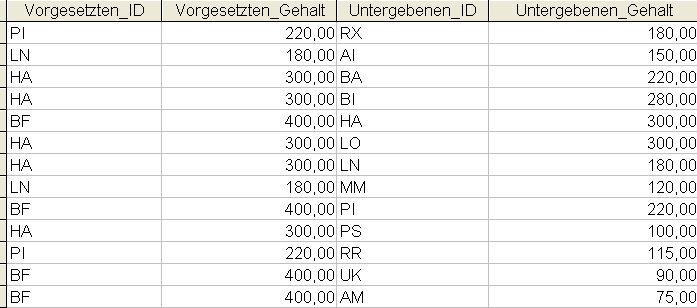
* + 1. Beispiel 55: Lehrer mit Gehalt kleiner gleich dem des direkten Vorgesetzten

SELECT chef.l\_id AS Vorgesetzten\_ID, chef.l\_gehalt AS Vorgesetzten\_Gehalt, untergebener.l\_id AS Untergebenen\_ID, untergebener.l\_gehalt AS Untergebenen\_GehaltFROM lehrer AS chef, lehrer AS untergebenerWHERE 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\_GehaltFROM lehrer AS chef INNER JOIN lehrer AS untergebener ON untergebener.l\_l\_chef = chef.l\_id WHERE untergebener.l\_gehalt <= chef.l\_gehalt;

Ausgabe:



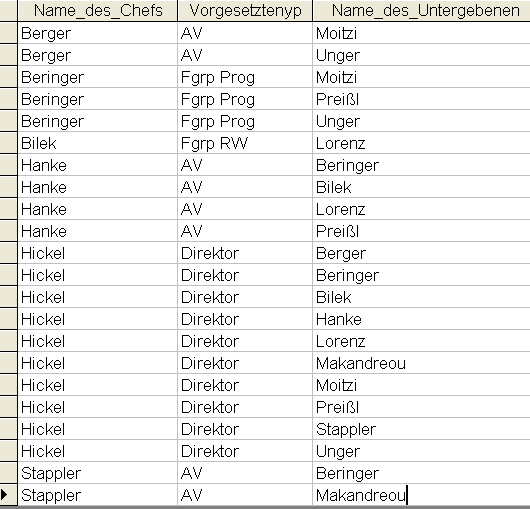
* + 1. Beispiel 56: Liste aller Vorgesetztenbeziehungen

SELECT chef.l\_name AS Name\_des\_Chefs, v\_art AS Vorgesetztenyp, untergebener.l\_name AS Name\_des\_UntergebenenFROM lehrer AS chef, vorgesetzte, lehrer AS untergebener WHERE chef.l\_id = vorgesetzte.v\_l\_vorg AND untergebener.l\_id = vorgesetzte.v\_l\_untORDER 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\_UntergebenenFROM (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\_untORDER BY chef.l\_name ASC, untergebener.l\_name;

Ausgabe:

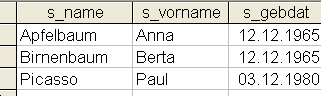


Datumswerte

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

SELECT s\_name, s\_vorname, s\_gebdat FROM schuelerWHERE month(s\_gebdat) = ’12’ORDER BY s\_name;

Ausgabe:



* + 1. Beispiel 58: Ausgabe der Schüler, die am 3.12.1980 Geburtstag haben

SELECT s\_name, s\_vorname, s\_gebdat FROM schuelerWHERE d ay(s\_gebdat) = ’3’ AND month(s\_gebdat) = ’12’ AND year(s\_gebdat) = ’1980’;Ausgabe:



* + 1. 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 LehrergeburtsjahrFROM lehrer, schuelerWHERE year(s\_gebdat) > year(l\_gebdat) AND l\_id = ‘HA’ORDER BY s\_gebdat;

Ausgabe:



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 des öfteren der Begriff ‚Gruppenwechsel’.

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

* + 1. Beispiel 60: Schüleranzahl

SELECT COUNT(s\_schnr) AS SchüleranzahlFROM schueler;

Ausgabe:



* + 1. Beispiel 61: Schüleranzahl je Klasse

SELECT COUNT(s\_schnr) AS Schüleranzahl, s\_k\_klasse AS Klasse FROM schuelerGROUP BY s\_k\_klasse;

Ausgabe:



* 1. Gruppeneinschränkung

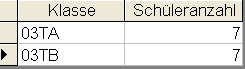
Die ermittelten Ausgabedatensätze der Gruppierung können mittels HAVING nochmals eingeschränkt werden.

SELECT aggregatfunktion(spaltenname1), spaltenname2 FROM tabellenname WHERE bedingung GROUP BY spaltenname2HAVING aggregatfunktion(spaltenname1) vergleichsoperator vergleichswert;

* + 1. Beispiel 62: Klassen mit mehr als 6 Schülern

SELECT s\_k\_klasse AS Klasse, COUNT(\*) AS SchüleranzahlFROM schuelerGROUP BY s\_k\_klasseHAVING COUNT(\*) > 6;

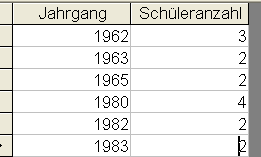
Ausgabe:



* + 1. Beispiel 63: Schüleranzahl je Jahrgang mit mehr als einem Schüler

SELECT year(s\_gebdat) AS Jahrgang, COUNT(\*) AS SchüleranzahlFROM schuelerGROUP BY year(s\_gebdat)HAVING COUNT(\*) > 1;

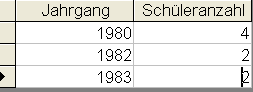
Ausgabe:



* + 1. Beispiel 64: Jahrgänge ab 1975 mit mehr als einem Schüler

SELECT year(s\_gebdat) AS Jahrgang, COUNT(\*) AS SchüleranzahlFROM schuelerWHERE s\_gebdat >= #01/01/1975#GROUP BY year(s\_gebdat) HAVING COUNT(\*) > 1ORDER BY 1;

Ausgabe:

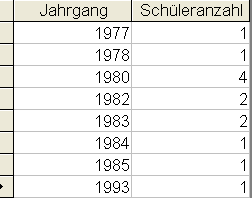


WHERE und HAVING dürfen nicht verwechselt werden!

* + 1. Beispiel 65: Jahrgänge ab 1975

SELECT year(s\_gebdat) AS Jahrgang, COUNT(\*) AS SchüleranzahlFROM schuelerWHERE s\_gebdat >= #01/01/1975#GROUP BY year(s\_gebdat) ORDER BY 1;

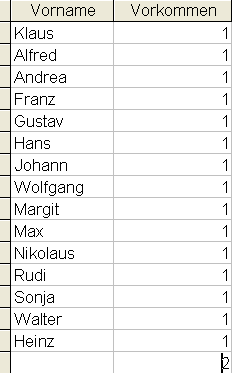
Ausgabe:



* + 1. Beispiel 66: Vornamenhäufigkeit bei Lehrern (mit Nullwert)

SELECT l\_vorname AS Vorname, COUNT(\*) AS VorkommenFROM lehrer GROUP BY l\_vornameORDER BY COUNT(\*);

Ausgabe:



* + 1. Beispiel 67: Vornamenhäufigkeit bei Lehrern (ohne Nullwert)

SELECT l\_vorname AS Vorname, COUNT(\*) AS VorkommenFROM lehrer WHERE l\_vorname IS NOT NULLGROUP BY l\_vornameORDER BY COUNT(\*);

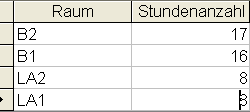
Ausgabe:



* + 1. Beispiel 68: Lehrsäle mit mehr als 5 Stunden Unterricht

SELECT st\_r\_raum AS Raum, COUNT(\*) AS StundenanzahlFROM stunden GROUP BY st\_r\_raum HAVING COUNT(\*) > 5ORDER BY COUNT(\*) DESC;

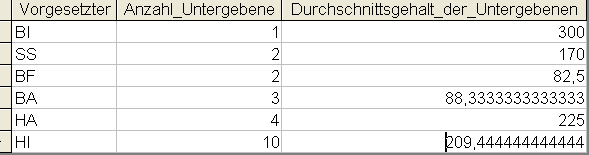
Ausgabe:



* + 1. 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\_UntergebenenFROM lehrer AS vorgesetzter, lehrer AS untergebene, vorgesetzteWHERE vorgesetzter.l\_id = v\_l\_vorg AND v\_l\_unt = untergebene.l\_idGROUP BY vorgesetzter.l\_idORDER BY 2;

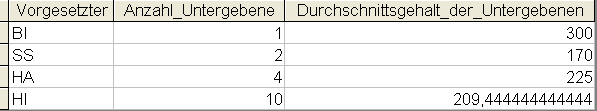
Ausgabe:



* + 1. Beispiel 70: Vorgesetzte mit Durchschnittsgehalt > 150 der Untergebenen

SELECT vorgesetzter.l\_id AS Vorgesetzter, COUNT(\*) AS Anzahl\_Untergebene, AVG(untergebene.l\_gehalt) AS Durchschnittsgehalt\_der\_UntergebenenFROM lehrer AS vorgesetzter, lehrer AS untergebene, vorgesetzteWHERE vorgesetzter.l\_id = v\_l\_vorg AND v\_l\_unt = untergebene.l\_idGROUP BY vorgesetzter.l\_idHAVING AVG(untergebene.l\_gehalt) > 150ORDER BY 2;

Ausgabe:



* + 1. Beispiel 71: Unterrichtsstunden je Lehrer

SELECT MAX(l\_name) AS Lehrer, l\_id AS Lehrerkürzel, COUNT(\*) AS StundenanzahlFROM lehrer, stundenWHERE l\_id = st\_l\_lehrerGROUP BY l\_idORDER BY 1;

oder

SELECT MAX(lehrer.l\_name) AS Lehrer, lehrer.l\_id AS Lehrerkürzel, COUNT(\*) AS StundenanzahlFROM lehrer INNER JOIN stunden ON lehrer.l\_id = stunden.st\_l\_lehrerGROUP BY lehrer.l\_idORDER BY 1;

Ausgabe:



* + 1. Beispiel 72: Lehrer, die mehr als 10 Stunden unterrichten

SELECT MAX(l\_name) AS Lehrer, l\_id AS Lehrerkürzel, COUNT(\*) AS StundenanzahlFROM lehrer, stundenWHERE l\_id = st\_l\_lehrerGROUP BY l\_idHAVING COUNT(\*) > 10ORDER BY 1;

oder

SELECT MAX(lehrer.l\_name) AS Lehrer, lehrer.l\_id AS Lehrerkürzel, COUNT(\*) AS StundenanzahlFROM lehrer INNER JOIN stunden ON lehrer.l\_id = stunden.st\_l\_lehrerGROUP BY lehrer.l\_idHAVING COUNT(\*) > 10ORDER BY 1;

Ausgabe:



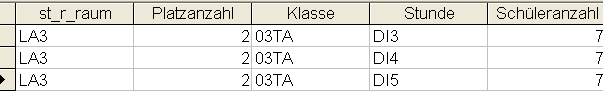
* + 1. Beispiel 73: Stunden mit mehr Schülern als Sitzplätzen

SELECT st\_r\_raum, MAX(r\_plaetze) AS Platzanzahl, k\_id AS Klasse, st\_stunde AS Stunde, COUNT(\*) AS SchüleranzahlFROM raeume, stunden, klassen, schuelerWHERE st\_r\_raum = r\_id AND st\_k\_klasse = k\_id AND k\_id = s\_k\_klasseGROUP BY st\_r\_raum, st\_stunde, k\_idHAVING (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 SchüleranzahlFROM ((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\_idGROUP BY stunden.st\_r\_raum, stunden.st\_stunde, klassen.k\_idHAVING (MAX(raeume.r\_plaetze) - COUNT(\*)) < 0;

Ausgabe:



* 1. Subselect

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

SELECT spaltenname1 FROM tabellenname1WHERE 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 GeburtsdatumFROM schuelerWHERE s\_gebdat = (SELECT MAX(s\_gebdat) FROM schueler);

Ausgabe:



* + 1. Beispiel 75: Schüler, die bereits Prüfungen absolviert haben (Subselect)

SELECT DISTINCT s\_vorname AS Vorname, s\_name AS NameFROM schuelerWHERE s\_schnr IN (SELECT p\_s\_kandidat FROM pruefungen);

Ausgabe:



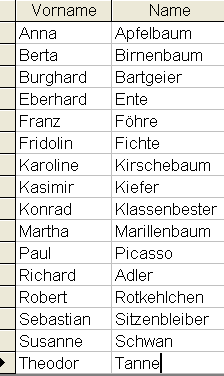
* + 1. Beispiel 76: Schüler mit absolvierten Prüfungen (Join)

SELECT DISTINCT s\_vorname AS Vorname, s\_name AS NameFROM schueler, pruefungenWHERE s\_schnr = p\_s\_kandidat;

oder

SELECT DISTINCT schueler.s\_vorname AS Vorname, schueler.s\_name AS NameFROM schueler INNER JOIN pruefungen ON schueler.s\_schnr = pruefungen.p\_s\_kandidat;

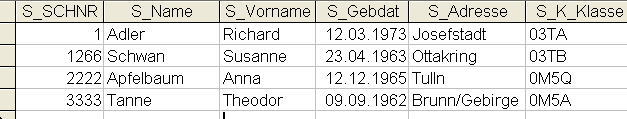
Ausgabe:



* + 1. Beispiel 77: Klassensprecher

SELECT \*FROM schuelerWHERE s\_schnr IN (SELECT k\_s\_klaspr FROM klassen);

Ausgabe:



* + 1. Beispiel 78: Klassensprecher der Klasse mit BA als Klassenvorstand

SELECT s\_schnr, s\_vorname, s\_name, s\_k\_klasseFROM schuelerWHERE s\_schnr IN (SELECT k\_s\_klaspr FROM klassen WHERE k\_l\_klavst = ‘BA’);

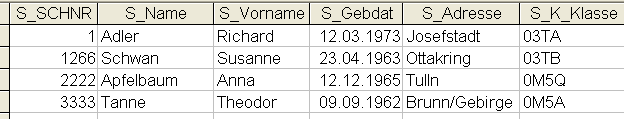
Ausgabe:



* + 1. Beispiel 79: Klassensprecher (mit ANY)

SELECT \*FROM schuelerWHERE s\_schnr = ANY (SELECT k\_s\_klaspr FROM klassen);

Ausgabe:



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

* + 1. Beispiel 80: Lehrer mit dem höchsten bekannten Gehalt (mit ALL)

SELECT \*FROM lehrerWHERE l\_gehalt >= ALL (SELECT l\_gehalt FROM lehrer WHERE l\_gehalt IS NOT NULL);

Ausgabe:



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.

* + 1. Beispiel 81: Lehrer mit dem niedrigsten bekannten Gehalt

SELECT \*FROM lehrerWHERE l\_gehalt <= ALL (SELECT l\_gehalt FROM lehrer WHERE l\_gehalt IS NOT NULL);

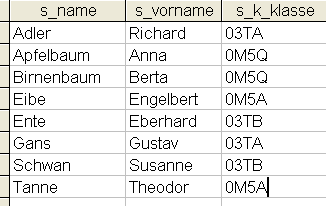
Ausgabe:



* + 1. Beispiel 82: Klassensprecher und Klassensprecherstellvertreter (EXISTS)

SELECT s\_name, s\_vorname, s\_k\_klasseFROM schuelerWHERE 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:



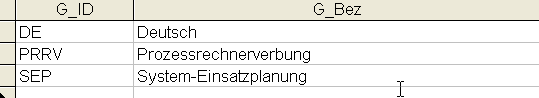
SELECT \* FROM tabellenname WHERE EXISTS (subselect);

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.

* + 1. Beispiel 83: Ungeprüfte Gegenstände

SELECT \*FROM gegenstaendeWHERE g\_id NOT IN (SELECT DISTINCT p\_g\_fach FROM pruefungen)ORDER BY g\_id;

Ausgabe:



* + 1. Beispiel 84: Unterrichtende, aber nicht prüfende Lehrer

SELECT l\_name, l\_vornameFROM lehrerWHERE 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:



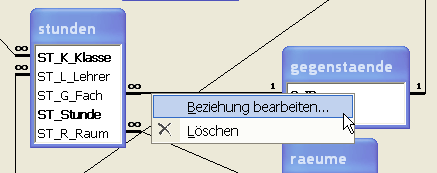
* + 1. Beispiel 85: Falsche Fremdschlüssel im Stundenplan

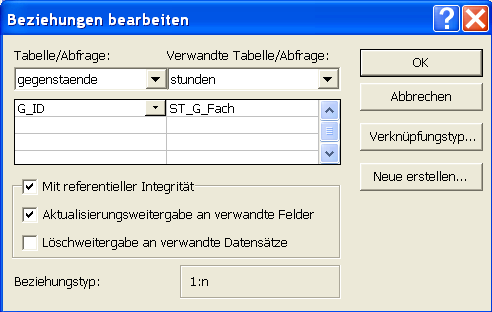
SELECT \*FROM stundenWHERE NOT EXISTS (SELECT \* FROM lehrer WHERE l\_id = st\_l\_lehrer) OR NOT EXISTS (SELECT \* FROM klassen WHERE k\_id = st\_k\_klasse) OR NOT EXISTS (SELECT \* FROM gegenstaende WHERE g\_id = st\_g\_fach) OR NOT EXISTS (SELECT \* FROM raeume WHERE r\_id = st\_r\_raum);

Ausgabe:



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





* + 1. Beispiel 86: Lehrer mit nicht existierenden Vorgesetzten

SELECT \*FROM lehrerWHERE l\_l\_chef NOT IN (SELECT l\_id FROM lehrer);

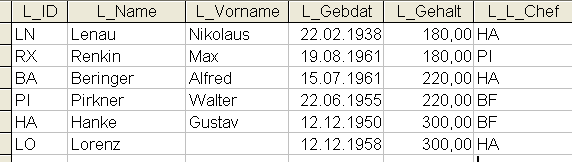
Ausgabe:



* + 1. Beispiel 87: Lehrer mit gleichem Gehalt

SELECT \*FROM lehrerWHERE l\_gehalt = ANY (SELECT l\_gehalt FROM lehrer GROUP BY l\_gehalt HAVING COUNT(\*) > 1)ORDER BY l\_gehalt, l\_id;

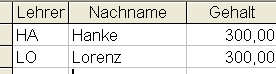
Ausgabe:



* + 1. Beispiel 88: Lehrer mit dem zweithöchsten Gehalt

SELECT l\_id AS Lehrer, l\_name AS Nachname, l\_gehalt AS GehaltFROM lehrerWHERE 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:



Outer Join

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

SELECT \* FROM tabellenname1 RIGHT JOIN tabellenname2ON 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 DurchschnittsnoteFROM gegenstaende LEFT JOIN pruefungen ON gegenstaende.g\_id = pruefungen.p\_g\_fachGROUP BY gegenstaende.g\_id;

Ausgabe:



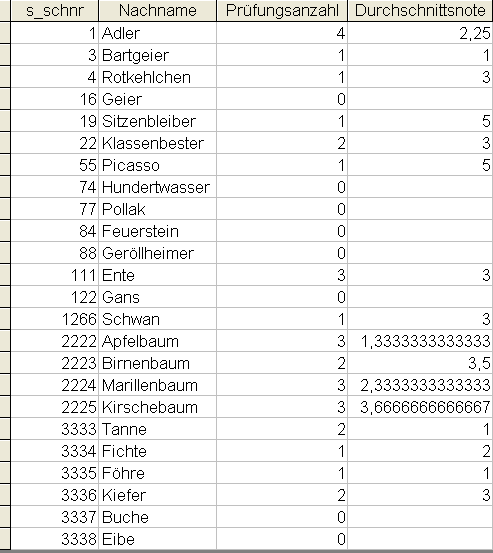
Ein einfacher, innerer Verbund (… FROM gegenstaende INNER JOIN pruefungen ON …) würde nur die Durchschnittsnoten derjenigen Gegenstände auflisten, zu denen auch Prüfungseinträge in der Tabelle pruefungen abgespeichert sind.

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

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 DurchschnittsnoteFROM schueler LEFT JOIN pruefungen ON schueler.s\_schnr = pruefungen.p\_s\_kandidatGROUP BY schueler.s\_schnr;

Ausgabe:



Mengenoperationen

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

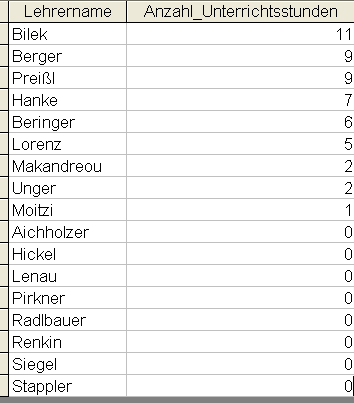
SELECT spaltenname1 FROM tabellenname1 UNIONSELECT spaltenname1 FROM tabellenname2;

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 lehrerWHERE l\_id NOT IN (SELECT st\_l\_lehrer FROM stunden) UNIONSELECT MAX(l\_name) AS Lehrername, COUNT(\*)FROM lehrer, stundenWHERE l\_id = st\_l\_lehrerGROUP BY l\_idORDER BY 2 DESC;

Ausgabe:



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

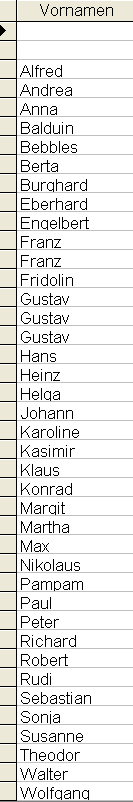
SELECT l\_vorname AS VornamenFROM lehrerUNIONSELECT s\_vornameFROM schuelerORDER BY 1;Ausgabe:



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

SELECT l\_vorname AS VornamenFROM lehrerUNION ALLSELECT s\_vornameFROM schuelerORDER BY 1;

Ausgabe:

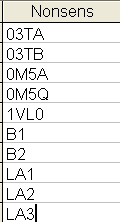


SELECT spaltenname1 FROM tabellenname1 UNION ALLSELECT spaltenname1 FROM tabellenname2;

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.

* + 1. Beispiel 94: Alle Klassen vereinigt mit allen Räumen

SELECT k\_id AS NonsensFROM klassenUNION SELECT r\_idFROM raeume;Ausgabe:



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.

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:



* + 1. Beispiel 96: Anzahl unterschiedlicher Lehrergehälter

SELECT COUNT(\*) AS Anzahl\_unterschiedliche\_GehälterFROM (SELECT DISTINCT l\_gehalt FROM lehrer);

Ausgabe:



* + 1. 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\_DurchschnittFROM (SELECT DISTINCT l\_gehalt FROM lehrer);

Ausgabe:



* + 1. 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 AnzahlFROM lehrerGROUP BY l\_gehaltHAVING COUNT(\*) >= ALL (SELECT COUNT(\*) FROM lehrer GROUP BY l\_gehalt);

Ausgabe:



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



* + 1. 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:



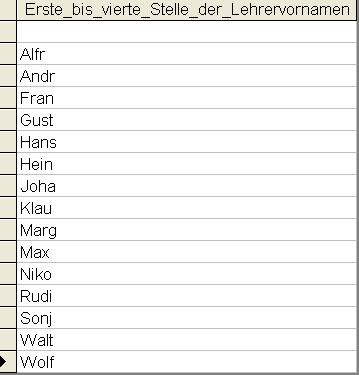
Extraktion von Stringbestandteilen

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\_LehrervornamenFROM lehrerORDER BY 1;

Ausgabe:



* + 1. Beispiel 101: Letzter und vorletzter Buchstabe alle Lehrervornamen

SELECT DISTINCT RIGHT(l\_vorname, 2) AS Letzte\_und\_vorletzte\_Stelle\_der\_LehrervornamenFROM lehrerORDER BY 1;

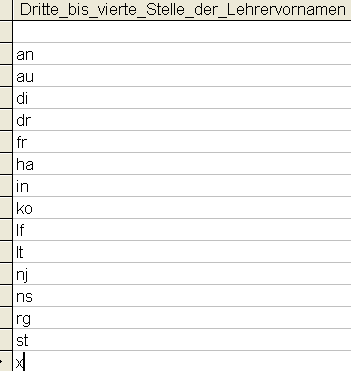
Ausgabe:



* + 1. Beispiel 102: Dritter bis vierter Buchstabe alle Lehrervornamen

SELECT DISTINCT MID(l\_vorname, 3, 2) AS Dritte\_bis\_vierte\_Stelle\_der\_LehrervornamenFROM lehrerORDER BY 1;

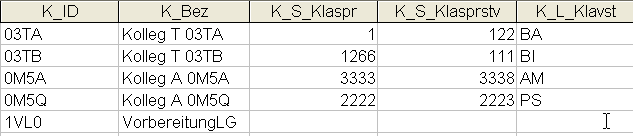
Ausgabe:



1. ANHANG A: Inhalt aller Tabellen der Schuldatenbank
   * 1. Tabelle gegenstaende



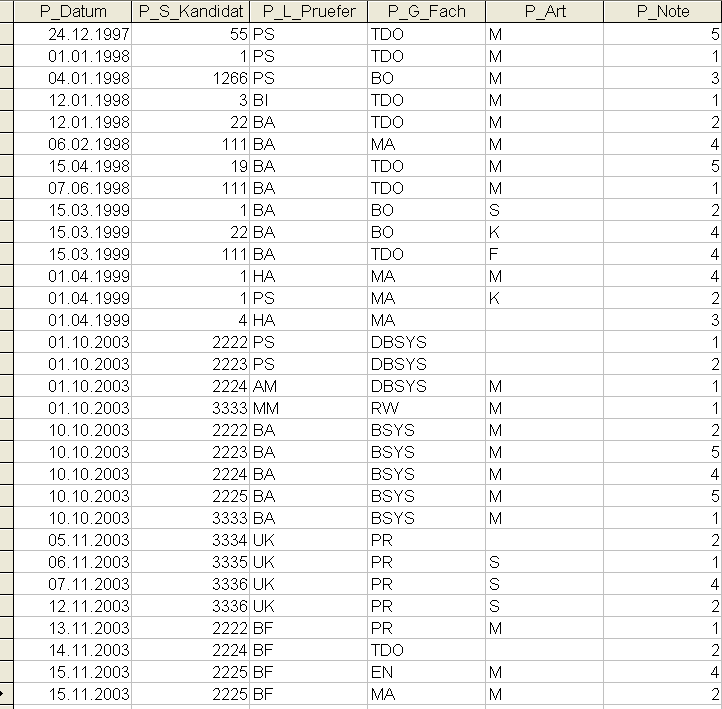
* + 1. Tabelle klassen



* + 1. Tabelle lehrer



Tabelle pruefungen



* + 1. Tabelle raeume

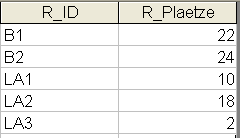


Tabelle schueler



Tabelle stunden

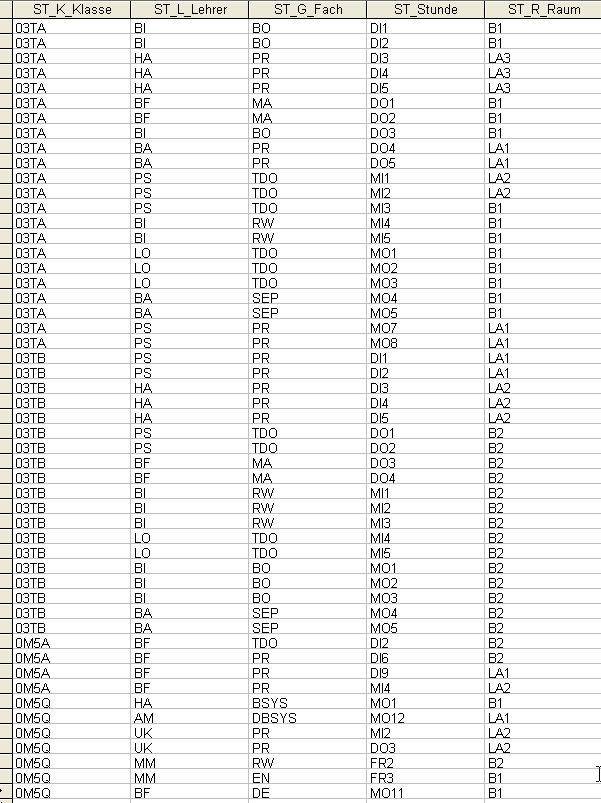


Tabelle vorgesetzte



1. 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 |
| 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 |
| 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 |
| 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.

1. 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 |
| .. Diplomarbeitsprojekte 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 … |

1. 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!

1. ANHANG E: Raum für Ihre Notizen

1. unter Nutzung von Vergleichsoperatoren, logischen Operatoren et.al. [↑](#footnote-ref-1)
2. Entity Relationship Diagramme / Modelle [↑](#footnote-ref-2)
3. American National Standards Institute [↑](#footnote-ref-3)
4. International Organization for Standardization [↑](#footnote-ref-4)
5. 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> ; dieser Seite wurde auch der obige Extrakt der wichtigsten Datentypen entnommen. [↑](#footnote-ref-5)
6. 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. [↑](#footnote-ref-6)
7. 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/einfuhrung-in-datentypen-und-feldeigenschaften-HA010341783.aspx> ; dieser Seite wurde auch der obige Extrakt der wichtigsten Datentypen entnommen. [↑](#footnote-ref-7)
8. 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. [↑](#footnote-ref-8)
9. .. und verlangsamt im Gegenzug ein Insert/Update. Ein Index sollte daher immer mit Bedacht gewählt werden. [↑](#footnote-ref-9)
10. Die Modulooperation a mod b liefert den Rest der ganzzahligen Division zurück. [↑](#footnote-ref-10)
11. Referentielle Integrität besagt, dass für jeden Fremdschlüsselwert (in der Tabelle A) der entsprechende Primärschlüsselwert (in der Tabelle B) vorhanden sein muss. [↑](#footnote-ref-11)