Inhalt

[Begriffe(Kurzerklärungen) 5](#_Toc421984861)

[Modellierung 7](#_Toc421984862)

[Regeln 7](#_Toc421984863)

[Martin Notation 7](#_Toc421984865)

[Martin Notation Beispiel 7](#_Toc421984866)

[UML Diagramme 7](#_Toc421984867)

[UML-Beispiel 8](#_Toc421984868)

[Generalisierung/Spezialisierung: 8](#_Toc421984869)

[Das Relationenmodell 9](#_Toc421984870)

[Vom ERD zum relationalen ERD: 10](#_Toc421984871)

[Normalformen 10](#_Toc421984872)

[Erste Normalform 10](#_Toc421984873)

[Zweite Normalform 10](#_Toc421984874)

[Dritte Normalform 11](#_Toc421984875)

[Access 12](#_Toc421984876)

[Abfragen mit Access Beispiele 12](#_Toc421984877)

[Access Abfragen mit Joins 13](#_Toc421984878)

[Cross Join 13](#_Toc421984879)

[Inner Join 13](#_Toc421984880)

[Outer Join 13](#_Toc421984881)

[ACID 15](#_Toc421984882)

[SQL-Queries 15](#_Toc421984883)

[Integrity Constraints 15](#_Toc421984884)

[Kommentare 16](#_Toc421984885)

[Datenbanken 16](#_Toc421984886)

[DDL 16](#_Toc421984887)

[CREATE DATABASE 16](#_Toc421984888)

[CREATE TABLE 17](#_Toc421984889)

[CREATE INDEX 18](#_Toc421984890)

[Domänen/Defaults 19](#_Toc421984894)

[CREATE VIEW 19](#_Toc421984895)

[Trigger 20](#_Toc421984897)

[DML 22](#_Toc421984900)

[INSERT INTO 22](#_Toc421984901)

[UPDATE 23](#_Toc421984902)

[DELETE 23](#_Toc421984903)

[DCL 24](#_Toc421984904)

[CREATE LOGIN 24](#_Toc421984905)

[CREATE USER 25](#_Toc421984906)

[CREATE ROLE 25](#_Toc421984907)

[GRANT/REVOKE 25](#_Toc421984908)

[DQL 26](#_Toc421984909)

[SELECT 26](#_Toc421984910)

[SELECT JOIN mit WHERE 26](#_Toc421984911)

[SELECT JOIN mit JOIN 26](#_Toc421984912)

[Sub-Queries 27](#_Toc421984913)

[Transaktionen 27](#_Toc421984914)

[Programmierung mit SQL 28](#_Toc421984915)

[FUNCTIONS 31](#_Toc421984916)

[PROCEDURES 32](#_Toc421984917)

[Cursor 33](#_Toc421984918)

[Systemkatalog 34](#_Toc421984919)

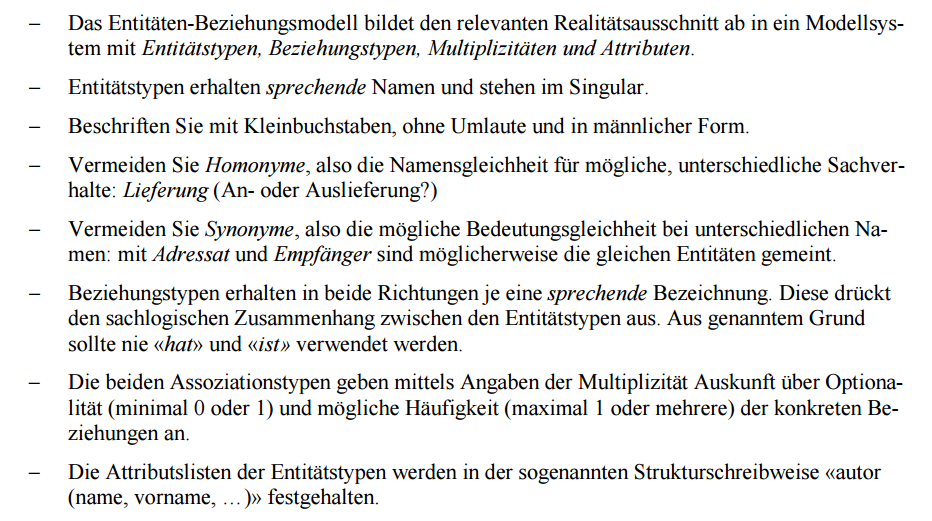
# Begriffe(Kurzerklärungen)

|  |  |
| --- | --- |
| Begriff | Erklärung |
| Mutationsanomalie | Mehrere Änderungen für einen Entität |
| Einfügeanomalie |  |
| Löschanomalie |  |
| Metadaten | Metadaten der Metainformationen sind Daten, die Informationen über Merkmale anderer Daten enthalten, aber nicht diese Daten selbst. |
| Entitätstypen | z.B. Buch, Autor, Verlag |
| Entitätsmenge | z.B. Alle Bücher |
| Assoziation | z.B Autor schreibt Buch |
| Multiplizität | z.B. Ein Autor kann kein oder mehrere Bücher schreiben |
| Domäne | Bestimmt den Typ und den Wertebereich eines Datenbankfeldes |
| Tupelintegrität | Jeder Tupel(Datensatz) ist durch einen Primärschlüssel eindeutig erkennbar. |
| Referentielle Integrität | Jede Referenz auf eine andere Datenbank ist mit einem eindeutigen Fremdschlüssel (PK zu FK) verknüpft und somit eindeutig zu zuordnen. |
| Index | Ein Index besteht aus einer Ansammlung von Zeigern (Verweisen), die eine Ordnungsrelation auf eine oder mehrere Spalten in einer Tabelle definieren. |
| Cross Join | Beim Cross Join werden sämtliche mögliche Anordnungen zweier Tabellen dargestellt. Die Tabellen müssen dazu nicht verknüpft sein. |
| Inner Join | Beim Inner Join wird zuerst der Cross Join ausgeführt ,dann wird anhand der Verknüpfung (PK zu FK) die Zusammenhängenden werte gefiltert und angezeigt |
| Outer Join | Beim Outer Join werden zuerst der Cross und der Inner Join ausgeführt. Danach werden von der linken (Ersten) Tabelle sämtliche Werte zusätzlich ausgegeben, auch wenn sie nicht verknüpft(PK zu FK) sind |
| Null | Null entspricht nicht dem nummerischen Wert 0 und auch nicht einem leeren String. Null ist einfach gar nichts (Null Objekt, Leere Referenz) |
| Erste Normalform | Die Wertebereiche aller Attribute sind Atomar ( Feld beinhaltet nur einen Wert) |
| Zweite Normalform | Ist erfüllt wenn sie in der 1 Normalform ist, alle Primärschlüssel eindeutig sind und alle Attribute abhängig von ihren Primärschlüsseln sind. |
| Dritte Normalform | Ist erfüllt, wenn sie in der 2 Normalform ist, und kein Nichtschlüsselattribut Transitiv voneinander abhängig ist. C ist abhängig von B und B ist abhängig von A dann kann ich von A auf C schliessen. |
| DCL (Data Controll Language) | Definition von Zugriffsrechten und Speicherstrukturen z .B. CREATE LOGIN |
| DDL ( Data Definition Language) | Definition von Datenobjekten z.B. ALTER TABLE |
| DML (Data Manipulation Language) | Anlegen, Ändern und Löschen von Daten z.B: DELTE FROM |
| DQL ( Data Query Language) | Abfrage von Daten z.B. SELECT \* FROM |
|  |  |
|  |  |

Datenmanagement (INM22) EDD

# Modellierung

## Regeln



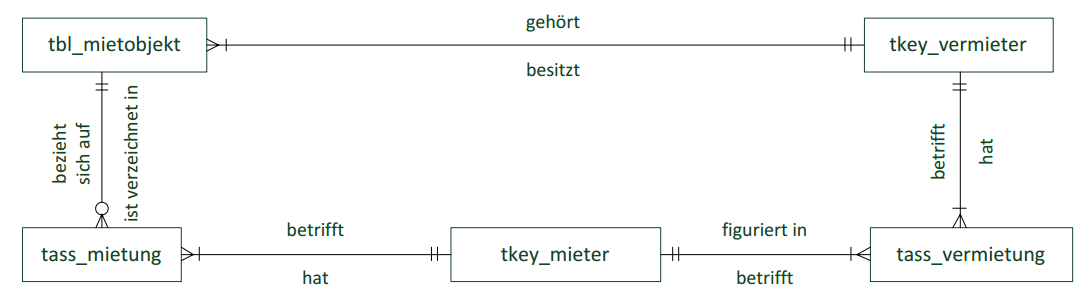
Beispiel einer Attributliste, die zu beiden Diagrammtypen erstellt werden muss. Wichtig: Primärschlüssel immer unterstreichen und Fremdschlüssel immer Umkreisen!

## 

## Martin Notation



### Martin Notation Beispiel



## UML Diagramme

Basiselemente:

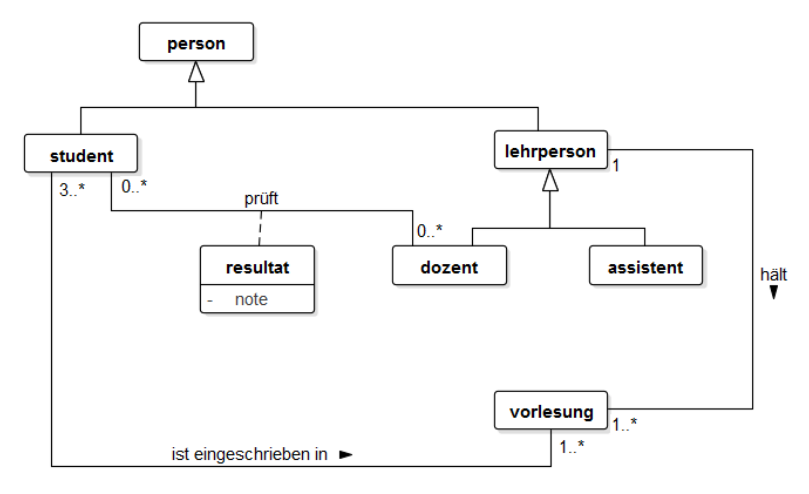
Computergenerierter Alternativtext:


Assoziazionsklassen:

Computergenerierter Alternativtext:
kunde 
kauft 
kauf 
anzahl 
artikel 
kunde kauft anzahl artikel 
Abbildung 17: Assoziationsklasse im UML-Klassendiagramm 

Computergenerierter Alternativtext:
Des Weiteren besteht die Möglichkeit, bei den Assoziationsenden die Rolle der Klasse 
zu spezifizieren. 
+Leiter 
1 
angstellter 
+Mitarbeiter 
leitet 
beschäft 
abteilung 
1 

### UML-Beispiel



## Generalisierung/Spezialisierung:

Computergenerierter Alternativtext:
kunde 
person 
(Generalisierungsmenge: Rolle der Person) 
angestellter 
vollständig und disjunkt? 
lieferant 

Vollständig = Der Supertyp hat keine eigenen Elemente, also jedes Element einem Subtyp entspricht

Disjunkt = Ein Kunde kann kein Angestellter oder Lieferant sein

Computergenerierter Alternativtext:
Arten von Spezialisierungen 
x 
disju nkte Spezialisierungen 
(Partitionierung) 
x 
z 
Superklasse 
Subklassen 
überlappende 
Spezialisierungen 
x 
z 
z 
vollständig 
überlappend (complete, overlapping) 
z 
vollständig, disjunkt(comp/ete, disjoint) 
z 
pattiell, disjunkt (incomplete, disjoint) 
partiell, überlappend (incomplete, overlapping) 

Aggregation /Komposition:

Aggregation

Computergenerierter Alternativtext:
mitarbeiter 
team 
freiberufler 

Komposition = Existenzielle Abhängigkeit z.B. Raum/Zimmer existiert nicht ohne Gebäude

Computergenerierter Alternativtext:
hand 
5 
finger 

# Das Relationenmodell

Computergenerierter Alternativtext:
4.2.1 Relationen 
Jeder Entitätstyp wird zu einer Tabelle. Jeder Beziehungstyp wird (vorerst) zu einer Ta- 
belle. 
Die Theorie bezeichnet Tabellen als Relationen. Diese Bezeichnung ist glücklicherweise 
wenig gebräuchlich, da sie mit Relationship (Beziehung) verwechselt werden kann. 
Für jede Tabelle gilt: 
eindeutiger Tabellenname 
wir übernehmen die Benennungen d ERDs; Beispiel: kunde 
bei der DB-Implementierung werden wir uns an ein Benennungskonzept halten 
und der Bezeichnung ein Präfix voranhängen; Beispiel: tbl kunde 
jede Entität - z.B. jeder Einzelkunde - ist darin eine Zeile 
die Zeilen heissen Tupel (alternativ: Datensätze, Records) 
die Reihenfolge der Tupel ist bedeutungslos, sie sind also z.B. nie sortiert 
jedes Attribut - z.B. Wohnort aller Kunden - ist darin eine Spalte 
ein Entitätstyp hat mindestens zwei Attribute, eine Tabelle also mindestens 
zwei Spalten 
Attribute erhalten eine in der Tabelle eindeutige Benennung 
sofern wir das ERD attribuiert haben, übernehmen wir die Benennungen des 
ERDs 
es gibt natürliche und künstliche Attribute: vorname ist natürlich, kun— 
den nummer ist künstlich 
in jedem Koordinatenpunkt steht ein Feld 
Felder enthalten stets einen und nur einen Attributswert oder das Feld ist leer 
(genannt: NULL-Wert oder NULL-Marke) 

Primärschlüssel immer unterstreichen

Fremdschlüssel immer umkreisen

Primärschlüssel --> Tupelintegrität = Ein Tupel muss eindeutig identifizierbar sein

Fremdschlüssel --> referentielle Integrität = Referenz muss eindeutig zu weisbar sein.

# Vom ERD zum relationalen ERD:

Attributiv beginnt mit tkey und sind reine Attributlieferanten

Im Bild z.B. plzOrt und sprache

Computergenerierter Alternativtext:
gehört zu 
zor 
wohnt an 
wird verwendet 
rac 
schr e b t 
a 
von 
or 

Fundamentaler Entitätstyp beginnt mit tsi und sind Tabellen mit denen wir Arbeiten. Sie haben mindestens einen oder mehr Fremdschlüssel.

Assoziativer Entitätstyp beginnt mit tass und sind Tabellen die bei einer n zu n Beziehung gebildet werden.

Im bild z.b. Kompetenz

Computergenerierter Alternativtext:
gehört zu 
zor 
wohnt an 
fig. 
autor 
geh . 
wird verwendet von 
rac 
schreibt 
zu 
ko 
betrifft 
enz 
fig. in 
f 
geb 

## Normalformen

### Erste Normalform

Eine Tabelle ist in der ersten Normalform falls die Wertebereiche der Attribute Atomar sind, also ein Feld beinhaltet immer nun einen Wert

Bei folgendem Beispiel verletzt die Spalte Typ die Erste NF. Müsste in diesem Fall in 2 Tabelleneinträge gesplittet werden

Computergenerierter Alternativtext:
Folgende Tabelle ist nicht in erster Normalform: 
ISBN 
1234567890123 
Autor 
Willi 
Titel 
Sachbuch 
Wenn Nattern knattern 
Lehrbuch 

### Zweite Normalform

Eine Tabelle ist in zweiter Normalform wenn sie in erster Normalform ist und wenn jedes Nichtschlüsselattribut vom zugehörigen Primärschlüssel voll funktional abhängig ist.

Computergenerierter Alternativtext:
6.2 Zweite Normalform (2NF) 
Eine Tabelle ist in zweiter Normalform, wenn sie in erster Normalform ist und wenn je- 
des Nichtschlüsselattribut vom zugehörigen Primärschlüssel voll funktional abhängig 
ist. 
Funktionale Abhängigkeit 
Bestimmt jeder Wert aus einer Domäne A genau einen Wert aus einer Domäne B, 
dann ist B funktional abhängig von A. 
Beispiele 
11.12.2006 -9 Montag 
9783642151255 -9 Fischer/Hofer: 
Lexikon der Informatik 
nzz . ch —+ Neue Zürcher Zeitung, 
Zürich 
A: Kalenderdatum; 
A: ISBN; 
A: 
Internet-Domänen; 
B: Wochentage: 
Buchtitel. 
B: 
Domäneninhaber 
B: 

Computergenerierter Alternativtext:
Bestimmt bei zusammengesetzten Primärschlüsseln ausschliesslich der ganze 
Schlüssel - und nicht schon ein Teil davon - einen und nur einen Attributswert (o- 
der NULL) zu jedem Attribut, dann sprechen wir von voller funktionaler Ab- 
hängigkeit. 
Beispiele 
Al: Flugnummer, A2: Kalenderdatum; B: Kapitän 
Al: StudNr, A 2: Fach, A 3: Jahr; B: Prüfungsnote 

Alle Primärschlüssel müssen eindeutig sein, Wert eines Attributs ist abhängig von allen seinen Primärschlüsseln.

Im folgenden Beispiel müsste daher die Spalte Titel gestrichen werden, da der Titel nicht abhängig von der Bestell Nr ist.

Computergenerierter Alternativtext:
Folgende Tabelle 
Bs tNr 
KGB007 
KGB007 
ist nicht in zweiter Normalform: 
Anzahl 
25 
10 
ISBN 
1234567890123 
2345678901234 
Wenn 
Wenn 
Titel 
Nattern knattern 
Wanzen tanzen 

### Dritte Normalform

Eine Tabelle ist in dritter Normalform, wenn sie in zweiter Normalform ist und kein Nichtschlüsselattribut von irgendeinem Schlüssel transitiv abhängig ist.

Computergenerierter Alternativtext:
6.3 Dritte Normalform (3NF) 
Eine Tabelle ist in dritter Normalform, wenn sie in zweiter Normalform ist und 
kein Nichtschlüsselattribut von irgendeinem Schlüssel transitiv abhängig ist. 
Transitive Abhängigkeit 
Wir sprechen von einer transitiven Abhängigkeit dann, wenn ein Wert eines 
Nichtschlüsselattributs einen anderen Attributswert bestimmt. 
B 
f (A) und C 

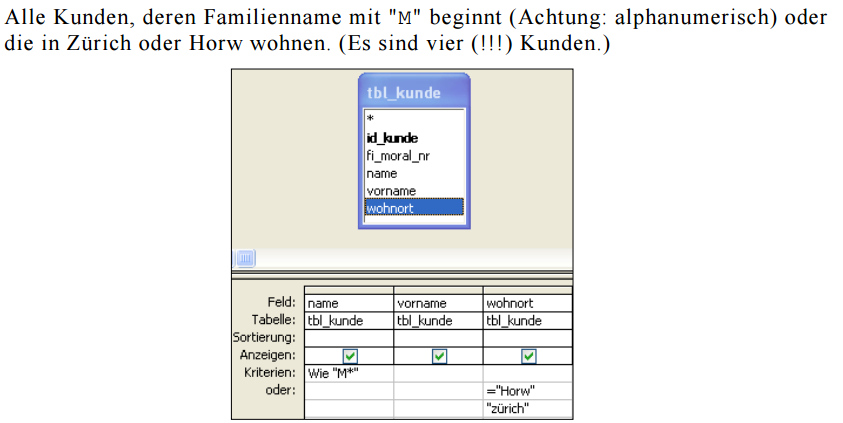
**Transitiv Abhängig** = C ist abhängig von B und B ist abhängig von A daher kann ich von A auf C schliessen

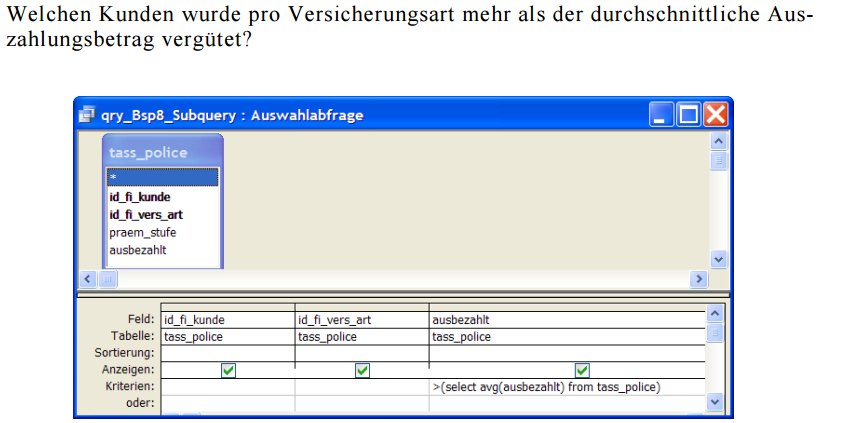
Im Folgenden Beispiel kann ich von VerlagId auf Verlag schliessen(ohne, dass VerlagId ein Primärschlüssel ist), das verletzt die Dritte NF, das heisst ich werde eine zusätzliche Tabelle Verlag erstellen, mit VerlagId als Primärschlüssel und Verlag und diese dann in meiner Buch Tabelle verknüpfen

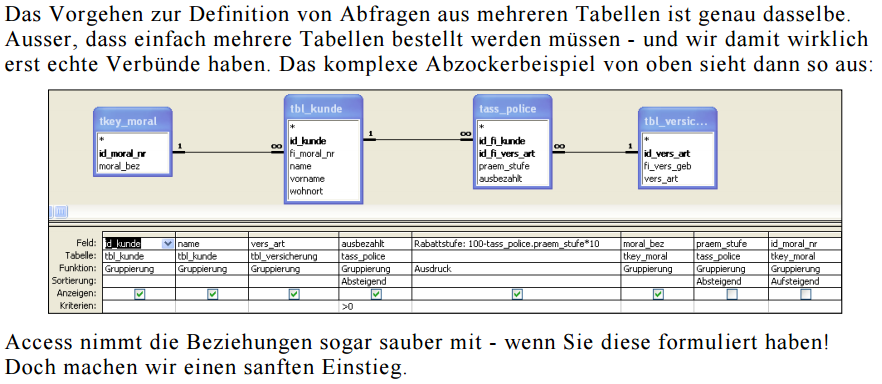
Computergenerierter Alternativtext:
Folgende Tabelle ist nicht in dritter Normalform: 
Verlag 
Kriecher 
ISBN 
1234567890123 
Verl Id 
De455 
Titel 
Wenn Nattern knattern 

# Access

## Abfragen mit Access Beispiele







### Access Abfragen mit Joins

#### Cross Join

Computergenerierter Alternativtext:
Freundllcherweise hat uns Access auch die Beziehung "mltgenommen". Klicken Sle nun 
diese einmal an und loschen Sie die Beziehung. Führen Sie die Abfrage nochmals aus. 
tkey_mordl 
tH_kunde 
m oral_bez 
tkey 
-iii 
Fel d: 
Tabelle: 
Sortieruva: 
An zaaen: 
Kr teren: 
tbl_kurde 
Das schockierende Ergebnls ist ein kartesisches Produkt - ein Cross Join. 
kundenname kundenvorname Wohnort 
Zahlun smoral 
ut 
schlecht 
sehr gut 
schlecht 
sehr gut 
schlecht 
sehr gut 
schlecht 
sehr ut 
schlecht 
schlecht 
sehr gut 
sehr ut 
schlecht 
Düsentneb 
00sentrieb 
Hauser 
Hauser 
Hauser 
Hilton 
Hilton 
Hilton 
Klausen 
Klausen 
klausen 
Meier 
Meier 
Meier 
Menzer 
Menzer 
Menzer 
Muller 
Müller 
Muller 
Daniel 
Daniel 
Daniel 
Martin 
Martin 
Martin 
Heidi 
Heidi 
Heidi 
Max 
Mar 
Max 
Claudia 
C audia 
Claudia 
Entenhausen 
Entenhausen 
entenhausen 
Kriens 
Kriens 
Kriens 
Malibu 
Malibu 
Malibu 
ZOnch 
ZOIich 
ZOnch 
Luzern 
Luzern 
Luzern 
Bern 
Bern 
Bern 

Bein Cross Join werden also anzahlt Tabellen Zeilen miteinander Multipliziert, so dass jede mögliche Konstellation entsteht

#### Inner Join

Vor einem Inner Join wird immer ein Cross Join gemacht jedoch nur die Datensätze angezeigt die eine Verknüpfung haben

Computergenerierter Alternativtext:
8.3.1 
Inner Join 
Gehen wir einleitend von einer ganz normalen Fragestellung aus. 
welche Versicherungsart? 
tbl kunde 
oral _nr 
_StLfe 
—Ube zahl t 
Feld: 
Welche Kunden haben 
Tabelle: 
Krterien: 
Aufstei rd 
vorn erne 
fi 
Das Ergebnis ist der der gewollte «Normalfall»: ein Verbund (Join) von zwei oder mehr 
Tabellen. Zwecks Abgrenzung zu den weniger normalen Fällen nennen ihn künf g Inner 
Join. 
Kundenname Kundennyrnam Wohnort 
Versicherun scode 
1700 
Hauser 
Hauser 
Hauser 
Hauser 
Hauser 
Klausen 
Meer 
Meier 
Menzel 
Menzer 
Menzel 
Menzer 
Müller 
Muller 
Muller 
Müller 
Matin 
Martin 
Matin 
Malin 
Heidi 
Heidi 
Heidi 
Claudia 
Claudia 
Claudia 
Claudia 
Claudia 
Kriens 
Kriens 
"eng 
Kriens 
Kriens 
Zürich 
Zürich 
Zürich 
Luzern 
Luzern 
Horw 
Bern 
Sem 
Sem 

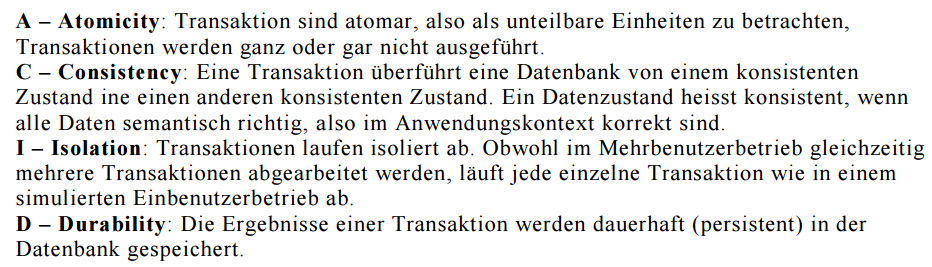
#### Outer Join

Dasselbe wie Inner Join, jedoch wird jeder Datensatz der Ausgangstabelle angezeigt, auch wenn keine Beziehung besteht.

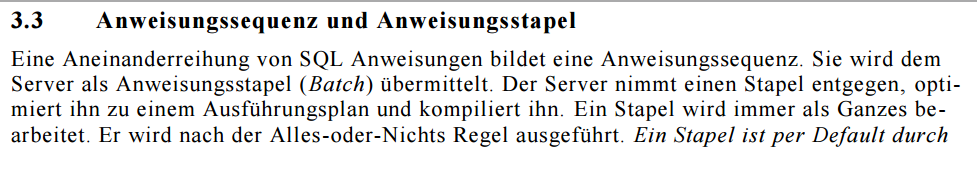
Computergenerierter Alternativtext:
tkey_moral 
tbl kunde 
moral_bez 
fi_moral_nr 
name 
vorname 
Feld: 
name 
Tabelle: 
tkey_moral 
tbl_kunde 
Sortierung: 
Anzeigen : 
Kriterien: 
Als Resultat erhalten Wir folgendes: 
Zahlungsmorall Kundenname 
schlecht 
schlecht 
schlecht 
gut 
gut 
sehr gut 
sehr gut 
unbekannt 
Hilton 
Hauser 
Düsentrieb 
Meier 
Klausen 
Müller 
Menzer 

Datenmanagement (INM22) BED

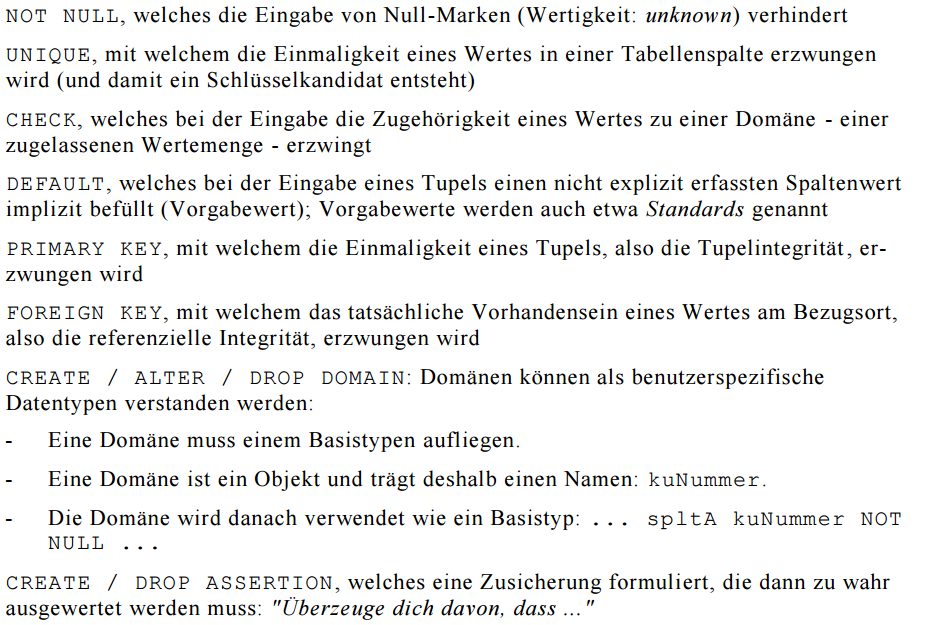
# ACID



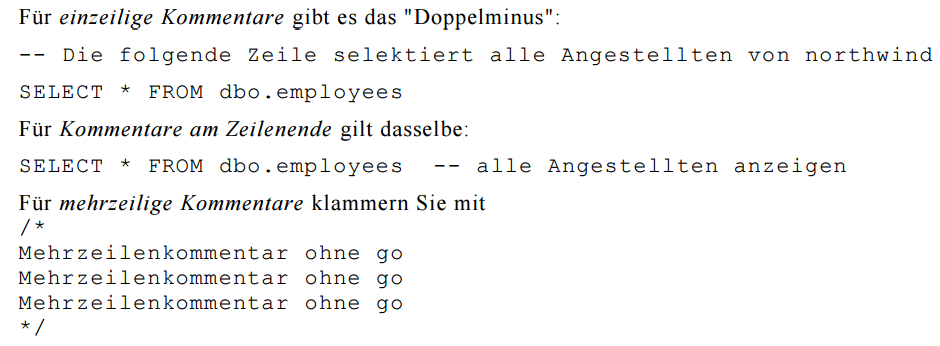
# SQL-Queries



## Integrity Constraints

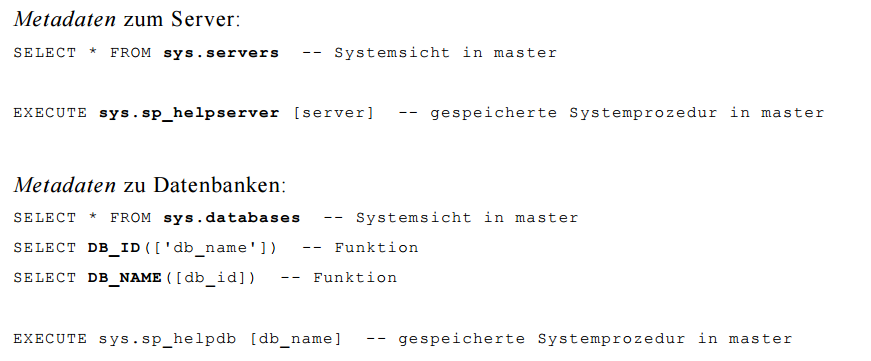


## Kommentare



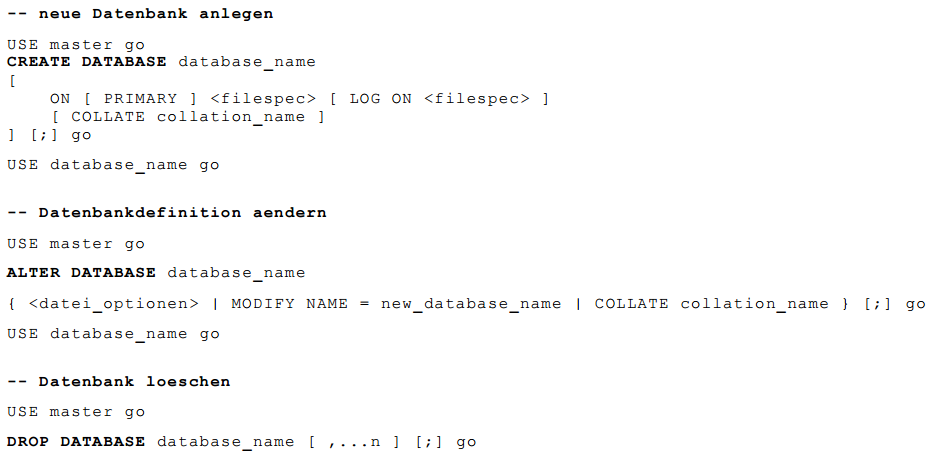
## Datenbanken

USE „Datenbankname“ wird verwendet um zwischen Datenbanken zu wechseln

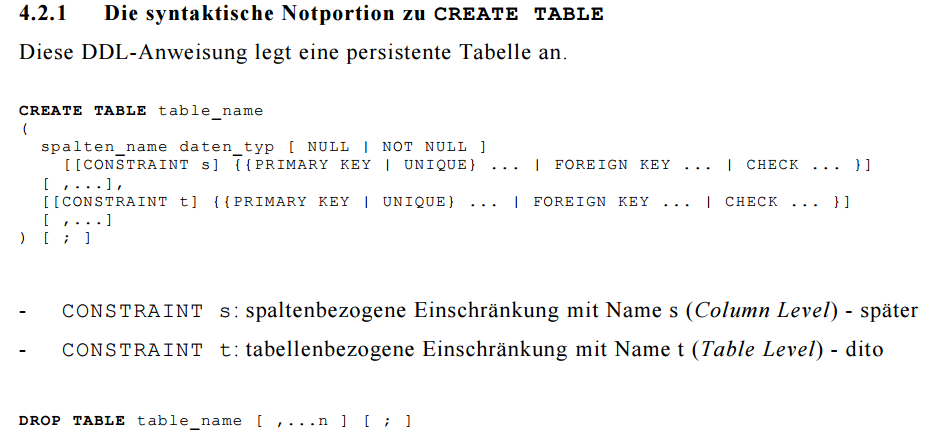


## DDL

### CREATE DATABASE



### CREATE TABLE



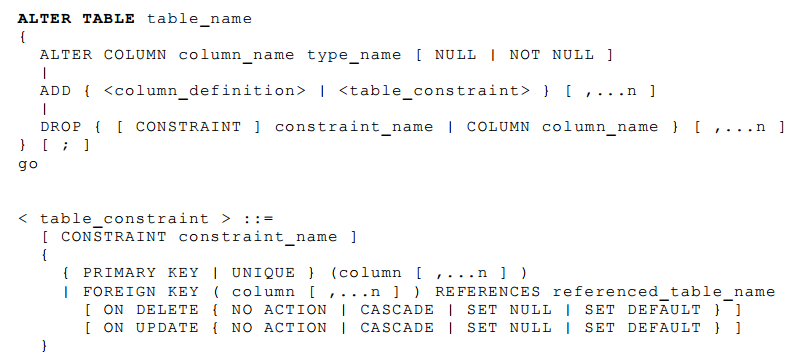
CREATE TABLE tkey\_moral(

id\_moral\_nr INT NOT NULL,

moral\_bez VARCHAR(30) NOT NULL

CONSTRAINT UQ\_moral\_bez UNIQUE

)



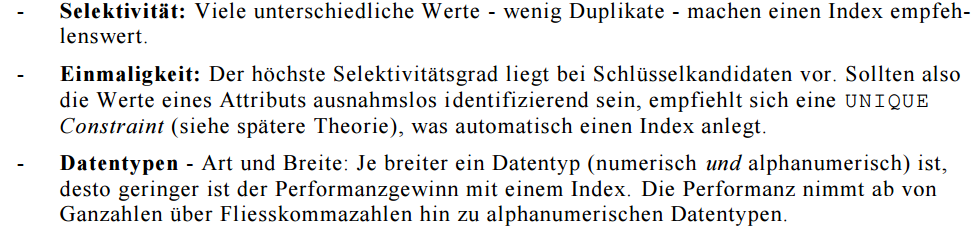
ALTER TABLE tass\_police

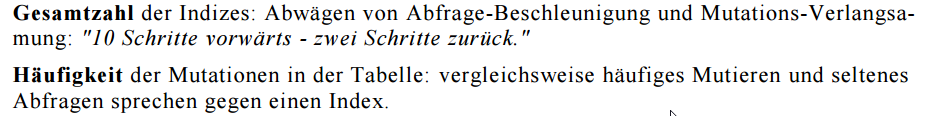
ADD PRIMARY KEY (id\_fi\_kunde,id\_fi\_vers\_art),

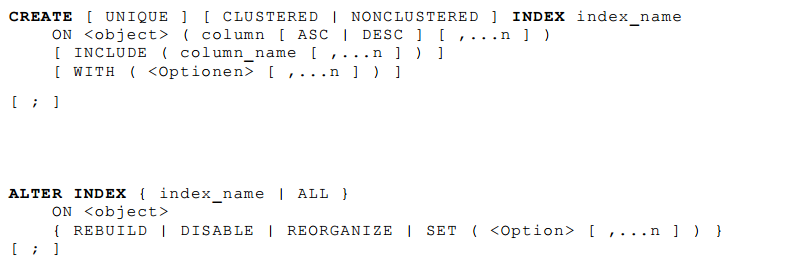
FOREIGN KEY (id\_fi\_kunde) REFERENCES tbl\_kunde ON DELETE CASCADE,

FOREIGN KEY (id\_fi\_vers\_art) REFERENCES tkey\_versicherung ON DELETE CASCADE

### CREATE INDEX







## 

CREATE INDEX ix\_kunde\_name

ON tbl\_kunde (name)

## 

## 

ALTER INDEX ix\_kunde\_name

ON tbl\_kunde REORGANIZE



DROP INDEX ix\_kunde\_name on tbl\_kunde

### Domänen/Defaults

Domänen werden verwendet um Werte von Feldern zu definieren/beschränken. Mit dem Default Befehl kann einem Feld einen Standardwert geben, der gesetzt wird falls beim INSERT INTO kein Wert für dieses Feld angegeben wird.

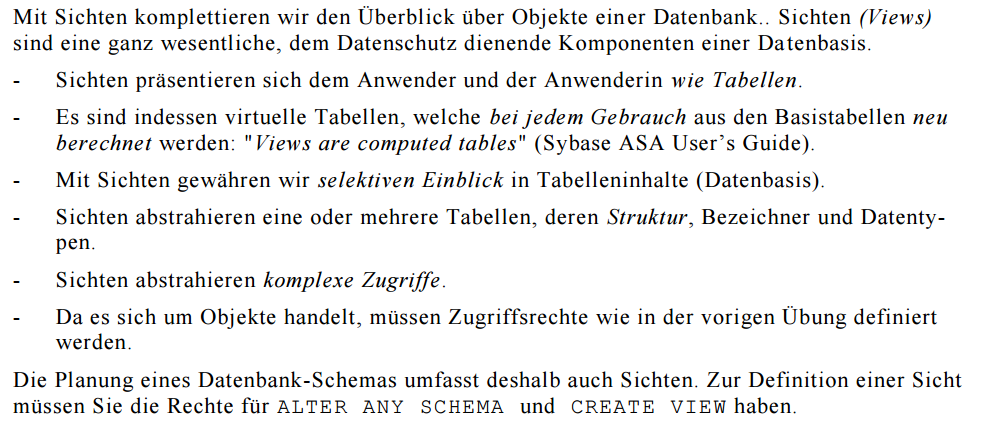
--tp\_moralisches ist nun ein Datentyp numeric mit Zahlen von 0-9

CREATE TYPE tp\_moralisches FROM numeric(9,0)

ALTER table tbl\_kunde

ADD default 'Luzern' for wohnort

### CREATE VIEW



## 

CREATE VIEW v\_kunden (id, name, vorname, ort , police, gebiet ) AS

SELECT k.id\_kunde, k.name, k.vorname, k.wohnort, v.vers\_bez, v.vers\_gebiet

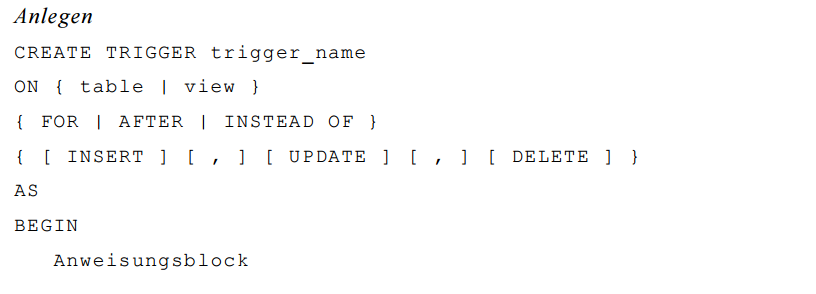
FROM tbl\_kunde AS k, tkey\_versicherung AS v, tass\_police AS p

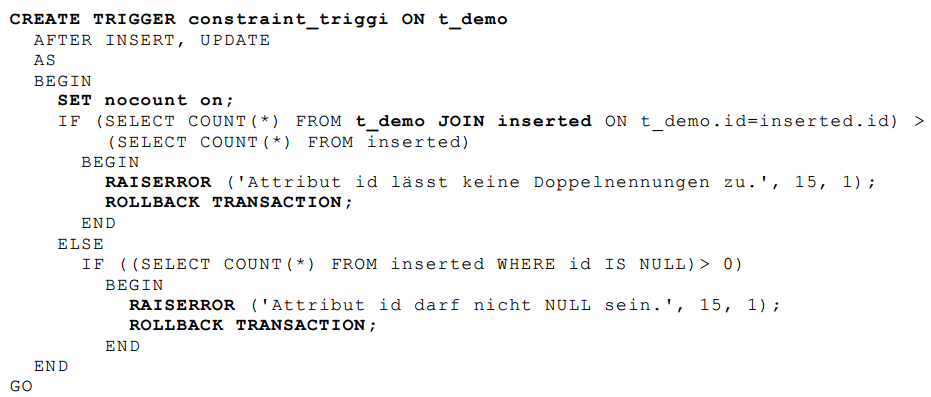
WHERE k.id\_kunde = p.id\_fi\_kunde

AND p.id\_fi\_vers\_art = v.id\_vers\_art

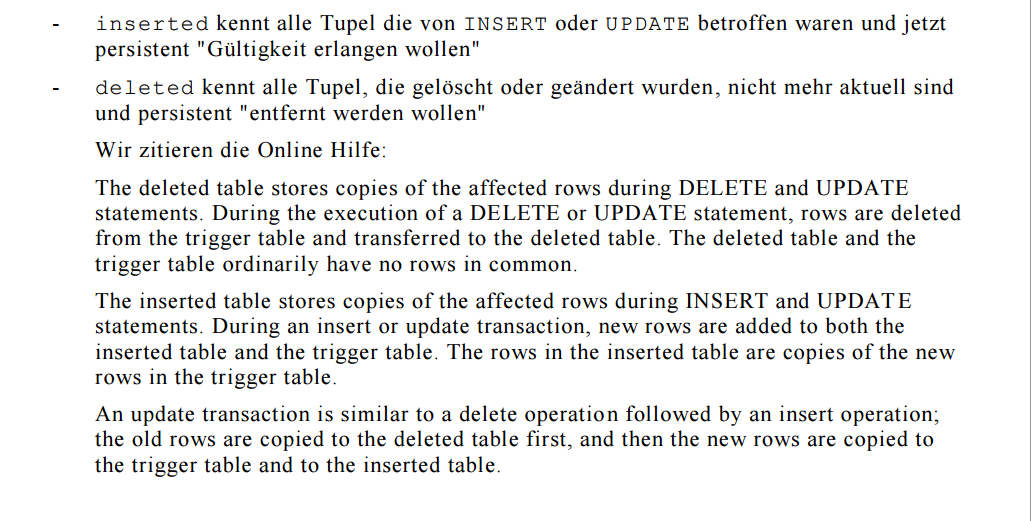
SELECT \* FROM v\_kunden

### Trigger

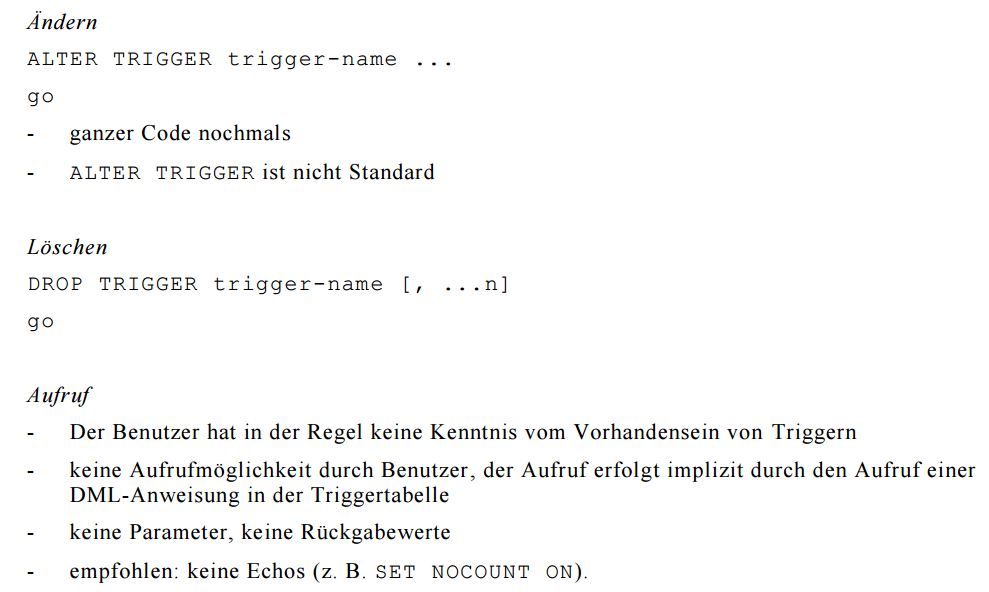




## 

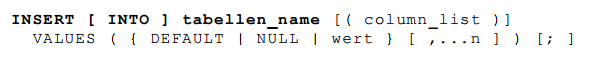


## 



## DML

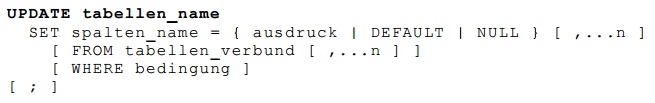
### INSERT INTO



INSERT INTO tbl\_kunde (id\_kunde,name,vorname,wohnort,jahrgang)

VALUES (4711,'Düsentrieb','Daniel','Entenhausen',1952)

### UPDATE

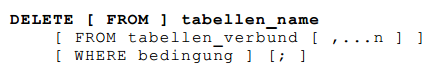


UPDATE tbl\_kunde

SET id\_kunde ='4711'

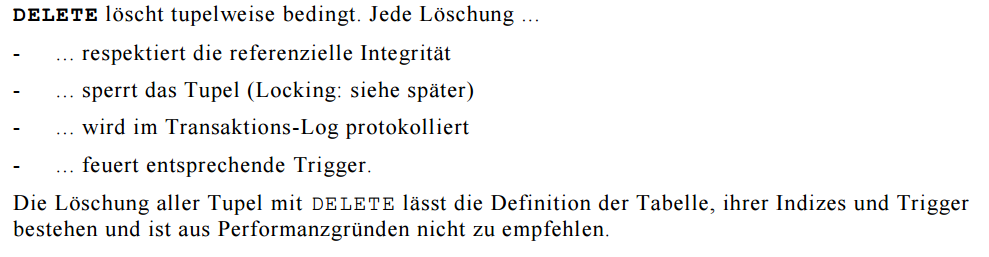
WHERE name = 'Hauser'

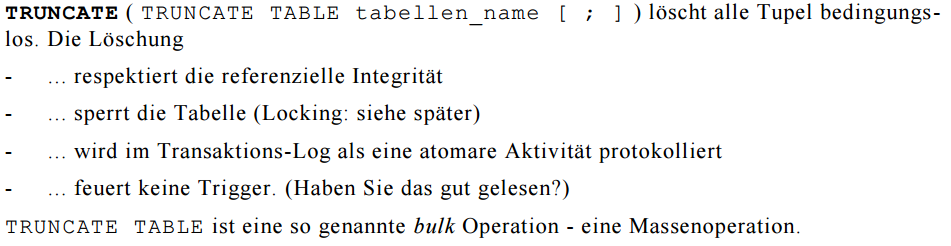
### DELETE



DELETE FROM tbl\_kunde

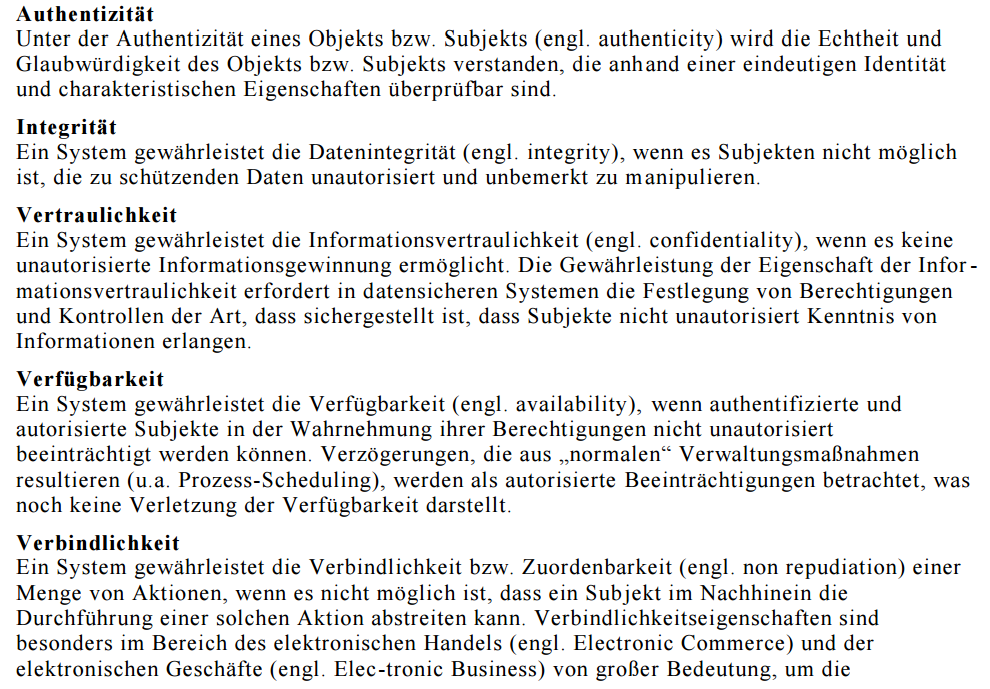
WHERE name ='Fischer' OR name ='M?ller' AND vorname ='Martin'



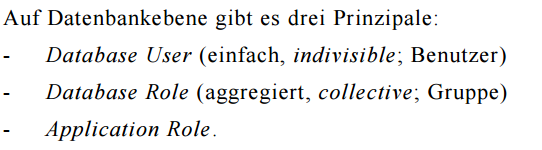




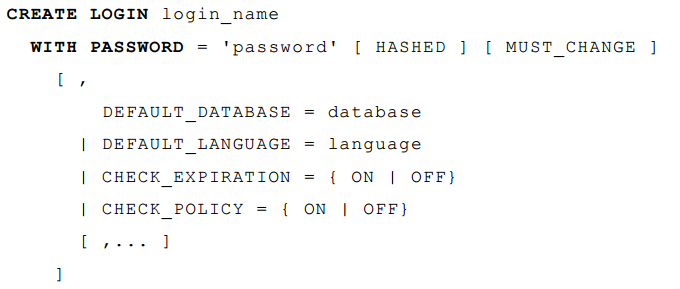
## DCL







### CREATE LOGIN



CREATE LOGIN tom

### CREATE USER



CREATE USER tom

### CREATE ROLE

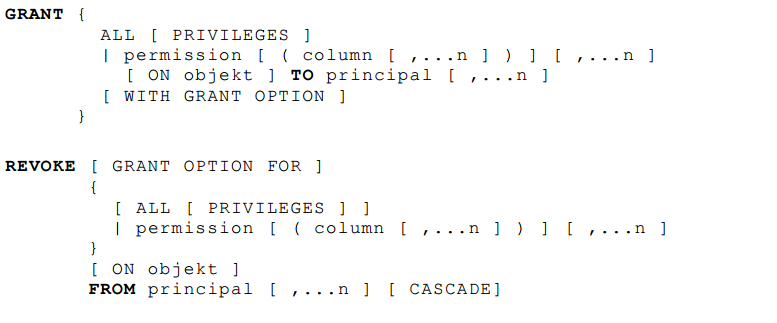


CREATE ROLE gl

--Fügt tom zur Rolle gl hinzu

EXEC sp\_addrolemember 'gl', 'tom'

### GRANT/REVOKE

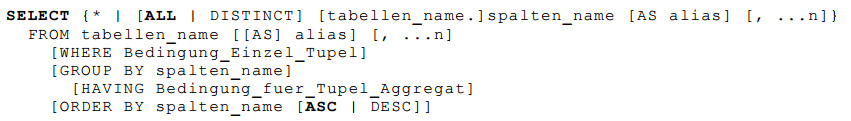


REVOKE ALL PRIVILEGES FROM romulus

GRANT SELECT, UPDATE, INSERT, DELETE ON tbl\_kunde TO tom

## DQL

### SELECT



SELECT dt\_stueck\_titel FROM tbl\_stueck

WHERE dt\_stueck\_titel LIKE '%Heaven%'

OR dt\_stueck\_titel LIKE '%Hell%'

SELECT COUNT( DISTINCT dt\_stueck\_titel) FROM tbl\_stueck

WHERE dt\_stueck\_titel LIKE 'Let%'

SELECT MAX(dt\_zeit), MIN(dt\_zeit), AVG(dt\_zeit) FROM tbl\_stueck

SELECT SUM(dt\_zeit) FROM tbl\_stueck

SELECT i.dt\_name AS interpret, COUNT (s.fi\_interpret) AS stuecke

FROM tbl\_stueck AS s, tkey\_interpret AS i

WHERE i.id\_interpret = s.fi\_interpret

GROUB BY i.dt\_name

HAVING COUNT(s.fi\_interpret) >= 10

ORDER BY stuecke DESC

### SELECT JOIN mit WHERE

SELECT st.dt\_stao AS Standort

FROM tkey\_standort AS st, tbl\_stueck AS s, tass\_stueck\_stil AS ss

WHERE st.id\_stao = s.fi\_stao

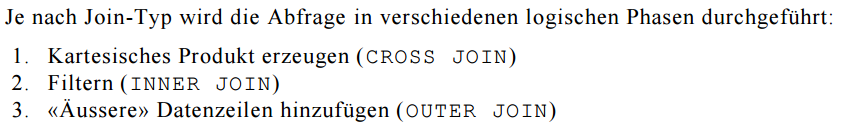
AND ss.id\_fi\_stueck\_nr = s.id\_stueck\_nr

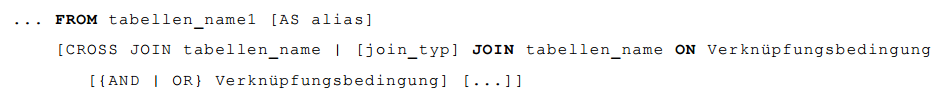
AND ss.id\_fi\_stil = 'f'

GROUP BY st.dt\_stao

ORDER BY Standort desc

### SELECT JOIN mit JOIN





SELECT st.dt\_stao AS Standort

FROM tbl\_stueck as s

JOIN tass\_stueck\_stil AS ss ON ss.id\_fi\_stueck\_nr = s.id\_stueck\_nr

JOIN tkey\_standort AS st ON st.id\_stao = s.fi\_stao

WHERE ss.id\_fi\_stil ='f'

GROUP BY st.dt\_stao

ORDER BY Standort desc

### Sub-Queries

SELECT dt\_stueck\_titel AS Titel, dt\_zeit AS Zeit

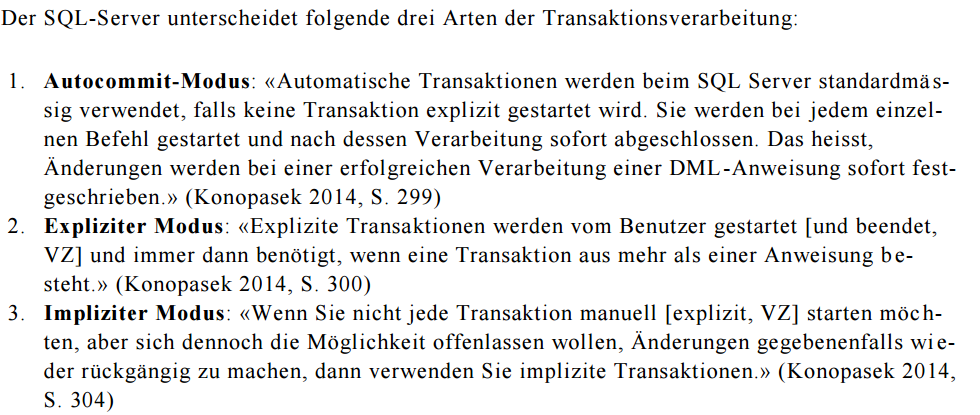
FROM tbl\_stueck

WHERE dt\_zeit BETWEEN (SELECT AVG(dt\_zeit) FROM tbl\_stueck)\*0.9

AND (SELECT AVG(dt\_zeit) FROM tbl\_stueck)\*1.1

ORDER BY dt\_zeit

### Transaktionen



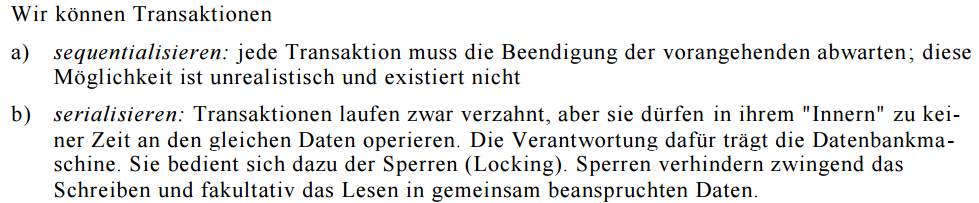
BEGIN TRANSACTION;

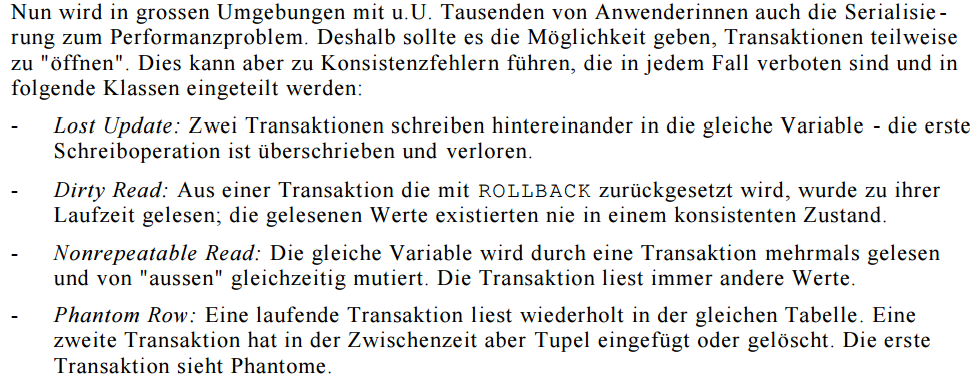
UPDATE tass\_police

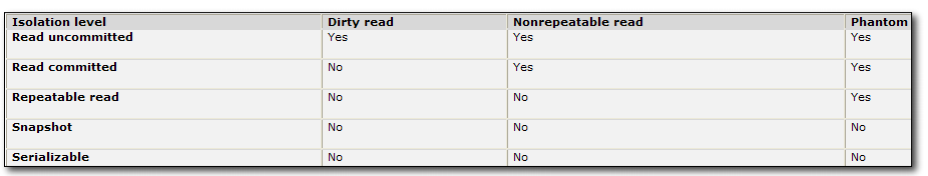
SET bezahlt = 0

WHERE id\_fi\_kunde =3533 AND id\_fi\_vers\_art = 1300;

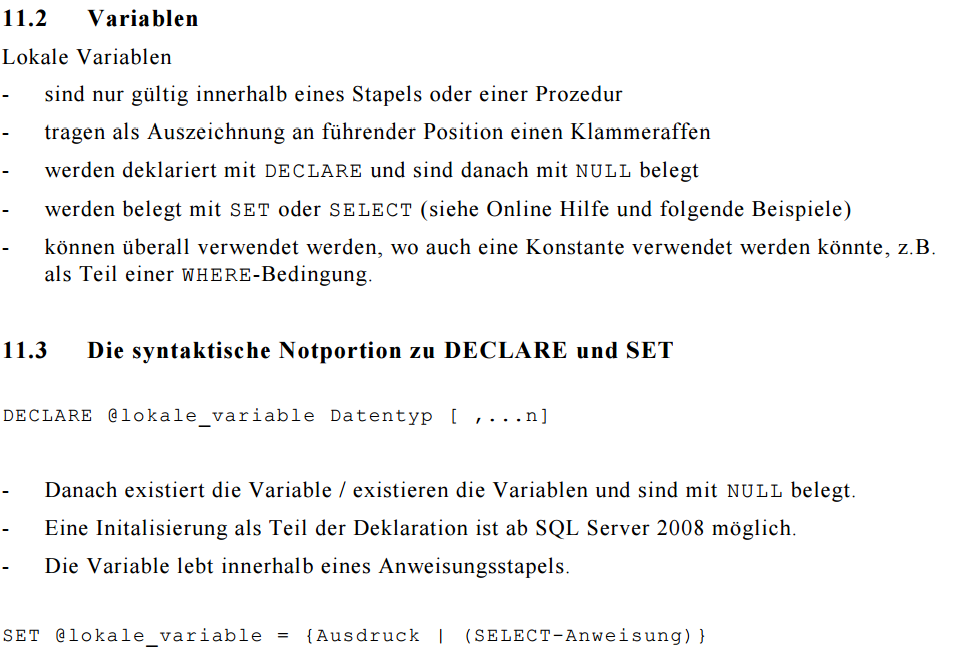
COMMIT TRANSACTION/ ROLLBACK TRANSACTION;

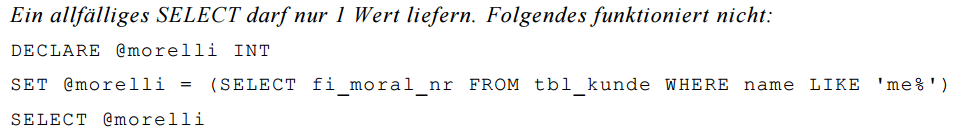


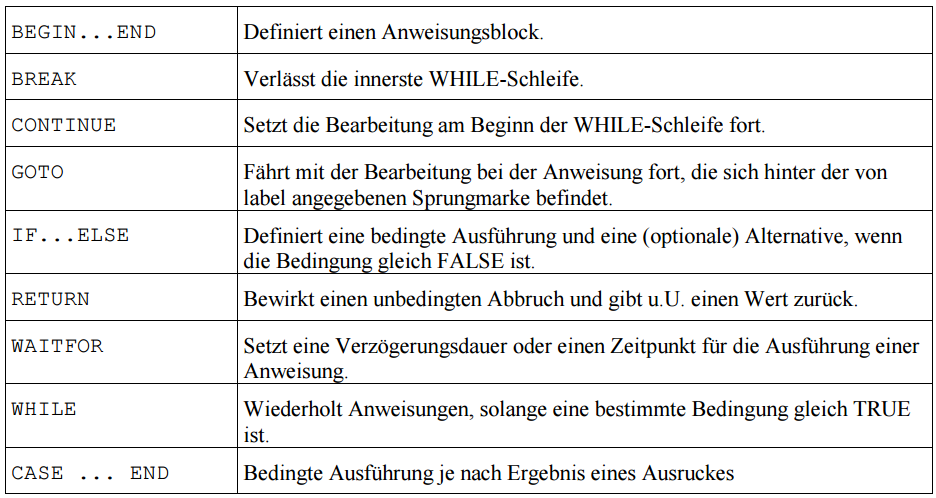




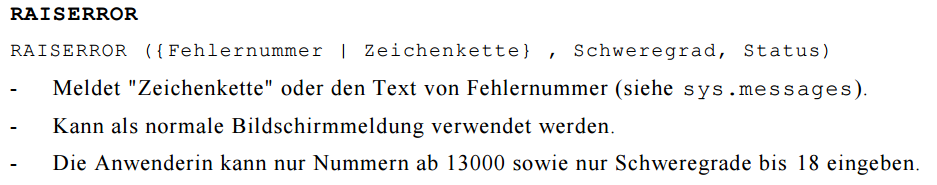
## Programmierung mit SQL

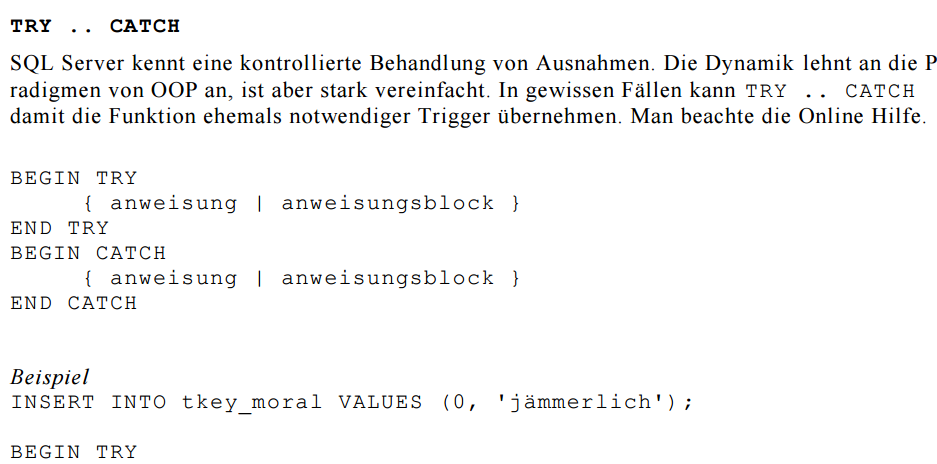




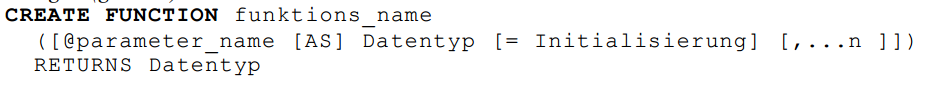


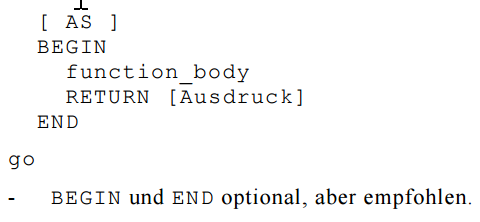






### FUNCTIONS





CREATE FUNCTION f\_bezahlt\_in\_dollar (@name varchar(30), @vers\_bez varchar(50), @kurs Decimal(8,2))

RETURNS decimal(8,2)

AS

BEGIN

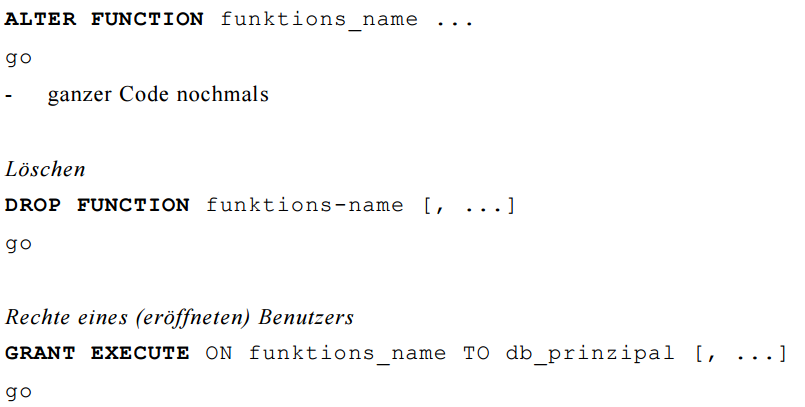
RETURN (SELECT bezahlt\*@kurs AS 'Dollari' FROM tass\_police AS p

JOIN tkey\_versicherung AS v on v.id\_vers\_art = p.id\_fi\_vers\_art

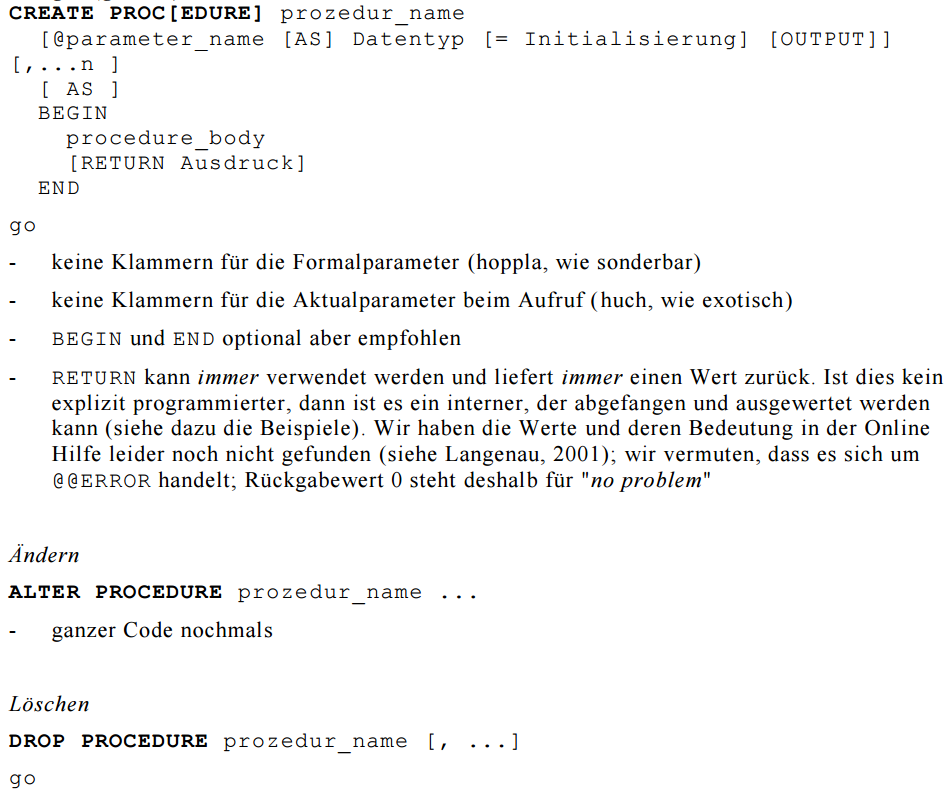
JOIN tbl\_kunde as k ON k.id\_kunde = p.id\_fi\_kunde

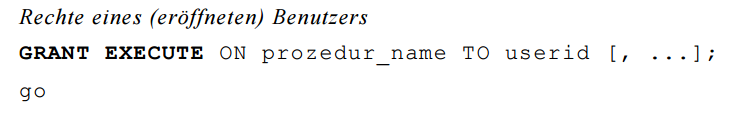
WHERE k.name = @name AND v.vers\_bez = @vers\_bez)

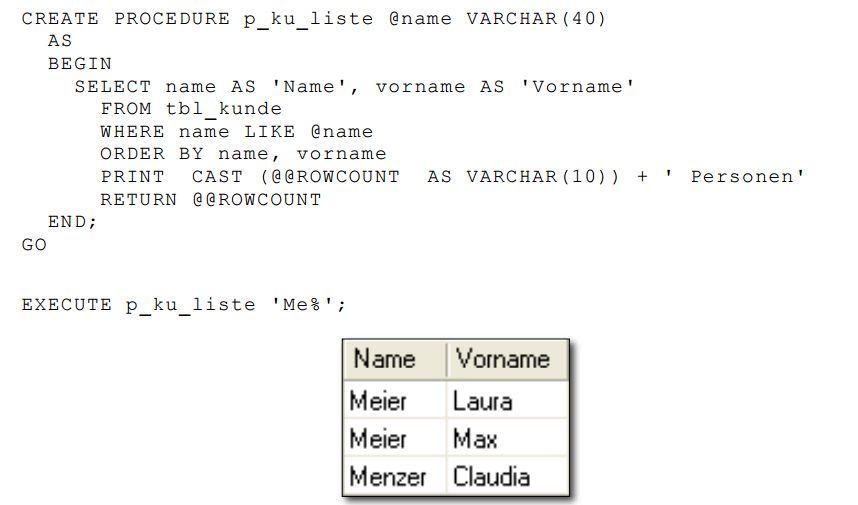
END



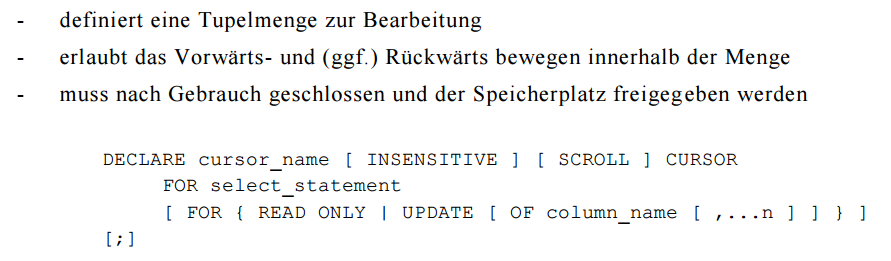
### PROCEDURES

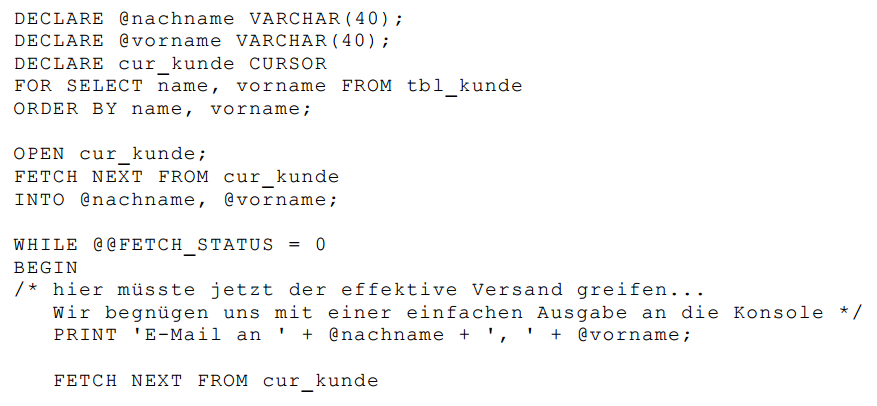


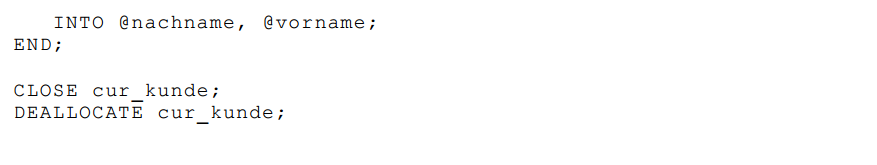




### Cursor







## Systemkatalog

