

Datenbanksysteme

SQL DQL

Jan Haase

2024

Abschnitt 6

Themenübersicht

- Warum Datenbanken?
- Grundbegriffe und Datenbankentwurf
- Entity-Relationship-Modelle
- Relationales Datenbankmodell
- Normalisierung

 **Arbeiten mit relationalen Datenbanken (SQL)**

SQL: Überblick

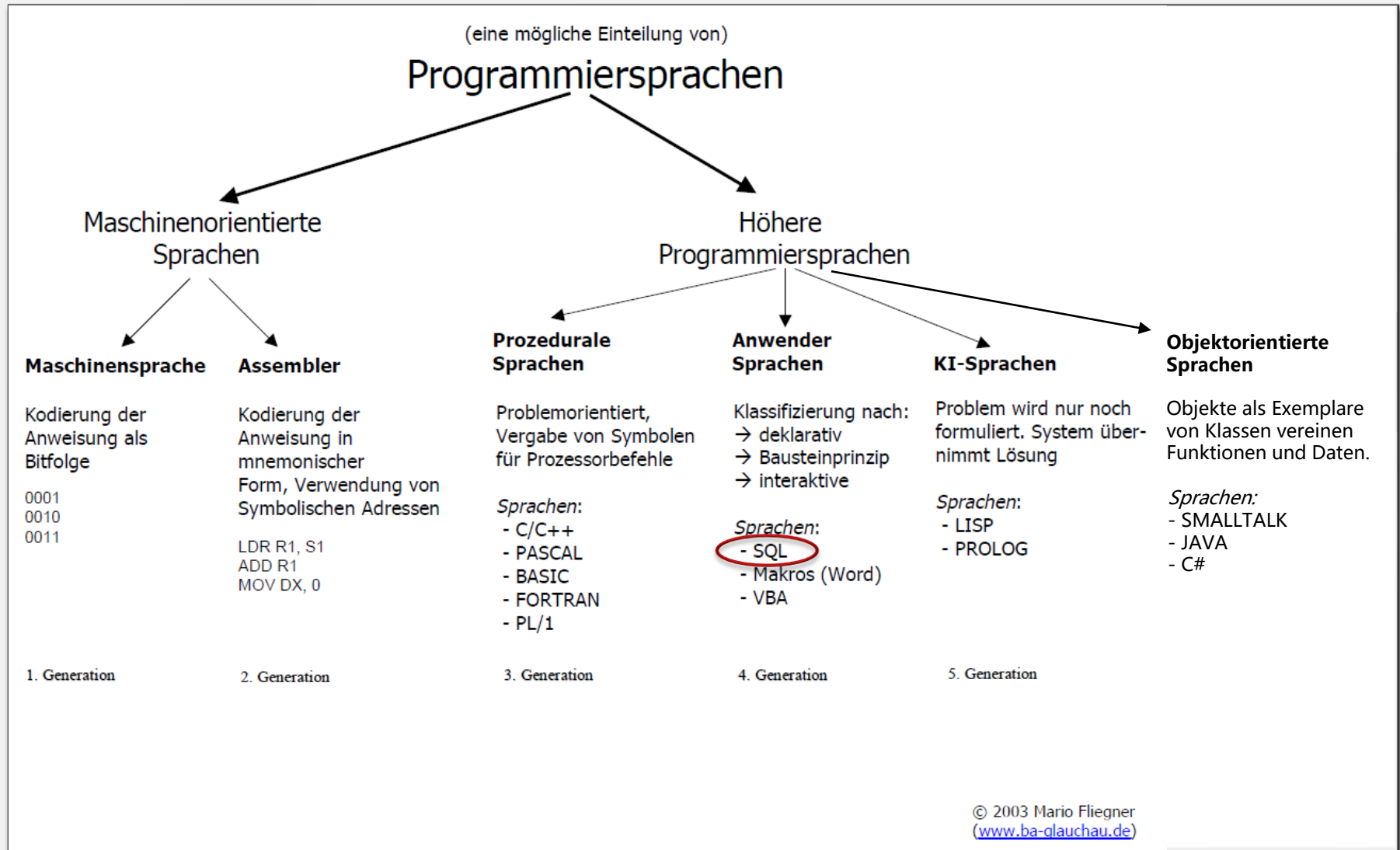
- SQL steht für Structured Query Language
- Entwickelt seit den 1970er Jahren
- Orientiert an Relationentheorie, aber Umsetzung nicht konsequent

- Als ISO-Standard definiert:
 - Wichtige Meilensteine:
 - 1986-89 (SQL1)
 - 1992 (SQL2 bzw. SQL-92)
 - 1999 (SQL3 bzw. SQL:1999)
 - Aktuell: SQL:2011 bzw. ISO/IEC 9075:2011
 - Sehr umfangreich
 - Kernbereich: Teil 1 Framework, Teil 2 Foundation

- Sehr hohe Verbreitung in der Praxis für relationale Datenbanken
- Umsetzung in konkreten DBMS aber oft mit Abweichungen vom Standard, z. B. bei Datentypen



SQL: Einordnung



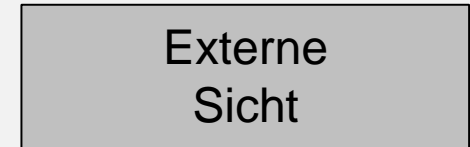
Kategorien von SQL-Befehlen

- **DQL (Data Query Language)**
Abfrage und Zusammenstellung von Daten

- SELECT

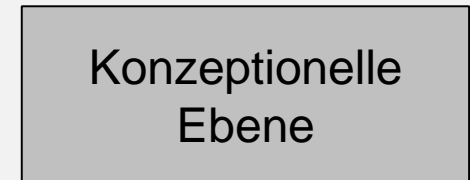
- **DML (Data Manipulation Language)**
Umgang mit Tabelleninhalten

- INSERT
- UPDATE
- DELETE

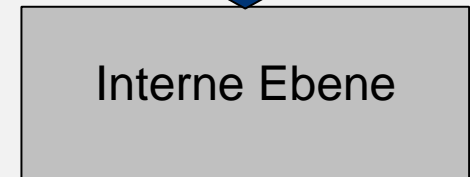


- **DDL (Data Definition Language)**
Erstellen und Ändern von Datenbanken und Tabellen

- CREATE
- ALTER
- DROP



- **DAL (Data Administration Language)**
 - TCL (Transaction Control Language)
 - DCL (Data Control Language)



Themenübersicht

- Warum Datenbanken?
- Grundbegriffe und Datenbankentwurf
- Entity-Relationship-Modelle
- Relationales Datenbankmodell
- Normalisierung

 **Arbeiten mit relationalen Datenbanken (SQL)**

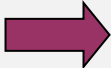
Themenübersicht

- Grundbegriffe und Datenbankentwurf
- Entity-Relationship-Modelle
- Relationales Datenbankmodell
- Normalisierung
- Arbeiten mit relationalen Datenbanken (SQL)

DQL (Data Query Language)

- DML
- DDL
- DAL

Themenübersicht

- Grundbegriffe und Datenbankentwurf
- Entity-Relationship-Modelle
- Relationales Datenbankmodell
- Normalisierung
- Arbeiten mit relationalen Datenbanken (SQL)
 - **DQL (Data Query Language)**
 -  **Einfache SQL-Abfragen**
 - NULL-Werte
 - Unterabfragen
 - JOINS
 - Group BY
 - DML
 - DDL
 - DAL

Grundstruktur SQL-DQL-Anweisungen

- **Befehl:** SELECT
- **Feldselektion:** „*“ oder Aufzählung der Attribute
- **Tabellenselektion:** FROM mit Tabellennamen
Verknüpfung mehrerer Tabellen durch kartesisches Produkt/JOIN
- optional:
 - **Klauseln:** WHERE mit Operatoren
(=, <>, <, >, <=, >=, LIKE, BETWEEN, IN)
 - **Verknüpfung der Klauseln:**
Logische Operatoren AND, OR und NOT
Klammernsetzung ist möglich
 - **Sortieren der Ergebnisse:** ORDER BY
Richtung mit ASC - aufsteigend und DESC - absteigend

SELECT Allgemeine Form

```
SELECT [ALL | DISTINCT] {spalten | *}
FROM tabelle [alias] [tabelle [alias]] ...
[WHERE {bedingung | unterabfrage}]
[GROUP BY spalten [HAVING {bedingung | unterabfrage}]]
[ORDER BY spalten [ASC | DESC]...];
```

Klausel	Erläuterung
SELECT [DISTINCT]	Wähle die Werte aus der/den Spalte(n) [mehrfache Datensätze nur einmal] ...
FROM	... aus der Tabelle bzw. den Tabellen ...
WHERE	... wobei die Bedingung(en) erfüllt sein soll(en) ...
GROUP BY	... und gruppiere die Ausgabe von allen Zeilen mit gleichem Attributwert zu einer einzigen ...
HAVING	... wobei darin folgende zusätzliche Bedingung(en) gelten müssen/muss ...
ORDER BY [ASC/DESC]	... und sortiere nach den Spalten [auf- bzw. absteigend] .

Hinweis: Oracle-Befehle können groß/klein geschrieben werden.

Vergleichstexte sind „case sensitive“

<http://www.tinohempel.de/info/info/datenbank/sql.htm>

SELECT Beispiel 1

Gegeben sei folgende Tabelle

Bestellung

BESTELLNUMMER	EINKAEUFER	DATUM	KUNDENNUMMER
1001	John	01.01.14	1
1011	Bill	01.02.14	2
1101	Joe	01.03.14	3
1231	Paul	03.07.15	4
1232	Paul	04.07.15	4

select EINKAEUFER from BESTELLUNG;
select BESTELLUNG.EINKAEUFER from BESTELLUNG;
select B.EINKAEUFER from BESTELLUNG B;

select distinct EINKAEUFER
from BESTELLUNG;

select BESTELLNUMMER, EINKAEUFER
from BESTELLUNG;

EINKAEUFER
John
Bill
Joe
Paul
Paul

EINKAEUFER
Bill
Joe
John
Paul

BESTELLNUMMER	EINKAEUFER
1001	John
1011	Bill
1101	Joe
1231	Paul
1232	Paul

SELECT Beispiel 2

Rechnung

RECHNUNGSNUMMER	KUNDE	SUMME
1	Müller	434
2	Meier	5116
3	Woitila	648
4	Eastwood-LA	987
5	Eastwood-NY	6765

Bestellung

BESTELLNUMMER	EINKAEUFER	DATUM	KUNDENNUMMER
1001	John	01.01.14	1
1011	Bill	01.02.14	2
1101	Joe	01.03.14	3
1231	Paul	03.07.15	4
1232	Paul	04.07.15	4

select * from BESTELLUNG;

BESTELLNUMMER	EINKAEUFER	DATUM	KUNDENNUMMER
1001	John	01.01.14	1
1011	Bill	01.02.14	2
1101	Joe	01.03.14	3
1231	Paul	03.07.15	4
1232	Paul	04.07.15	4

**select RECHNUNGSNUMMER,
SUMME, SUMME*0.19
from RECHNUNG;**

RECHNUNGSNUMMER	SUMME	SUMME*0.19
1	434	82,46
2	5116	972,04
3	648	123,12
4	987	187,53
5	6765	1285,35

SELECT Beispiel 3

Rechnung

RECHNUNGSNUMMER	KUNDE	SUMME
1	Müller	434
2	Meier	5116
3	Woitila	648
4	Eastwood-LA	987
5	Eastwood-NY	6765

Bestellung

BESTELLNUMMER	EINKAEUFER	DATUM	KUNDENNUMMER
1001	John	01.01.14	1
1011	Bill	01.02.14	2
1101	Joe	01.03.14	3
1231	Paul	03.07.15	4
1232	Paul	04.07.15	4

**select RECHNUNGSNUMMER,
SUMME, SUMME*0.19 MWST
from RECHNUNG;**

RECHNUNGSNUMMER	SUMME	MWST
1	434	82,46
2	5116	972,04
3	648	123,12
4	987	187,53
5	6765	1285,35

**Select 'BESTNR: ' || BESTELLNUMMER || ' - ' ||
DATUM, 2017 JAHR
from BESTELLUNG;**

'BESTNR: ' BESTELLNUMMER ' - '	DATUM	JAHR
BestNr: 1001 -	01.01.14	2017
BestNr: 1011 -	01.02.14	2017
BestNr: 1101 -	01.03.14	2017
BestNr: 1231 -	03.07.15	2017
BestNr: 1232 -	04.07.15	2017

SELECT Beispiel 4

Rechnung

RECHNUNGSNUMMER	KUNDE	SUMME
1	Müller	434
2	Meier	5116
3	Woitila	648
4	Eastwood-LA	987
5	Eastwood-NY	6765

Bestellung

Kunde

BESTELLNUMMER	EINKAEUFER	DATUM	KUNDENNUMMER
1001	John	01.01.14	1
1011	Bill	01.02.14	2
1101	Joe	01.03.14	3
1231	Paul	03.07.15	4
1232	Paul	04.07.15	4

**select KUNDE from RECHNUNG
order by KUNDE asc;**

**select KUNDE from RECHNUNG
order by KUNDE desc;**

**select EINKAEUFER, BESTELLNUMMER
From BESTELLUNG
order by EINKAEUFER asc,
BESTELLNUMMER desc;**

KUNDE
Eastwood-LA
Eastwood-NY
Meier
Müller
Woitila

KUNDE
Woitila
Müller
Meier
Eastwood-NY
Eastwood-LA

SELECT mit Auswahlkriterium 1

Rechnung

RECHNUNGSNUMMER	KUNDE	SUMME
1	Müller	434
2	Meier	5116
3	Woitila	648
4	Eastwood-LA	987
5	Eastwood-NY	6765

**select KUNDE from RECHNUNG
where RECHNUNGSNUMMER in (1,3,5);**

KUNDE
Müller
Woitila
Eastwood-NY

**select KUNDE from RECHNUNG
where RECHNUNGSNUMMER between 1 and 3;**

KUNDE
Müller
Meier
Woitila

**select BESTELLNUMMER, EINKAEUFER
from BESTELLUNG
where BESTELLNUMMER = 1101;**

BESTELLNUMMER	EINKAEUFER
1101	Joe

SELECT mit Auswahlkriterium 2

Bestellung

BESTELLNUMMER	EINKAEUFER	DATUM	KUNDENNUMMER
1001	John	01.01.14	1
1011	Bill	01.02.14	2
1101	Joe	01.03.14	3
1231	Paul	03.07.15	4
1232	Paul	04.07.15	4

**select EINKAEUFER from BESTELLUNG
where BESTELLNUMMER > 1000 and
BESTELLNUMMER < 1100;**

EINKAEUFER
John
Bill

**select EINKAEUFER from BESTELLUNG
where EINKAEUFER like '%o%';**

EINKAEUFER
John
Joe

**select distinct EINKAEUFER from BESTELLUNG
where EINKAEUFER like '%l';**

EINKAEUFER
Bill
Paul

SELECT mit Auswahlkriterium 3

Bestellung

BESTELLNUMMER	EINKAEUFER	DATUM	KUNDENNUMMER
1001	John	01.01.14	1
1011	Bill	01.02.14	2
1101	Joe	01.03.14	3
1231	Paul	03.07.15	4
1232	Paul	04.07.15	4

**select EINKAEUFER from BESTELLUNG
where EINKAEUFER like '___l%';**

EINKAEUFER
Bill

**select EINKAEUFER from BESTELLUNG
where not EINKAEUFER = 'Paul';**

EINKAEUFER
John
Bill
Joe

**Select Einkaeufer from Bestellung
where Einkaeufer <> 'Paul';**

EINKAEUFER
John
Bill
Joe

SELECT mit Auswahlkriterium 4

BESTELLNUMMER	EINKAEUFER	DATUM	KUNDENNUMMER
1001	John	01.01.14	1
1011	Bill	01.02.14	2
1101	Joe	01.03.14	3
1231	Paul	03.07.15	4
1232	Paul	04.07.15	4

**select EINKAEUFER from BESTELLUNG
where EINKAEUFER = 'paul';**

EINKAEUFER

**select EINKAEUFER from BESTELLUNG
where lower(EINKAEUFER) = 'paul';**

EINKAEUFER
Paul
Paul

**select upper(EINKAEUFER) from BESTELLUNG
where lower(EINKAEUFER) = 'paul';**

UPPER(EINKAEUFER)
PAUL
PAUL

Themenübersicht

- Grundbegriffe und Datenbankentwurf
- Entity-Relationship-Modelle
- Relationales Datenbankmodell
- Normalisierung
- Arbeiten mit relationalen Datenbanken (SQL)
 - **DQL (Data Query Language)**
 - Einfache SQL-Abfragen
 - ➡ **NULL-Werte**
 - Unterabfragen
 - JOINS
 - Group BY
 - DML
 - DDL
 - DAL

Fehlende Informationen

- Problemstellung:
Oftmals sollen in einer Datenbank Informationen gespeichert werden, obwohl einzelne Informationen unbekannt sind. Zum Beispiel ist in einer Produkt-Datenbank das Gewicht eines Produktes unbekannt. Wie speichern wir diese Information nun in unserer Datenbank?
- Ideen:
 - Spezieller Wert des jeweiligen Datentyps, z.B. 0.
 - Gefährlich, da keine Abgrenzung zwischen Gewicht = unbekannt und Gewicht = 0 (z.B. Software) mehr möglich ist.
 - Extremer Wert des jeweiligen Datentyps, z.B. -1
 - Ok, für kleinere Datenbanken
 - Problematisch in größeren Systemen
 - Schwierigkeiten bei der Zusammenführung
 - Wartbarkeit und Dokumentation

Fehlende Informationen: NULL Werte

- **Lösungsansatz in SQL: NULL**

- Hat keinen Wert
- Wahrheitswert: unbekannt

Achtung!
Gewicht = NULL
wäre fehlerhaft!

- **Abfrage in SQL**

```
SELECT * FROM Produkt WHERE Gewicht IS NULL;
```

oder

```
SELECT * FROM Produkt WHERE Gewicht IS NOT NULL;
```

- **Probleme**

```
SELECT * FROM Produkt  
WHERE Gewicht<=10 OR Gewicht>10;
```

- Die klassische Logik sagt uns, dass hierbei alle Datensätze angezeigt werden müssten, aber: wenn ein Datensatz bei Gewicht den Wert NULL enthält, wird dieser nicht mit ausgegeben!

Zwei-wertige Logik

- Bereits bekannte, zwei-wertige Logik

X	Y	NOT X	X AND Y	X OR Y
w	w	f	w	w
w	f	f	f	w
f	w	w	f	w
f	f	w	f	f

Drei-wertige Logik

- Umgang mit NULL-Werten erfordert drei-wertige Logik.

X	Y	NOT X	X AND Y	X OR Y
w	w	f	w	w
w	f	f	f	w
w	u	f	u	w
f	w	w	f	w
f	f	w	f	f
f	u	w	f	u
u	w	u	u	w
u	f	u	f	u
u	u	u	u	u

Zusammenfassung

- SQL
 - Überblick
 - Kategorien von SQL-Befehlen
 - DQL
 - DML
 - DDL
 - DAL
 - DQL
 - Grundstruktur SQL-DQL-Anweisung
 - Aggregatfunktionen
 - NULL-Werte