Prüfungsrelevantes Wissen „SQL“   
für die Prüfung zum   
Fachinformatiker für Anwendungsentwicklung

Autor und Urheberrecht: Matthias Brinkmann  
Das Dokument darf zur Prüfungsvorbereitung an andere Prüflinge derzeit kopiert und verteilt werden. Bei nachfolgenden Versionen kann es eventuell Einschränkungen geben, die in der jeweiligen Version vermerkt sind.

Hinweise zur aktuellen Version: Die Inhalts-Punkte 4.7.4 und 4.7.5 sind derzeit nur Platzhalter

Versionierung:

|  |  |
| --- | --- |
| 08.02.2022 | Zweiter Zwischenstand, alle Punkte ab 4.6.1 sind ergänzt, kleinere Änderungen an vorhergehenden Punkten, DROP DATABASE und TRUNCATE TABLE ergänzt |
| 09.02.2022 | Benennung der zunächst noch fehlenden Inhalten, Vormerkung zur Änderung der Reihenfolge |
| 15.02.2022 | Weitere Date-Funktionen, Constraints ergänzt |

Inhalt

[1. Datenbank-Operationen 3](#_Toc96062955)

[1.1 Datenbank anlegen: 3](#_Toc96062956)

[1.2 Datenbank benutzen: 3](#_Toc96062957)

[1.3 Datenbank löschen 3](#_Toc96062958)

[2. Tabellen anlegen / Struktur ändern und anzeigen 3](#_Toc96062959)

[2.1 Tabelle anlegen: 3](#_Toc96062960)

[2.2 Spalte ändern: 3](#_Toc96062961)

[2.3 Spalte hinfügen: 4](#_Toc96062962)

[2.4 Struktur der Tabelle anzeigen 4](#_Toc96062963)

[2.5 CREATE TABLE-Statement anzeigen 4](#_Toc96062964)

[2.6 DROP TABLE 4](#_Toc96062965)

[3. Daten bearbeiten 4](#_Toc96062966)

[3.1 Datensatz einfügen, Variante mit SET: 4](#_Toc96062967)

[3.2 Datensatz einfügen, Variante mit VALUES: 4](#_Toc96062968)

[3.3 Datensatz ändern: 4](#_Toc96062969)

[3.4 Datensatz löschen: 4](#_Toc96062970)

[3.5 DELETE FROM und TRUNCATE 5](#_Toc96062971)

[4. SELECT 5](#_Toc96062972)

[4.1 Vollständiges SELECT-Statement 5](#_Toc96062973)

[4.2 SELECT 5](#_Toc96062974)

[4.2.1 SELECT \* 5](#_Toc96062975)

[4.2.2 SELECT mit statischem Text 5](#_Toc96062976)

[4.2.3 SELECT mit Rechenoperationen 6](#_Toc96062977)

[4.2.4 SELECT mit zeilenbasierten Funktionen 6](#_Toc96062978)

[4.3 FROM 6](#_Toc96062979)

[4.4 JOINs 6](#_Toc96062980)

[4.4.1 Subselect 6](#_Toc96062981)

[4.5 WHERE 6](#_Toc96062982)

[4.5.1 LIKE 7](#_Toc96062983)

[4.5.2 BETWEEN 7](#_Toc96062984)

[4.5.3 IN 7](#_Toc96062985)

[4.5.4 NOT 7](#_Toc96062986)

[4.6 Nicht-Aggregierte Funktionen (Single-Row-Funktionen) in SELECT 7](#_Toc96062987)

[4.6.1 UPPER / LOWER 7](#_Toc96062988)

[4.6.2 CONCAT 8](#_Toc96062989)

[4.6.3 CONCAT\_WS 8](#_Toc96062990)

[4.6.4 MOD 8](#_Toc96062991)

[4.6.5 LEFT / RIGHT 8](#_Toc96062992)

[4.7 Filtern mit Datums- und Zeitwerten 8](#_Toc96062993)

[4.7.1 Umwandeln eines Timestamps in DATETIME 8](#_Toc96062994)

[4.7.2 Einen Wert zwischen zwei Datumswerten ermitteln 9](#_Toc96062995)

[4.7.3 Kleiner, Größer, Gleich mit Datumswerten 9](#_Toc96062996)

[4.7.4 Umwandeln/Formatieren von Datumswerten 9](#_Toc96062997)

[4.7.5 DATEDIFF 10](#_Toc96062998)

[4.7.5 TO\_DAYS 10](#_Toc96062999)

[5. SELECTS miteinander verbinden 10](#_Toc96063000)

[5.1 UNION 10](#_Toc96063001)

[6. Datentypen 10](#_Toc96063002)

[7. Daten kombinieren 11](#_Toc96063003)

[7.1 UNION 11](#_Toc96063004)

[7.2 INTERSECT 11](#_Toc96063005)

[7.3 MINUS 11](#_Toc96063006)

[8. Transaktions-Steuerung 11](#_Toc96063007)

[9. CONSTRAINTS 11](#_Toc96063008)

[9.1 Anlegen eines FOREIGN-KEY-Constraints: 11](#_Toc96063009)

[9.2 Löschen eines FOREIGN-KEY-Constraints 11](#_Toc96063010)

[10. Berechtigungen (GRANT) 11](#_Toc96063011)

[Fehlend: 12](#_Toc96063012)

[Erweiterndes Wissen: 12](#_Toc96063013)

[Anlage 1 Stichwort-Verzeichnis 12](#_Toc96063014)

[Anlage 2 Lexikon 12](#_Toc96063015)

# 1. Datenbank-Operationen

## 1.1 Datenbank anlegen:

CREATE DATABASE c20;

## 1.2 Datenbank benutzen:

Ohne USE werden alle Befehle entweder ohne ausgewählte Datenbank (Fehlermeldung „No Database selected“) oder auf die falschen Datenbank ausgeführt.

USE c20;

## 1.3 Datenbank löschen

DROP DATABASE c20;

ACHTUNG! Der Befehl wird wie immer OHNE JEDE Rückfrage ausgeführt! Im dümmsten Fall ist die Datenbank ganz weg!

# 2. Tabellen anlegen / Struktur ändern und anzeigen

## 2.1 Tabelle anlegen:

CREATE TABLE datKunden (  
 nummer INT(11) NOT NULL PRIMARY KEY auto\_increment,  
 name VARCHAR(255) NOT NULL,  
 land CHAR(2) NOT NULL,  
 plz VARCHAR(10) NOT NULL,  
 ort VARCHAR(100) NOT NULL,  
 geschlecht CHAR(1) NULL,  
 umsatz DECIMAL(8,2) NOT NULL default 0.0);

## 2.2 Spalte ändern:

ALTER TABLE datKunden MODIFY geschlecht CHAR(1) NOT NULL default '?';

## 2.3 Spalte hinfügen:

ALTER TABLE datKunden ADD geburtsdatum DATE NULL default NULL;

Bei Datumswerten: Default auf NULL setzen, sonst füllt er je nach Datenbank automatisch mit   
0000-00-00!

## 2.4 Struktur der Tabelle anzeigen

DESC datKunden;

Zeigt die Struktur-Ansicht einer Tabelle. Merke: Struktur zeigt die „Spalten“ als Zeilen mit all ihren Definitionen an. Die Datenansicht wird mit SELECT angezeigt!

## 2.5 CREATE TABLE-Statement anzeigen

SHOW CREATE TABLE datKunden;

## 2.6 DROP TABLE

Eine Tabelle löschen:

DROP TABLE datKunden;

ACHTUNG! Der Befehl wird ohne jede Rückfrage ausgeführt. Bei Unachtsamkeit ist die Tabelle weg!

# 3. Daten bearbeiten

## 3.1 Datensatz einfügen, Variante mit SET:

INSERT INTO datKunden SET name='A', strasse='B';

## 3.2 Datensatz einfügen, Variante mit VALUES:

INSERT INTO datKunden (name, strasse) VALUES ('A', 'B');

## 3.3 Datensatz ändern:

UPDATE datKunden SET name='C' WHERE nummer=1;

Achtung! UPDATE immer mit WHERE nutzen, sonst werden alle Datensätze geändert!

## 3.4 Datensatz löschen:

DELETE FROM datKunden WHERE nummer=1;

Achtung! DELETE immer mit WHERE nutzen, sonst werden alle Datensätze gelöscht!

## 3.5 DELETE FROM und TRUNCATE

DELETE FROM löscht alle Datensätze einer Tabelle, wenn nicht ein WHERE dazu gesetzt wird. Dabei bleibt z.B. der letzte Wert von AUTO\_INCREMENT erhalten. Waren vorher 555 Datensätze in der Tabelle, hat der nächste Datensatz die ID 556. Will man die Tabelle quasi „auf Werkeinstellung“ zurücksetzen wird der Befehl

TRUNCATE TABLE datKunden;

verwendet. AUTO\_INCREMENT wird zurückgesetzt und auch bei den meisten Datenbank-Systemen die Adressierung/Reservierung des Speichers wieder freigegeben. Die logische Alternative wäre zunächst ein SHOW CREATE TABLE auszuführen, dann ein DROP TABLE und dann den erzeugten CREATE TABLE-Befehl neu auszuführen.

# 4. SELECT

## 4.1 Vollständiges SELECT-Statement

Merkwort: WGHOL (WHERE – GROUP BY – HAVING – ORDER BY – LIMIT)

SELECT K.name,  
 K.strasse,  
 K.land,  
 K.plz,  
 K.ort,  
 COUNT(A.nummer) AS AnzahlAuftraege  
FROM datKunden AS K  
LEFT JOIN datAuftraege AS A ON datAuftraege.fk\_kunden\_nummer=datKunden.nummer  
WHERE K.land='D'  
GROUP BY K.nummer  
HAVING AnzahlAuftraege > 5  
ORDER BY K.name DESC  
LIMIT 0,10;

## 4.2 SELECT

SELECT leitet die Liste aller Felder, Berechnungen, zeilenbasierten Funktionen und aggregierten Funktionen ein.

### 4.2.1 SELECT \*

Select wählt Datensätze aus einer oder mehreren Tabellen aus, ohne dabei Daten zu verändern. Es entsteht lediglich eine „Ansicht“ auf die Daten.

SELECT \* FROM datKunden;

Zeigt alle Spalten an aus der Tabelle datKunden.

SELECT name, vorname FROM datKunden;

Zeigt die Spalten „name“ und „vorname“ aus der Tabelle „datKunden“ an.

### 4.2.2 SELECT mit statischem Text

SELECT 'Testtext', name FROM datKunden;

Ergänzt eine Spalte mit dem statischen Text „Testtext“.

### 4.2.3 SELECT mit Rechenoperationen

SELECT umsatz \* 1.2 AS gewinnmarge;

Einfache Rechenoperationen: Zeigt die Spalte „umsatz“ multipliziert mit 1,2 an und vergibt den Alias-Namen „gewinnmarge“ an die Spalte.

### 4.2.4 SELECT mit zeilenbasierten Funktionen

SELECT UPPER(name) FROM datKunden;

Benutzung in Verbindung mit Funktionen: Zeigt die Spalte „name“ in Großbuchstaben an.

## 4.3 FROM

Adressiert die abzufragende(n) Tabelle(n). I.d.R. sollte nur eine Tabelle adressiert werden, es geht aber zum Beispiel auch:

SELECT Tabelle1.\*, Tabelle2.\* FROM Tabelle1, Tabelle2

Das bildet ein kartesisches Produkt. Die Erklärung, warum das nicht gemacht werden sollte, findet sich in der Präsentation zu JOINs.

## 4.4 JOINs

JOINs holen weitere Tabellen hinzugenommen werden. Die Datensätze werden über einen Primärschlüssel (Primary Key) in der Haupt-Tabelle und den Fremdschlüssel (Foreign Key) in der untergeordneten Tabelle zugeordnet. Es können mehrere Tabellen mit unterschiedlichen JOINs hinzugefügt werden. Der häufigste verwendete JOIN ist ein LEFT JOIN. Als „linke“ Tabelle wird immer die Tabelle aus der FROM-Klausel betrachtet, alle per JOIN zugeordneten Tabellen werden „rechts“ davon angeordnet.

Es gibt unterschiedliche Arten von JOINs. Informationen dazu finden sich in der Präsentation zu JOINs.

Die ON-Klausel funktioniert vergleichbar einer WHERE-Klausel, auch hier können Funktionen, AND, OR, etc. verwendet werden.

### 4.4.1 Subselect

SELECT   
 name, strasse, land, plz, ort  
FROM  
 datKunden  
WHERE  
 land IN (SELECT   
 code  
 FROM  
 datLaender  
 )

## 4.5 WHERE

Filtert das Ergebnis. Die Bedingungen werden mit AND oder OR verbunden. Dabei gilt „Konjunktion vor Disjunktion“. Es werden also erst alle AND-Verknüpfungen abgearbeitet und dann die OR-Verbindungen. Die Bedingungen lassen sich aber durch Klammern jeweils priorisieren, vergleichbar der Klammern in der Mathematik, wo „Punkt vor Strich“ gilt. Zulässig sind reguläre binäre Operatoren (z.B. =, <, >, <>, …)

WHERE nummer = 1

WHERE nummer > 100 AND name LIKE 'M%'

WHERE nummer > 100 AND name LIKE 'M%' OR nummer > 1000

WHERE nummer > 100 AND (name LIKE 'M%' OR nummer > 1000)

### 4.5.1 LIKE

Like führt eine Suche mit Pattern-Matching durch. In mySQL deaktiviert Like die case-sensitivity:

WHERE name LIKE 'M%'

Sucht alle Namen, die mit „M“ oder „m“ beginnen.

WHERE name LIKE 'M\_nn';

Sucht alle Namen bei denen der einzelne Buchstabe zwischen M und nn irgendeinem Buchstaben entspricht.

### 4.5.2 BETWEEN

Filtert nach einem Werte-Bereich:

WHERE nummer BETWEEN 5 AND 10

Die Randwerte (5 und 10 in diesem Beispiel) werden in das Ergebnis inkludiert!

### 4.5.3 IN

Definiert eine Werteliste nach der gefiltert wird:

WHERE nummer IN (1,3,9)

### 4.5.4 NOT

Alle Operationen lassen sich negieren mit NOT:

WHERE nummer NOT IN (1,3,9)

## 4.6 Nicht-Aggregierte Funktionen (Single-Row-Funktionen) in SELECT

Damit sind Funktionen gemeint, die sich auf einzelne Datensätze beziehen und nicht wie z.B. SUM auf mehrere Datensätze die zusammengefasst (aggregiert) werden.

### 4.6.1 UPPER / LOWER

Formatiert den Text in Groß-, bzw Kleinbuchstaben:

SELECT UPPER(name) FROM datKunden;

SELECT LOWER(name) FROM datKunden;

### 4.6.2 CONCAT

Verbindet statische String-Werte bzw. Felder miteinander. Z.B.:

SELECT CONCAT(vorname, ' ', nachname) AS name FROM datMitarbeiter;

Ergibt ein Feld „name“ das aus Vorname und Nachname besteht. Alternativ z.B.:

SELECT CONCAT('Herr ', nachname) FROM datMitarbeiter;

Das lässt sich auch kombinieren mit z.B. der IF-Anweisung:

SELECT CONCAT(IF(geschlecht = 'M', 'Herr ', 'Frau '), nachname FROM datMitarbeiter;

### 4.6.3 CONCAT\_WS

Oft braucht man bestimmte Listen auch Komma- oder Semikolon-getrennte Listen aus unterschiedlichen Feldern, um z.B. eine CSV-Datei direkt im Statement vorzubereiten:

CONCAT\_WS(';', nachname, vorname, geschlecht) FROM datMitarbeiter;

Der erste Parameter ';' gibt dabei das Trennzeichen an, das Ergebnis wäre:

Brinkmann;Matthias;M

Der Befehl „GROUP\_CONCAT“ kann ein ähnliches Ergebnis erreichen, bezieht sich aber nicht auf einzelne Datensätze („Single-Row“) sondern auf eine aggregierte Funktion („GROUP BY“) und wird später erklärt.

### 4.6.4 MOD

Modulo gibt den Rest einer Division zurück und wird häufig verwendet, um zu ermitteln, ob eine Zahl gerade oder ungerade ist:

SELECT MOD(nummer, 2) AS gerade FROM datMitarbeiter;

Gibt 0 bei geraden Zahlen und einen anderen Wert bei ungeraden Zahlen zurück.

### 4.6.5 LEFT / RIGHT

Gibt die äußert linken oder äußerst rechten Werte einer Spalte oder eines Wertes zurück:

SELECT LEFT(‚abcdefg‘, 4);

Ergibt abcd. Wird häufig gebraucht, wenn in Schnittstellen das Empfänger-System nur eine bestimmte Länge an Zeichen einer Spalte zulässt.

## 4.7 Filtern mit Datums- und Zeitwerten

Zu unterscheiden sind DATE und DATETIME im Gegensatz zu Unix-Timestamps. Timestamps geben die Anzahl an Sekunden seit dem 01.01.1970 zurück. Timestamps können für Werte vor dem 01.01.1970 auch negativ sein. Dadurch lässt sich mit Timestamps rechnen, als wären sie normale Integer-Werte.

### 4.7.1 Umwandeln eines Timestamps in DATETIME

Um Timestamps normal lesbar zu machen:

SELECT FROM\_UNIXTIME(meinzeitstempel) AS meindatetime FROM Tabelle;

Ergibt aus einem Timestamp einen lesbaren Datum- und Zeitwert.

### 4.7.2 Einen Wert zwischen zwei Datumswerten ermitteln

Um z.B. alle Mitarbeiter zu ermitteln, die zwischen dem 01.03.1981 und 05.05.1985 Gebburtstag hatten:

SELECT name, vorname   
FROM datMitarbeiter   
WHERE geburtsdatum BETWEEN '1981-03-01' AND '1985-05-05');

Die Formatierung ist bei den meisten RDBMS YYYY-MM-DD, also 4-stelliges Jahr, 2-stelliger Monat und 2-stelliger Tag, i.d.R. funktioniert aber auch: 1985-5-5.

Die Datumswerte VON und BIS gehören dabei ebenso wie die Werte in einem BETWEEN bei INTEGER-Werten zur Ergebnis-Menge dazu!

### 4.7.3 Kleiner, Größer, Gleich mit Datumswerten

Datumswerte lassen sich wie Integer-Werte mit „>“, „<“ und „=“ filtern:

SELECT name, vorname FROM datMitarbeiter WHERE geburtsdatum > '1981-23-05';

SELECT name, vorname FROM datMitarbeiter WHERE geburtsdatum < '1981-23-05';

SELECT name, vorname FROM datMitarbeiter WHERE geburtsdatum = '1981-23-05';

### 4.7.4 Umwandeln/Formatieren von Datumswerten

Datumswerte lassen sich individuell formatieren:

DATE\_FORMAT(<datum>, '<FormatString>');

Z.B.

DATE\_FORMAT(CURDATE(), '%d.%m.%Y');

Die nachfolgende Liste enthält ALLE Möglichkeiten, relevant für die Prüfung sind nur %d, %m, %Y (zumindest nach den Erfahrungen der vergangenen Prüfungen, Überraschungen sind immer möglich).

|  |  |
| --- | --- |
| Format | Beschreibung |
| %a | Abgekürzter Wochentag (So-Sa) |
| %b | Abgekürzter Name des Monats (Jan-Dez) |
| %c | Monat, numerisch (0-12) |
| %D | Tag des Monats mit englischen Suffix (0., 1., 2., 3.,) |
| %d | Tag des Monats, numerisch (00-31) |
| %e | Tag des Monats, numerisch (0-31) |
| %f | Mikrosekunden (000000-999999) |
| %H | Stunden (00-23) |
| %h | Stunden (01-12) |
| %I | Stunden (01-12) |
| %i | Minuten, numerisch (00-59) |
| %j | Tag des Jahres (001-366) |
| %k | Stunde (0-23) |
| %l | Stunde (1-12) |
| %M | Monatsname (Januar bis Dezember) |
| %m | Monat, numerisch (00-12) |
| %p | Vormittags oder nachmittags |
| %r | Zeit, 12-Stunden (hh: mm: ss gefolgt von AM oder PM) |
| %S | Sekunden (00-59) |
| %s | Sekunden (00-59) |
| %T | Zeit, 24 Stunden (hh: mm: ss) |
| %U | Woche (00-53), wobei der Sonntag der erste Tag der Woche ist |
| %u | Woche (00-53), wobei Montag der erste Tag der Woche ist |
| %V | Woche (01-53), wobei der Sonntag der erste Tag der Woche ist, verwendet mit X% |
| %v | Woche (01-53), wobei Montag der erste Tag der Woche, mit% verwendet x |
| %W | Wochentag (Sonntag bis Samstag) |
| %w | Tag der Woche (0 = Sonntag, 6 = Samstag) |
| %X | Jahr für die Woche, wo Sonntag, den ersten Tag der Woche, vier Ziffern, mit% V verwendet |
| %x | Jahr für die Woche, wo Montag der erste Tag der Woche, vier Ziffern, mit% v verwendet |
| %Y | Jahr, numerisch, vierstellig |
| %y | Jahr, numerisch, zweistellig |

### 4.7.5 DATEDIFF

Ermittelt in Tagen den Abstand zwischen zwei gegebenen Datums-Werten (positiv/negativ):

DATEDIFF(CURDATE(), '2022-12-31');

Gibt z.B. zurück, wie viele Tage das Jahr noch hat.

### 4.7.5 TO\_DAYS

Konvertiert ein Datum in die Tage ab Beginn der Zeitrechnung (Nicht wie bei Timestamps ab 01.01.1970!):

TO\_DAYS(CURDATE());

# 5. SELECTS miteinander verbinden

Die Ergebnisse mehrerer SELECT-Statements lassen sich miteinander verbinden. Das ist z.B. dann nötig, wenn aus der Tabelle datKunden und der Tabelle datLieferanten Datensätze gemeinsam angezeigt werden sollen:

## 5.1 UNION

Union verbindet die Ergebnisse von zwei SELECTs oder auch mehreren:

SELECT \* FROM datKunden  
UNION  
SELECT \* FROM datLieferanten;

WICHTIG: Nach den SELECTs wird kein Semikolon gesetzt, weil beide SELECT-Statements per UNION verbunden werden und ein GESAMTES Ergebnis ergeben! Erst zum Ende des verbundenen Ergebnisses wird das Semikolon gesetzt! Das Ergebnis sind alle Datensätze aus datKunden und darunter alle Datensätze aus datLieferanten.

# 6. Datentypen

**Inhalt folgt! Muss wahrscheinlich vorgezogen werden.**

# 7. Daten kombinieren

## 7.1 UNION

Kombiniert die Daten zweier oder mehrerer Abfragen:

SELECT \* FROM datKunden  
UNION  
SELECT \* FROM datLieferanten

Um nicht automatisch ein DISTINCT-Ergebnis (Eindeutige Datensätze) zu erhalten:

UNION ALL

## 7.2 INTERSECT

Bildet die Schnittmenge zweier oder mehrerer Abfragen:

SELECT \* FROM datKunden  
INTERSECT  
SELECT \* FROM datLieferanten

## 7.3 MINUS

Zieht das Ergebnis der einen Abfrage von der anderen Abfrage ab:

SELECT \* FROM datKunden  
MINUS  
SELECT \* FROM datLieferanten

# 8. Transaktions-Steuerung

**Inhalt folgt!**

# 9. CONSTRAINTS

## 9.1 Anlegen eines FOREIGN-KEY-Constraints:

ALTER TABLE datAuftraege   
ADD CONSTRAINT FK\_AuftraegeKunden   
FOREIGN KEY (fk\_kunden\_nummer)   
REFERENCES datKunden(nummer);

## 9.2 Löschen eines FOREIGN-KEY-Constraints

ALTER TABLE datAuftraege  
DROP FOREIGN KEY FK\_AuftraegeKunden;

# 10. Berechtigungen (GRANT)

**Inhalt folgt!**

# Fehlend:

**Erklärung DISTINCT, GROUP BY, Strukturdarstellung, Strukturierte/Unstrukturierte Daten, Aggregation, JOINing**

# Erweiterndes Wissen:

* Deuten von Fehlermeldungen

# Anlage 1 Stichwort-Verzeichnis

C

CONCAT 7

CONCAT\_WS 7

F

FROM\_UNIXTIME 8

I

IN 7

J

JOIN 6

L

LIKE 6

LOWER 7

M

MOD 7

N

NOT 7

S

SELECT 4, 5

U

UNION 9

UPPER 7

W

WHERE 6

# Anlage 2 Lexikon