integrata cegos

SQL – Die Sprache:
Interaktives Arbeiten mit SQL

Gesamtinhaltsverzeichnis

1

Relationale Datenbanken und Datenbankmanagement

1 Relationale Datenbanken und Datenbankmanagement

1.1 Relationale Datenbank

Tverlag

Verlagnr	Verlag
1111	Forkel

Tbuch

Buchnr	Erschj	Preis	Titel	Verlagnr
5	1988	3.50	Ansichten eines Clowns	NULL
27	NULL	99.99	die Jüdin von Toledo	NULL
6	1988	20.50	die Blechtrommel	NULL
7	1989	99.99	Der Name der Rose	NULL
8	1977	0.50	der Butt	1111
9	1990	55.00	DB2 fuer Sie	1111
11	1990	NULL	Elvis in Heidelberg	NULL
12	1989	NULL	a guide to db2	NULL
18	1989	99.99	Database Systems	NULL
1	NULL	NULL	C	NULL
2	NULL	NULL	C	NULL

Beispiel für eine kleine Datenbank mit zwei Tables.

- Die Spalte Tverlag.Verlagnr ist Primärschlüssel der Table Tverlag.
- Die Spalte Tbuch.Buchnr ist Primärschlüssel der Table Tbuch.
- Die Spalte Tbuch.Verlagnr ist Fremdschlüssel und präsentiert die Verlagsnummer des Buches (zum Beispiel ist das Buch mit der Buchnummer 8 dem Verlag 1111 zugeordnet).
- Die Spalten Tbuch.Verlagnr, Tbuch.Erschj und Tbuch.Preis präsentieren in einigen Zeilen die NULL.

Beachten Sie bitte: die Zeilen der Table Tbuch sind weder nach Titel noch nach Buchnr sortiert angelistet.

```
CREATE TABLE Tverlag
  (
   Verlagnr   INTEGER    NOT NULL
  ,Verlag     CHAR(20)   NOT NULL
  ,PRIMARY KEY (Verlagnr)
  )
;
CREATE TABLE Tbuch
  (
   Buchnr         INTEGER      NOT NULL
  ,Erschj         DECIMAL(4)
  ,Preis          DECIMAL(7,2)
  ,Verlagnr       INTEGER
  ,Titel          VARCHAR(127) NOT NULL
  ,PRIMARY KEY (Buchnr)
  )
;
ALTER TABLE Tbuch ADD CONSTRAINT
                  Tbuchconpreis
                  CHECK ( 0.00 <  Preis )
;
ALTER TABLE Tbuch ADD CONSTRAINT
                  FK_Tbuch_Tverlag
                  FOREIGN KEY (Verlagnr)
                  REFERENCES Tverlag(Verlagnr)
                  --ON DELETE NO ACTION
;
--DROP TABLE Tbuch
--;
--DROP TABLE Tverlag
--;
```

Eine **relationale Datenbank** ist eine Datenbank, die vom Benutzer als eine Menge von Tables wahrgenommen wird. Der gesamte Informationsgehalt der Datenbank wird in genau einer Weise präsentiert, nämlich als explizite Werte in Spaltenpositionen von Zeilen in Tables.

Zum Zwecke der Präsentation einer Table werden die Spalten und Zeilen natürlich immer in einer **für den Benutzer sinnvollen Weise** angelistet (Anordnung der Spalten einer Table von links nach rechts und Sortierung der Zeilen aufsteigend bzw. absteigend nach gewissen Kriterien).

Eine **Table** besteht aus einer oder mehreren **Spalten** und präsentiert keine, eine oder mehrere **Zeilen**. Jede **Spalte** hat einen Namen und einen zugrunde liegenden Datentyp (system- oder benutzerdefiniert). Eine **Zeile** enthält für jede Spalte einen Wert des zugrunde liegenden Datentyps.

Schlüsselkandidat, Primärschlüssel/Alternativschlüssel
Jede Table hat wenigstens einen Schlüsselkandidaten. Ein Schlüsselkandidat ist eine Spalte bzw. eine **minimale (irreduzible) Kombination von Spalten** die jede Zeile einer Table eindeutig identifiziert. D. h. für alle Zeiten und für die möglichen wechselnden Inhalte der Table gilt: zwei verschiedene Zeilen in der Table haben auch verschiedene Schlüsselkandidatenwerte.
Für jede Table wird einer der Schlüsselkandidaten als Primärschlüssel festgelegt. Eventuell vorhandene weitere Schlüsselkandidaten werden als Alternativschlüssel bezeichnet.

Ein **Fremdschlüssel** (eine oder mehrere Spalten) darf nur Werte präsentieren, die als Werte eines Primärschlüssels (bzw. Alternativschlüssels) vorhanden sind. Ein Fremdschlüsselwert stellt eine Beziehung her zu der Zeile, die den entsprechenden Primärschlüsselwert bzw. Alternativschlüsselwert enthält.

Eine **Integritätsbedingung (integrity constraint)**, die mit der Datenbank verbunden ist, darf **nie FALSCH** sein, sie muss vom Datenbankmanagementsystem (DBMS) durchgesetzt werden. Eine Integritätsbedingung kann als formaler Ausdruck einer „Geschäftsregel“ („business rule“) betrachtet werden.

NULL und dreiwertige Logik
Der Zugang zum Problem der fehlenden Information basiert bei SQL und deshalb bei den meisten kommerziellen Produkten auf der NULL und der dreiwertigen Logik (three-valued logic). Der Begriff „NULL-Wert“ wird im informellen Sprachgebrauch sehr oft benutzt, NULL ist aber kein Wert. Es wäre besser, den Begriff „Null-Marke“ zu benutzen. **NULL** bedeutet, dass der Wert in der Datenbank nicht definiert ist. Dies kann bedeuten, dass dieser Wert generell nicht existiert, dass er nicht bekannt ist oder dass er noch nicht erfasst ist.

1.2 INSERT, UPDATE, DELETE und SELECT

```
SELECT
  Buchnr, Preis, Titel
FROM Tbuch
WHERE Erschj = 1989
;
```

Buchnr	Preis	Titel
7	99.99	Name der Rose
18	99.99	Database Systems
12	NULL	a guide to db2

Lesen von Daten:

```
SELECT Buchnr, Preis, Titel
FROM Tbuch
WHERE Buchnr = 7
;
```

Einfügen einer neuen Zeile:

```
INSERT INTO Tbuch
(Buchnr, Erschj, Preis, Titel) VALUES
(99, 1991, 87.00, 'Oracle & DB2 & SQL Server')
;
```

Verändern von Daten:

```
UPDATE Tbuch
SET Preis = 25.00
WHERE Buchnr = 7
;
```

Löschen von Zeilen:

```
DELETE FROM Tbuch
WHERE Erschj = 1990
;
```

1.3 relationales Datenbankmanagementsystem

Ein Datenbankmanagementsystem (DBMS) ist ein System, das auf einem Computer Daten (Information) speichert und verwaltet, und dem Benutzer ermöglicht, diese Daten nach Bedarf zu lesen und zu ändern. Die Daten werden in Sätzen gespeichert, mehrere Sätze bilden eine Datei.

Eine Datenbank-Database ist ein Repository oder Container für eine Menge solcher Dateien.

Ein relationales Datenbankmanagementsystem (RDBMS) ermöglicht einem Benutzer das Erstellen und Pflegen einer relationalen Datenbank.

Die Benutzer eines relationalen Datenbankmanagementsystems können unter anderem folgende Operationen ausführen.

INSERT Einfügen von neuen Zeilen in eine bestehende TABLE

UPDATE Verändern von Daten in einer bestehenden TABLE

DELETE Löschen von Zeilen in einer bestehenden TABLE

SELECT Lesen von Daten aus bestehenden TABLEs

INSERT, UPDATE, DELETE und SELECT sind Operationen der Datenbanksprache SQL (Structured Query Language).

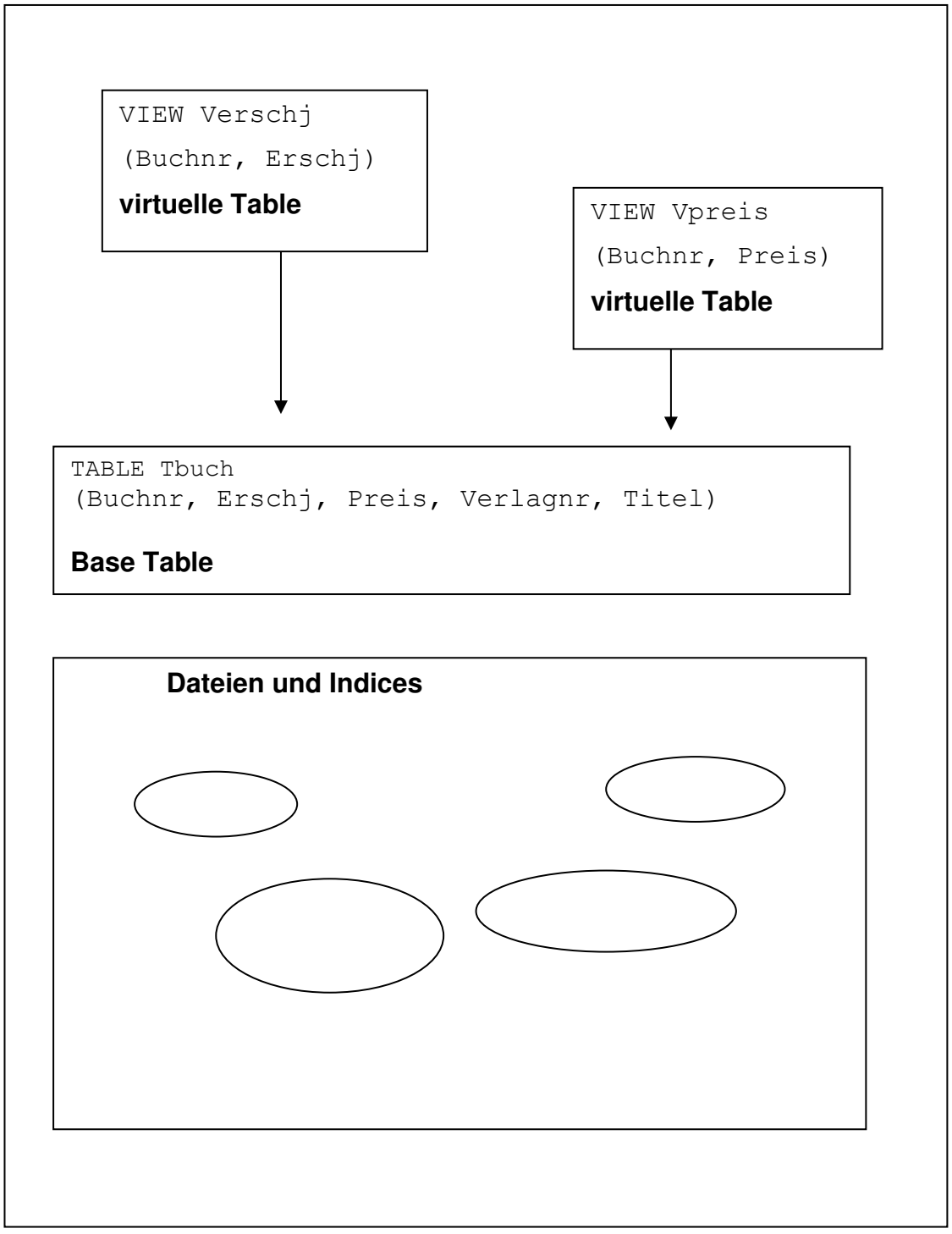
CREATE Hinzufügen einer neuen (leeren) TABLE zur Datenbank

DROP Entfernen einer TABLE (leer oder nicht-leer)

ALTER Ändern der Beschreibung für eine TABLE

CREATE, ALTER, DROP sind Operationen der Datenbanksprache SQL (Structured Query Language).

1.4 TABLE, VIEW, Daten werden präsentiert



Eine relationale Datenbank ist eine Datenbank, die vom Benutzer als eine Menge von Tables (TABLE bzw. VIEW) wahrgenommen wird.

VIEWs sind auch Tables!

Die Daten, die in Tables (TABLE bzw. VIEW) **als Zeilen sichtbar** sind, werden auf der internen physischen Ebene in Dateien und Indices **als Sätze** gespeichert.

Eine Table (TABLE bzw. VIEW) ist eine Abstraktion, die Details der Speicherung der Daten in Dateien und/oder Indices sind für den Benutzer nicht sichtbar.

Eine Formulierung wie „In einer relationalen Datenbank werden die Daten physisch gespeichert in Tables“ ist falsch und irreführend.

Eine Formulierung wie „die Sätze in einer Table“ ist weit verbreitet, aber sehr irreführend!

1.5 Historie

1969/70	E. F. Codd: "A Relational Model of Data for Large Shared Data Banks"
	(IBM San Jose Research Laboratory)
1974	SEQUEL als Vorläufer von SQL wird definiert
1975/79	SYSTEM R als Prototyp wird im Labor der IBM entwickelt
1979	ORACLE
1980	IBM SQL/DS für DOS/VSE, VM/CMS

Datenbankmanagementsysteme am Markt

Microsoft SQL Server, Access

SYBASE Adaptive Server Enterprise

ORACLE Oracle Database Server

IBM

DB2 for VSE and VM

DB2 for z/OS

DB2 for iSeries

DB2 for Linux UNIX and Windows

Informix

und viele weitere Hersteller mit ihren Produkten:

Ingres Postgres Mysql Teradata ...

der SQL-Standard

SQL86 SQL89 SQL:1992 SQL:1999 SQL:2003 SQL:2008 SQL:2011

1.6 Das relationale Datenmodell

Die Produkte am Markt, Relationale Datenbankmanagementsysteme bzw. SQL-Datenbankmanagementsysteme genannt, haben eine formale Grundlage, das relationale Datenmodell. Das relationale Datenmodell ist von **Edgar F. Codd** entwickelt worden. Codds Modell ist das einzige Datenmodell, das auf Mathematik beruht.

Das relationale Datenmodell besitzt die folgenden drei Aspekte:

- **Datenstruktur:** Die Daten in der Datenbank werden vom Benutzer als Tables wahrgenommen, die Daten werden als Tables präsentiert.
- **Datenintegrität:** Die Daten erfüllen gewisse Integritätsbedingungen (Integrity Constraints).
- **Datenmanipulation:** Die verfügbaren Operatoren zur Manipulation der Daten (Lesen, Ändern, Einfügen, Löschen) sind mengenorientierte Operatoren, mit deren Hilfe aus Tables neue, andere Tables abgeleitet werden können.

Das Relationale Modell beinhaltet die folgenden Integritätsbedingungen:

Entity Integrity Constraint: Keine Komponente eines Schlüsselkandidaten (Primärschlüssel bzw. Alternativschlüssel) darf ‚NULL-Werte' annehmen.

Referential Integrity Constraint: Jeder ‚nicht-NULL' Fremdschlüsselwert stimmt mit einem Wert des relevanten Schlüsselkandidaten der referenzierten Table überein.

Diese Integritätsbedingungen werden bei entsprechender Datendefinition vom Datenbankmanagementsystem durchgesetzt.

Das relationale Modell sagt nichts, aber auch gar nichts über die physische Speicherung der Daten. Diese ist dem jeweiligen Produkt, der Implementierung durch das Datenbankmanagementsystem vorbehalten.

Datenstruktur

Tables sind die logische Struktur in einer **relationalen Datenbank**. Eine **Table** besteht aus einer oder mehreren **Spalten** und präsentiert keine, eine oder mehrere **Zeilen**. **Eine Table ist die graphische Darstellung einer Relation.** Relation ist der mathematische Begriff für Table. Tuple ist der mathematische Begriff für Zeile. Attribut ist der mathematische Begriff für Spalte.

Jede **Spalte** hat einen Namen und einen zugrunde liegenden Datentyp (system- oder benutzerdefiniert). Eine **Zeile** enthält für jede Spalte einen Wert des zugrunde liegenden Datentyps.

Der **Datentyp** kann **skalar** sein wie zum Beispiel INTEGER, DECIMAL(7,2), CHAR(20). Der **Datentyp** kann aber auch **nonskalar** sein wie zum Beispiel ARRAY INTEGER [12] (der Type Generator ARRAY wird vom SQL Standard unterstützt).

Ein **skalarer Datentyp** kann von beliebiger innerer Komplexität sein. Die Werte eines skalaren Datentyps können nur mit Hilfe von Funktionen bzw. Methoden bearbeitet werden, die für den Datentyp definiert sind. Für Datentyp ist auch der Begriff Domain bzw. Klasse (Object Class) gebräuchlich.

Datenintegrität

Eine Integritätsbedingung (Integrity Constraint) ist ein so genannter boolscher Ausdruck, der mit einer Datenbank verbunden und nie falsch (FALSE) ist.

Eine Integritätsbedingung muss im Datenbankmanagementsystem formal deklariert werden und das DBMS muss sie durchsetzen.

```
ALTER TABLE Tbuch ADD CONSTRAINT Tbuchconpreis
CHECK ( 0.00 < Preis );
```

Zur Durchsetzung dieser Integritätsbedingung muss das System alle Operationen überwachen, die ein neues Buch einfügen oder den Preis eines existierenden Buches ändern wollen. Bei der ersten Deklaration der Integritätsbedingung muss das DBMS natürlich überprüfen, ob die Daten die Bedingung erfüllen. Wenn nicht, muss die Integritätsbedingung abgewiesen werden. Wenn das System die Integritätsbedingung akzeptiert, dann wird im Katalog unter dem Namen Tbuchconpreis die Bedingung registriert.

Integritätsbedingungen können wie folgt klassifiziert werden: **Typconstraint, Spaltenconstraint, Tableconstraint, Datenbankconstraint**. Das System kann sicherstellen, dass alle Integritätsbedingungen erfüllt sind; das heißt: **das System kann die Konsistenz der Daten sichern.** Das System kann aber nicht durchsetzen, dass die Daten korrekt sind, d. h. das System kann nicht sicherstellen, dass die Daten die reale Welt richtig wiederspiegeln.

Schlüsselkandidat

Jede Table hat wenigstens einen Schlüsselkandidaten. Ein Schlüsselkandidat ist eine Spalte bzw. eine minimale (irreduzible) Kombination von Spalten die jede Zeile einer Table eindeutig identifiziert. D. h. für alle Zeiten und für die möglichen wechselnden Inhalte der Table gilt: zwei verschiedene Zeilen in der Table haben auch verschiedene Schlüsselkandidatenwerte.

Primärschlüssel/Alternativschlüssel

Für jede Table wird einer der Schlüsselkandidaten als Primärschlüssel festgelegt. Eventuell vorhandene weitere Schlüsselkandidaten werden als Alternativschlüssel bezeichnet.

Fremdschlüssel

Ein Fremdschlüssel (eine oder mehrere Spalten) darf nur Werte präsentieren, die als Werte eines Primärschlüssels (bzw. Alternativschlüssels) vorhanden sind. Ein Fremdschlüsselwert stellt eine Beziehung her zu der Zeile, die den entsprechenden Primärschlüsselwert bzw. Alternativschlüsselwert enthält.

```
ALTER TABLE Tbuch ADD CONSTRAINT FK_Tbuch_Tverlag
FOREIGN KEY(Verlagnr)
REFERENCES Tverlag(Verlagnr);
```

Eine 1:N Beziehung wird mit Hilfe von Primärschlüssel bzw. Alternativschlüssel und Fremdschlüssel dargestellt. Ein Fremdschlüssel in einer Table T2 ist eine Spalte bzw. eine Spaltenkombination, welche einem Primärschlüssel bzw. Alternativschlüssel in einer anderen Table T1 entspricht und für alle Zeiten gilt, dass für einen aktuellen Inhalt von T2 und T1 jeder Wert im Fremdschlüssel identisch ist mit einem Wert im Primärschlüssel bzw. Alternativschlüssel in einer Zeile von T1.

```
DELETE FROM Tverlag WHERE Verlagnr = 1111;
```

Angenommen, dass diese Anweisung die Zeile für den Verlag 1111 löscht, dass die Datenbank Bücher des Verlags 1111 enthält und dass diese von der Anwendung nicht gelöscht werden. Dann entdeckt das System beim Prüfen der referentiellen Integrität, dass die Integritätsbedingung verletzt ist und weist die Delete-Anweisung ab.

Eine andere Lösung wäre, dass das System alle Bücher des Verlags 1111 automatisch mit löscht. Dieser Effekt kann durch eine geeignete Erweiterung der Fremdschlüsseldefinition erreicht werden.

```
ALTER TABLE Tbuch ADD CONSTRAINT FK_Tbuch_Tverlag
FOREIGN KEY(Verlagnr)
REFERENCES Tverlag(Verlagnr)
ON DELETE CASCADE;
```

Anmerkung zu „Tables ohne Schlüssel“:

Wenn wir sagen, dass jede Table einen Schlüssel besitzt, dann orientieren wir uns in diesem Kapitel und in der ganzen Broschüre am relationalen Datenmodell von Codd.

Die Sprache SQL erlaubt aber das Generieren von „Tables ohne Schlüssel“ die Duplikate zulassen.

Nach unserer Ansicht sollte

- eine Table,
- eine View,
- eine mit Hilfe einer Select-Anweisung abgeleitete Table

immer einen Schlüssel haben und damit nie Duplikate präsentieren.

Empfehlung: **Vermeiden Sie „Tables ohne Schlüssel“!**

Anmerkung zu „NULL-Wert“ bzw. „Null-Marke“:

Wir werden beim Einsatz von Datenbanksystemen mit dem Problem der fehlenden Information konfrontiert (z. B. „Geburtsdatum unbekannt“, „Adresse zur Zeit nicht bekannt“).

Der Zugang zum Problem basiert bei SQL und deshalb bei den meisten kommerziellen Produkten auf der NULL und der dreiwertigen Logik (three-valued logic).

Der Begriff „NULL-Wert“ wird im informellen Sprachgebrauch sehr oft benutzt, NULL ist aber kein Wert. Es wäre besser, den Begriff „Null-Marke“ zu benutzen. **NULL** bedeutet, dass der Wert in der Datenbank nicht definiert ist. Dies kann bedeuten, dass dieser Wert generell nicht existiert, dass er nicht bekannt ist oder dass er noch nicht erfasst ist.

Empfehlung: **Vermeiden Sie NULL!**

Auf der Ebene der Präsentation (logische Ebene) gibt es im relationalen Datenmodell keine physischen Pointer die eine Table mit einer anderen Table verknüpfen.

Der SQL Standard und die so genannten SQL-Datenbanksysteme am Markt erlauben aber auf der Ebene der Präsentation (logische Ebene) physische Pointer!

Anmerkung A: Gibt es auf der Ebene der Präsentation physische Pointer?

Wenn wir sagen, dass es auf der Ebene der Präsentation (logische Ebene) keine physischen Pointer gibt, dann orientieren wir uns am relationalen Datenmodell von Codd. Der zweite große Schnitzer besteht für C.J. Date in seiner Kritik am SQL-Standard und an den so genannten objekt/relationalen Produkten am Markt darin, dass auf der Ebene der Präsentation (logische Ebene) physische Pointer möglich sind.

Der zweite große Schnitzer (The Second Great Blunder):
„The blunder consists of mixing pointers and relations".

Anmerkung B: Ist eine Table dasselbe wie ein Klasse?

Der erste große Schnitzer besteht für C.J. Date in seiner Kritik am SQL-Standard und an den so genannten objekt/relationalen Produkten am Markt darin, dass Relation/Table und Klasse gleichgesetzt werden. Richtig ist aber die Gleichsetzung Klasse = Datentyp.

Der erste große Schnitzer (The First Great Blunder):
falsch: Klasse = Table/Relation
richtig: Klasse = Datentyp

Anmerkung C: Was ist ein Datentyp, was ist eine Domain?

Beachten Sie bitte, dass eine Formulierung wie die folgende schlichtweg falsch ist: „Diese Domänen dürfen nur atomare Werte enthalten, die nicht strukturiert sein dürfen."

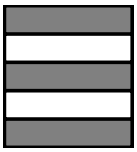
Hier eine korrekte Definition, das Zitat stammt aus The Relational Database Dictionary, O'Reilly 2006:

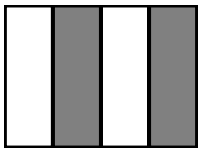
„**type** A named, finite set of values (not to be confused with the internal representation of the values in question, which is an implementation issue). Types can be either scalar or nonscalar (in particular, they can be tuple or relation types); consequently, attributes of relations can also be either scalar or nonscalar…"

Literaturempfehlung:
C. J. DATE An Introduction to Database Systems
Addison-Wesley Eighth Edition 2003

1.7 Relationale Algebra

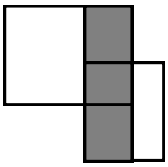
Selektion **Projektion** **Produkt**

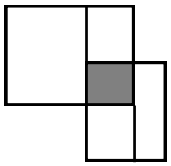


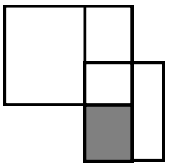


a b c	*	x y	=	a a b b c c		x y x y x y

Vereinigung UNION **Durchschnitt INTERSECT** **Differenz EXCEPT** **Division**







a a a b c		x y z x y	:	x z	=	a

a b c	8 8 12	**JOIN**	8 9 12	5,50 7,70 6,60	=	a b c	8 8 12	5,50 5,50 6,60

die Summarize-Operation

tab1

8 9 12	

tab2

8 8 12	a b c

tab1 summarize tab2 mit COUNT

8 9 12	2 0 1

Die relationale Algebra besteht aus einer Menge von mathematisch exakt definierten Operatoren, mit deren Hilfe aus Relationen neue, andere Relationen abgeleitet werden.

- Die **Restrikt-Operation** (auch Restriktion bzw. Selektion genannt): **Auswahl von Zeilen.**
- Die **Projekt-Operation** (auch Projektion genannt): **Auswahl von Spalten und entfernen von Duplikat-Zeilen** (mehrfach vorhandene Zeilen werden, sofern sie nach dem Entfernen von Spalten auftreten, bis auf je eine gestrichen. Die Projekt-Operation ist das algebraische Gegenstück zum „existential quantifier exists“ des Relationenkalküls.
- union, intersection, difference
- cartesian product
- Die **Divide-Operation:** Die Division ist, salopp gesagt, das algebraische Gegenstück zum „universal quantifier forall“ des Relationenkalküls.
- Die Join-Operation (salopp auch Join bzw. Verbund genannt): Verbinden von zwei Relationen auf der Basis gemeinsamer Werte.
- Die Summarize-Operation

Die erste Version der Algebra wurde von Codd definiert, sie enthielt acht Operatoren (die traditionellen Mengenoperatoren union, intersection, difference, und cartesian product und die speziellen relationalen Operatoren restrict, project, join, und divide). Später kam die Summarize-Operation dazu.

Das Ergebnis einer Operation der relationalen Algebra ist wieder eine Relation.

Die Sprache SQL mit ihren TABLEs, VIEWs und ihren SELECT-Anweisungen hält sich nicht an dieses mathematische Modell und ermöglicht SELECT-Anweisungen, mit deren Hilfe Listen mit Duplikaten generiert werden können!

- JOIN, UNION, EXCEPT, INTERSECT sind syntaktische Elemente des SQL-Standards.
- INTERSECT kann im SQL-Umfeld, sofern keine NULL im Spiele ist auf einfache Weise durch EXISTS ersetzt werden.
- **Der SQL-Dialekt von Oracle versteht MINUS, aber nicht den Operator EXCEPT.** EXCEPT bzw. bei Oracle MINUS kann im SQL-Umfeld, sofern keine NULL im Spiele ist auf einfache Weise durch NOT EXISTS ersetzt werden.
- SQL unterstützt außerdem korrelierte und nichtkorrelierte Subqueries (z.B. IN, EXISTS, …) und verschiedene Arten von Join (INNER, LEFT OUTER, RIGHT OUTER, FULL OUTER).
- Die Division kann auch auf verschiedene Weise realisiert werden.
- Die Aggregatfunktionen der Sprache SQL sind SUM, AVG, MAX, MIN und COUNT. Mit Hilfe der GROUP BY-Klausel kann eine Table in Gruppen aufgeteilt werden, so dass eine Aggregatfunktion wie zum Beispiel AVG auf jede Gruppe einzeln angewendet wird.
- Die Summarize-Operation der relationalen Algebra kann im Rahmen der Sprache SQL mit Hilfe von GROUP BY und LEFT OUTER JOIN bzw. korrelierten Subqueries realisiert werden.
- Jüngeren Datums sind die Sprachelemente CROSS APPLY und OUTER APPLY (unterstützt von SQL Server und Oracle) bzw. INNER JOIN TABLE und LEFT OUTER JOIN TABLE (unterstützt von DB2).

1.8 Was ist eine Projektion?

In den Manuals vieler Hersteller, in Büchern über Produkte am Markt, in den meisten Büchern über SQL, in den kostenlosen SQL-Kursen im Internet, überall finden Sie syntaktisch korrekte SELECT-Anweisungen die Listen mit Duplikaten generieren!

Die folgende SELECT-Anweisung ist syntaktisch korrekt, sie implementiert aber keine relationale Projektion und liefert eine Liste mit Duplikaten (Erschj und Preis zusammen sind nicht eindeutig). Das Ergebnis ist keine Table/Relation.

```
SELECT
Erschj, Preis FROM Tbuch
;

SELECT ALL
Erschj, Preis FROM Tbuch
;

ERSCHJ     PREIS
------ ---------
  1977      0.50
  1988      3.50
  1988     20.50
  1989     99.99
  1989     99.99
  1989      NULL
  1990     55.00
  1990      NULL
  NULL     99.99
  NULL      NULL
  NULL      NULL

11 rows selected.
```

Die folgende SELECT-Anweisung ist syntaktisch korrekt, sie implementiert eine relationale Projektion und liefert eine Liste ohne Duplikate. Das Ergebnis ist eine Relation/Table und gibt eine Antwort auf folgende Frage: Welche Ausprägungen von Erschj und Preis gibt es?

```
SELECT DISTINCT
Erschj, Preis FROM Tbuch
;

ERSCHJ     PREIS
------ ---------
  1977      0.50
  1988      3.50
  1988     20.50
  1989     99.99
  1989      NULL
  1990     55.00
  1990      NULL
  NULL     99.99
  NULL      NULL

9 rows selected.
```

1.9 SQL die Sprache

Datendefinition

- CREATE
- ALTER
- DROP

Datenmanipulation

- SELECT
- INSERT
- UPDATE
- DELETE

Datenkontrolle

- GRANT
- REVOKE

Transaktionsverarbeitung

- COMMIT
- ROLLBACK
- SAVEPOINT
- Der Isolation Level

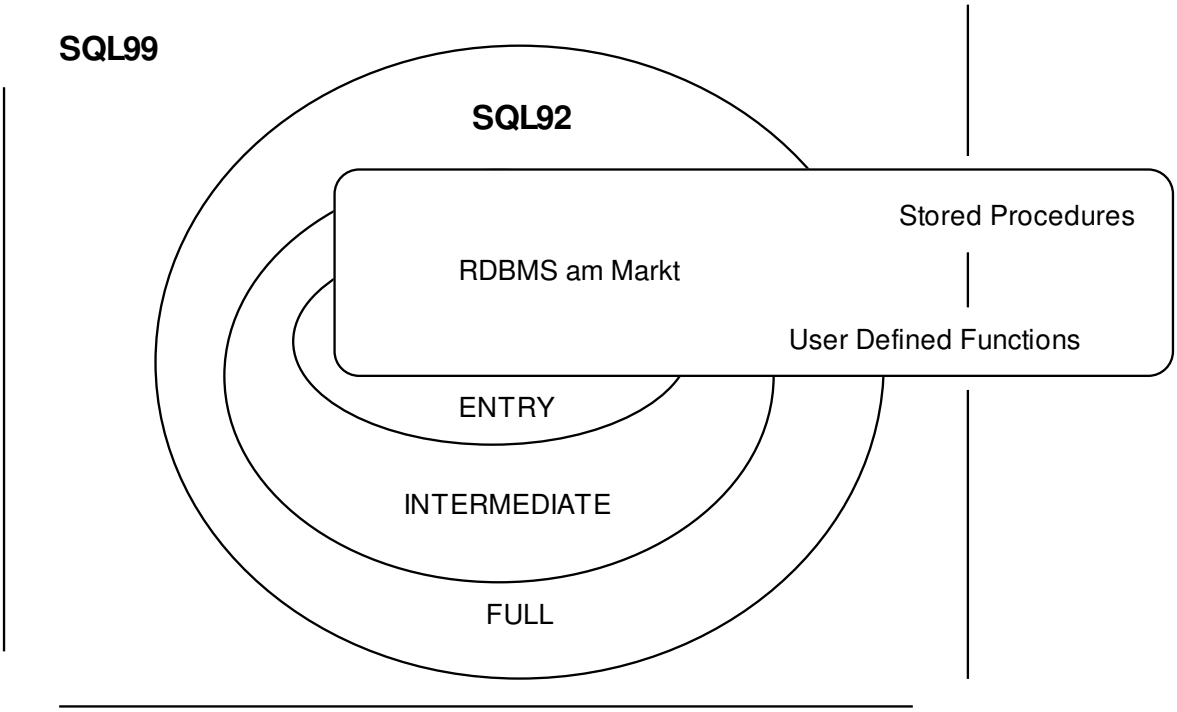
Weitere Sprachelemente für die Anwendungsentwicklung

- Embedded Static SQL
- Embedded Dynamic SQL
- Call Level Interface
- Returncode, SQLCODE, SQLSTATE

Proprietäre Sprachelemente

- Explizites Locking LOCK TABLE
- Information über die Zugriffpfade EXPLAIN

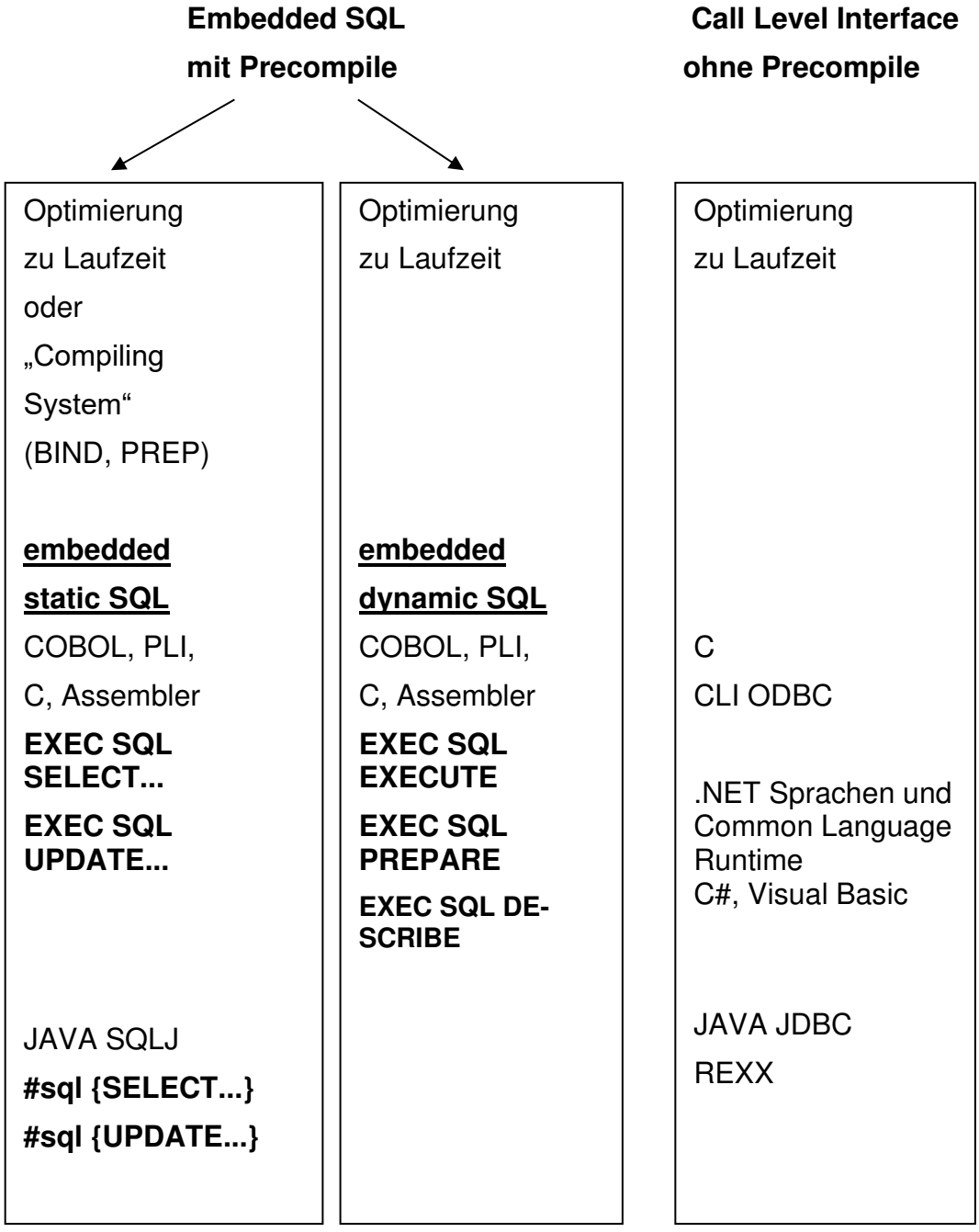
1.10 SQL und die Dialekte der Implementierungen



Ein großes Problem besteht darin, dass die Produkte am Markt Sprachelemente implementiert haben, **die im SQL Standard anders spezifiziert sind oder gar nicht Bestandteil des Standards sind**, in den bestehenden Anwendungen praktisch genutzt werden und von den anderen Produkten zur Zeit oder auch in Zukunft nicht unterstützt werden.

Man bezeichnet im **IT-Bereich** traditionell solche Dateiformate, Protokolle usw. aber auch Hardware als **proprietär**, die nicht allgemein anerkannten Standards entsprechen, also sozusagen „hauseigene“ Entwicklungen sind.

1.11 SQL Schnittstellen



Für die Verwendung von SQL in Verbindung mit Programmiersprachen wie JAVA, COBOL, Assembler usw. gibt es zwei Techniken.

- Eine herkömmliche Programmierschnittstelle (API Application Programming Interface) erlaubt die direkte Übergabe von SQL-Anweisungen an ein Datenbanksystem über Funktionsaufrufe, dabei ist kein Precompiler erforderlich. SQL-Anweisungen, die mittels **CLI/ODBC** bzw. **JDBC** ausgeführt werden, müssen zu Laufzeit interpretiert und optimiert werden.
- Bei **eingebetteter SQL** (engl. embedded SQL) wird ein Anwendungsprogramm im Quelltext mit gekennzeichneten SQL-Anweisungen erstellt (Beispiele: Embedded SQL im COBOL-Code, SQLJ im Java-Code). Während der Programmvorbereitung übersetzt ein **Precompiler** die SQL-Anweisungen in Funktionsaufrufe.

Bei **eingebetteter SQL** wird zwischen **statischer und dynamischer SQL** unterschieden.

- Bei **eingebetteter dynamischer SQL** ist die SQL-Anweisung dem System erst zum Zeitpunkt der Programmausführung bekannt. Das Datenbanksystem, d. h. der Optimizer muss die SQL-Anweisung zu Laufzeit des Programms interpretieren und den Zugriffspfad optimieren (Interpreter).
- Bei **eingebetteter statischer SQL** ist die SQL-Anweisung zum Zeitpunkt der Programmübersetzung bekannt und festgelegt (z. B. wenn die Abfrage eines Kontos vorformuliert ist und zur Laufzeit nur die Kontonummer eingesetzt wird). Deshalb kann schon bei der Übersetzung des Programms d.h. beim so genannten **BIND** der optimale Zugriffspfad festgelegt werden (z. B. Zugriff auf die Daten einer Kontonummer über einen Index). Eine Implementierung der BIND-Philosophie (Compiling-System) ist heute im IBM DB2 Umfeld vorhanden, die meisten anderen Datenbanksysteme optimieren hingegen zu Laufzeit (Interpreter).

SQL/PSM ist ein ISO Standard, der SQL um **prozedurale Sprachelemente** erweitert (WHILE, IF…ELSE, Definition von Variablen, Fehlerbehandlung usw.).

- **DB2** hat eine Teilmenge dieser Erweiterung implementiert und verwendet dafür den Namen SQL PL (SQL Procedural Language).
- **Oracle** vermarktet eine proprietäre Implementierung unter dem Namen *PL/SQL*.
- Bei **SQL Server** und **Sybase** heißt der proprietäre Dialekt Transact SQL.

1.12 Datenbankmanagementsysteme

1.12.1 Codd 12 Regeln

1985 veröffentlichte E. F. Codd 12 Regeln, denen ein Datenbankmanagementsystem genügen muss, damit es zu Recht als "relational" bezeichnet werden kann.

Regel Zero (foundation rule)
Ein relationales Datenbankmanagementsystem muss Datenbanken allein durch seine relationalen Fähigkeiten vollständig verwalten können.

Informationsregel (information rule)
Die gesamte Information in einer relationalen Datenbank wird explizit auf genau eine Weise dargestellt, nämlich durch Werte in Spaltenpositionen innerhalb von Zeilen von Tabellen.

Garantierter Zugriff (guaranteed access rule)
Auf jeden einzelnen Wert in einer relationalen Datenbank kann mit einer Kombination aus Tabellenname, Primärschlüsselwert und Spaltenname zugegriffen werden (logically addressable).

Systematische Behandlung von ‚NULL-Werten' (systematic treatment of null values)
‚NULL-Werte' werden systematisch und datentypunabhängig zur Darstellung fehlender Information (missing information and inapplicable information) unterstützt. ‚NULL-Werte' sind weder leere Zeichenketten oder Ketten von Leerzeichen, noch Zero oder irgendeine andere Zahl.

Dynamischer Online Katalog basierend auf dem relationalen Modell (dynamic online catalog based on the relational model)
Die Datenbankbeschreibung wird auf die gleiche Weise wie gewöhnliche Daten dargestellt, so dass autorisierte Benutzer sie mit derselben relationalen Sprache abfragen können, die sie auch auf gewöhnliche Daten anwenden.

Umfassende Daten-Sub-Sprache (comprehensive data sublanguage rule)
Ein relationales System darf mehrere Sprachen und verschiedene Terminal-Modi (z.B. Formular-Ausfüll-Modus) unterstützen. Es muss jedoch mindestens eine Sprache geben, deren Anweisungen – einer wohldefinierten Syntax entsprechend – als Zeichenketten ausgedrückt werden können. Sie muss die folgenden Punkte berücksichtigen:

- Daten- und Datensichtdefinitionen
- Datenmanipulation (interaktiv und durch Programme)
- Integritätsbedingungen und Autorisierung
- Transaktionsgrenzen

Datenänderungen durch Datensichten (view updating rule)
Alle Datensichten (Views), die theoretisch Datenänderungen zulassen, können auch durch das System geändert werden (all views that are theoretically updatable must be updatable by the system).

Mengenorientiertes Einfügen, Ändern und Löschen (high-level insert, update, and delete)
Das System muss mengenorientierte Einfüge-, Änderungs- und Lösch-Operatoren unterstützen (the system must support set-at-a-time insert, update, and delete operators).

Physische Datenunabhängigkeit (physical data independence)
Benutzer und Anwendungsprogramme bleiben unbeeinträchtigt, wenn Veränderungen an den Speicherstrukturen oder an den Zugriffsmethoden vorgenommen werden (user and user programs are immune to change in storage structure and access technique).

Logische Datenunabhängigkeit (logical data independence)
Benutzer und Anwendungsprogramme bleiben unbeeinträchtigt, wenn Veränderungen an der logischen Struktur einer Datenbank vorgenommen werden (user and user programs are immune to change in the logical structure of the database) .

Integritätsunabhängigkeit (integrity independence)
Integritätsbedingungen können in der relationalen Daten-Sub-Sprache definiert und im Katalog, getrennt von den Anwendungsprogrammen, gespeichert werden.

Verteilungsunabhängigkeit (distribution independence)
Existierende Programme funktionieren weiter erfolgreich unabhängig davon ob a) eine verteilte Version des DBMS eingeführt wird oder b) existierende verteilte Daten neu verteilt werden.

Unterwanderungsverbot (nonsubversion rule)
Falls ein System über eine niedere (satzorientierte) Sprache verfügt, kann diese nicht benutzt werden, um die Datenschutz- bzw. Integritätsbedingungen (relational security or integrity constraints) zu verletzen oder zu umgehen, die in der höheren (mengenorientierten) Sprache formuliert worden sind.

Sofern die gängigen Produkte am Markt an diesen 12 Regeln von Codd gemessen werden, darf sich keines dieser Produkte als relational bezeichnen! Oft werden diese Produkte am Markt deshalb als SQL-Datenbankmanagementsysteme bezeichnet!

Beachten Sie bitte, dass zum Beispiel die Forderung nach **‚Physischer Datenunabhängigkeit'** von den Markführern (den SQL-Datenbanksystemen DB2, ORACLE und SQL Server) **nur teilweise erfüllt** wird.

1.12.2 Warum Datenbanksysteme?

Ein Datenbanksystem bietet dem Unternehmen eine zentralisierte Kontrolle über die Daten.

- Inkonsistenzen können vermieden werden (inconsistency can be avoided (to some extend)).
- Die Integrität der Daten kann aufrechterhalten werden (integrity can be maintained).
- Redundanzen können eingeschränkt und kontrolliert werden (redundancy can be reduced).
- Transaktionsunterstützung kann gewährleistet werden (transaction support can be provided).
- Datenschutz kann durchgesetzt werden (security can be enforced).
- Die Daten können von verschiedenen Benutzern gleichzeitig benutzt werden (the data can be shared).
- Standards können durchgesetzt werden (standards can be enforced).
- Die verschiedensten Anforderungen können ausbalanciert werden (conflicting requirements can be balanced).
- Datenunabhängigkeit kann gewährleistet werden (data independence can be provided).

C. J. DATE An Introduction to Database Systems
Addison-Wesley Publishing Company
Seventh Edition 2000

Business Rules der realen Welt können im Datenbanksystem definiert werden. Bei allen Änderungen der Daten wird gegen diese **Integritätsbedingungen** (Integrity Constraints) geprüft. Daten sind **konsistent** (was das System betrifft), wenn sie alle im System definierten Integritätsbedingungen erfüllen.

Korrekt sind die Daten, wenn Sie auch die reale Welt richtig widerspiegeln.

Unter dem Begriff **Integrität** der Datenbank werden alle Fragen zusammengefasst, die im weitesten Sinne mit der Erhaltung **der Korrektheit der Daten** in der Datenbank zu tun haben.

Angenommen, der Preis eines Buches ist in der Datenbank zweimal, also redundant gespeichert. Wenn die beiden Einträge nicht übereinstimmen, dann ist einer der beiden Preise auf jeden Fall nicht korrekt, die Datenbank ist in einem so genannten inkonsistenten Zustand.

Eine Datenbank, die sich in einem inkonsistenten Zustand befindet präsentiert ihren Benutzern möglicherweise nicht korrekte oder sich widersprechende Informationen.

Angenommen, die **Redundanz** wird kontrolliert (indem Sie dem System bekannt gemacht wird). Dann kann das System garantieren, dass die Datenbank aus Benutzersicht nie inkonsistent ist, indem das System das Ändern eines Wertes automatisch auch beim anderen Wert durchführt (dieser Vorgang ist als propagating updates bekannt).

Inkonsistenz in der Datenbank ist ein Beispiel für Mangel von **Integrität**.

Eine **Transaktion** ist eine logische Verarbeitungseinheit (logical unit of work, unit of recovery). Die Daten einer Datenbank werden von einem konsistenten Zustand in einen anderen konsistenten Zustand überführt.

Beginn Transaktion

Girokonto +100

Sparkonto -100

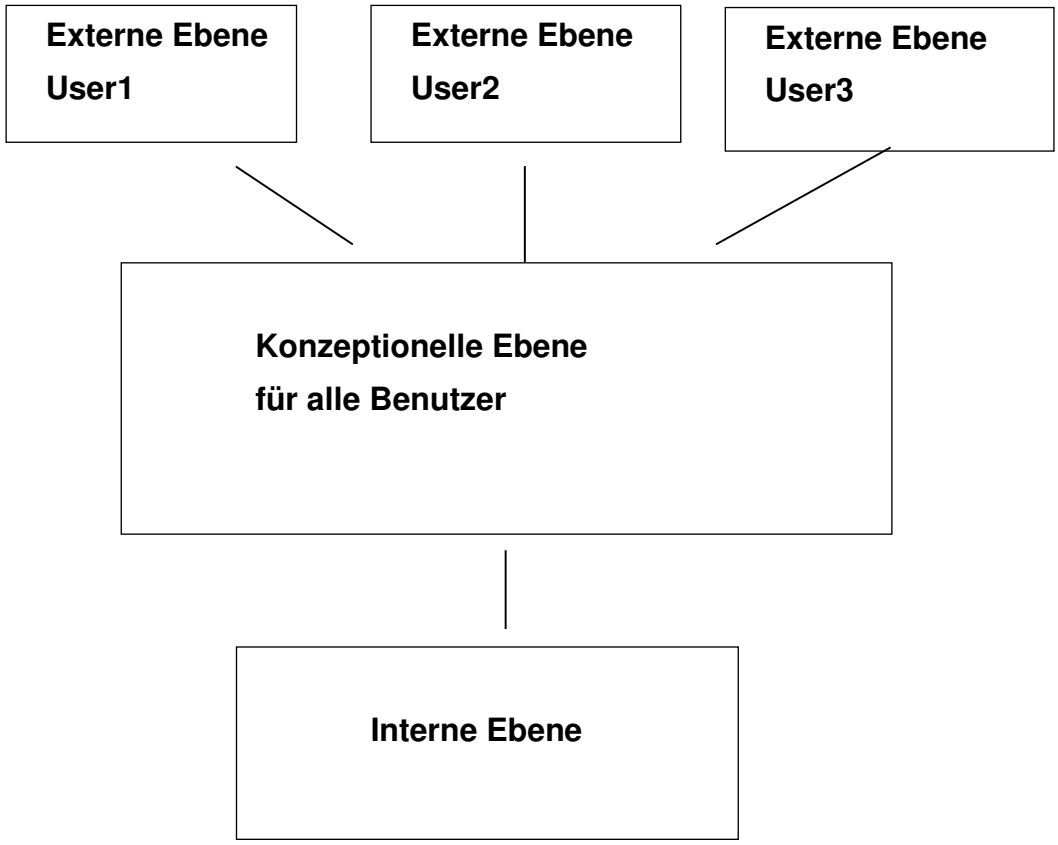
Ende Transaktion

Bei einer Umbuchungstransaktion müssen alle Veränderungen (Update-Operationen) oder keine durchgeführt werden.

Sofern das System fehlerfrei arbeitet sind die Daten am Ende einer Transaktion in einem konsistenten Zustand!

Datenunabhängigkeit (physische und logische) wird im Rahmen der 3-Ebenen-Architektur für ein Datenbankmanagementsystem diskutiert.

1.12.3 3-Ebenen-Architektur für ein Datenbankmanagementsystem



Konzeptionelle Ebene/Logische Ebene für alle Benutzer

Die konzeptionelle Ebene präsentiert alle Daten der Datenbank.

Interne Ebene/Physische Ebene Auf der internen Ebene wird mit Hilfe einer geeigneten Struktur festgehalten, wie die Daten zu speichern sind. Es versteht sich, dass dabei auf das konzeptionellen Datenmodell Bezug zu nehmen ist, wobei gleichzeitig auch die Besonderheiten des verfügbaren Datenbankmanagementsystems berücksichtigt werden müssen.

C.J.DATE beschreibt in seinem Buch

An Introduction to Database Systems
C.J. Date Eighth Edition 2004

die Interne Ebene wie folgt:

„The internal view is a low-level representation of the entire database; it consists of many occurrences of many types of internal record... Internal record is the ANSI/SPARC term for the construct that we have been calling stored record … The internal view is thus still at one remove from the physical level, since it does not deal in terms of physical records – also called blocks or pages - nor with any device-specific considerations such as cylinder or track sizes. In other words, the internal view effectively assumes an unbounded linear address space; details of how that address space is mapped to physical storage are highly system-specific and are deliberately omitted from the general architecture.

Note: In case you are not familiar with the term, we should explain that the block, or page, is the unit of I/O – that is, it is the amount of data transferred between secondary storage and main memory in a single I/O operation“

Externe Ebene/Logische Ebene für einen Benutzer

Die externe Ebene ist die Ebene des einzelnen Benutzers (Anwendungsprogramm, Endanwender, User). Sie kümmert sich darum, wie die Daten einem individuellen Benutzer präsentiert werden.

Es gibt zwei Arten von Datenunabhängigkeit, physische und logische.

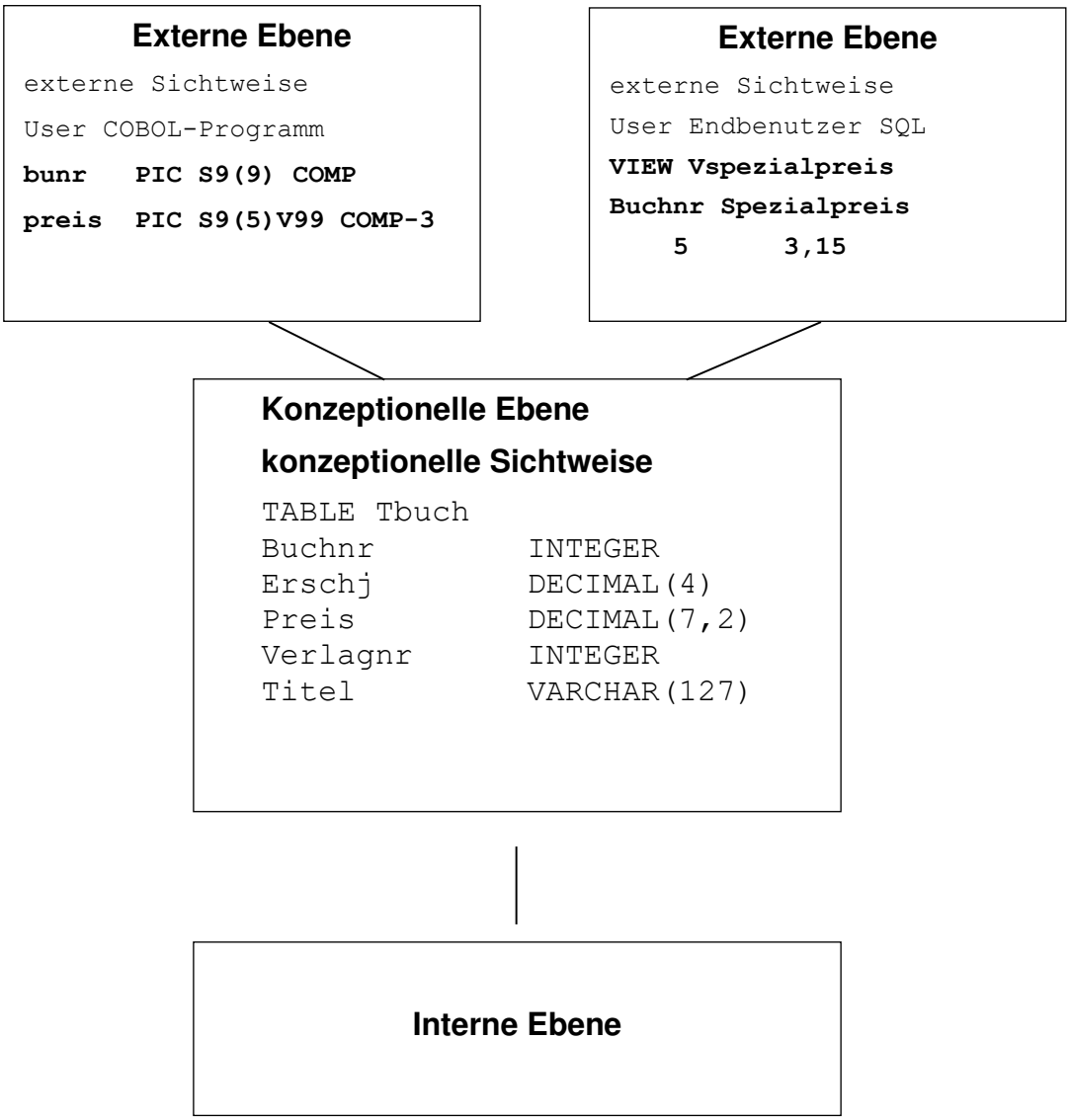
- **Physische Datenunabhängigkeit** Anwender und Anwendungsprogramme bleiben unbeeinträchtigt, wenn auf der internen Ebene Veränderungen an der physischen Darstellung der Daten oder an den Zugriffsmethoden vorgenommen werden.
- **Logische Datenunabhängigkeit** Anwender und Anwendungsprogramme bleiben unbeeinträchtigt, wenn Veränderungen (Erweiterung bzw. Restrukturierung) an der logischen Struktur der Datenbank auf der konzeptionellen Ebene vorgenommen werden.

Der Schlüssel für Physische Datenunabhängigkeit ist das Mapping zwischen der Konzeptionellen Ebene und der Internen Ebene.

Der Schlüssel für Logische Datenunabhängigkeit ist das Mapping zwischen der Konzeptionellen Ebene und der Externen Ebene.

Beachten Sie bitte, dass die Forderung nach ‚Physischer Datenunabhängigkeit' von den Markführern (den SQL-Datenbanksystemen DB2, ORACLE und SQL Server) nur teilweise erfüllt wird.

1.12.4 3-Ebenen-Architektur und relationale Datenbankmanagementsysteme



Es ist für das Verständnis hilfreich, an einem einfachen Beispiel darzustellen, wie in einem **relationalen System** die drei Ebenen typischerweise realisiert werden.

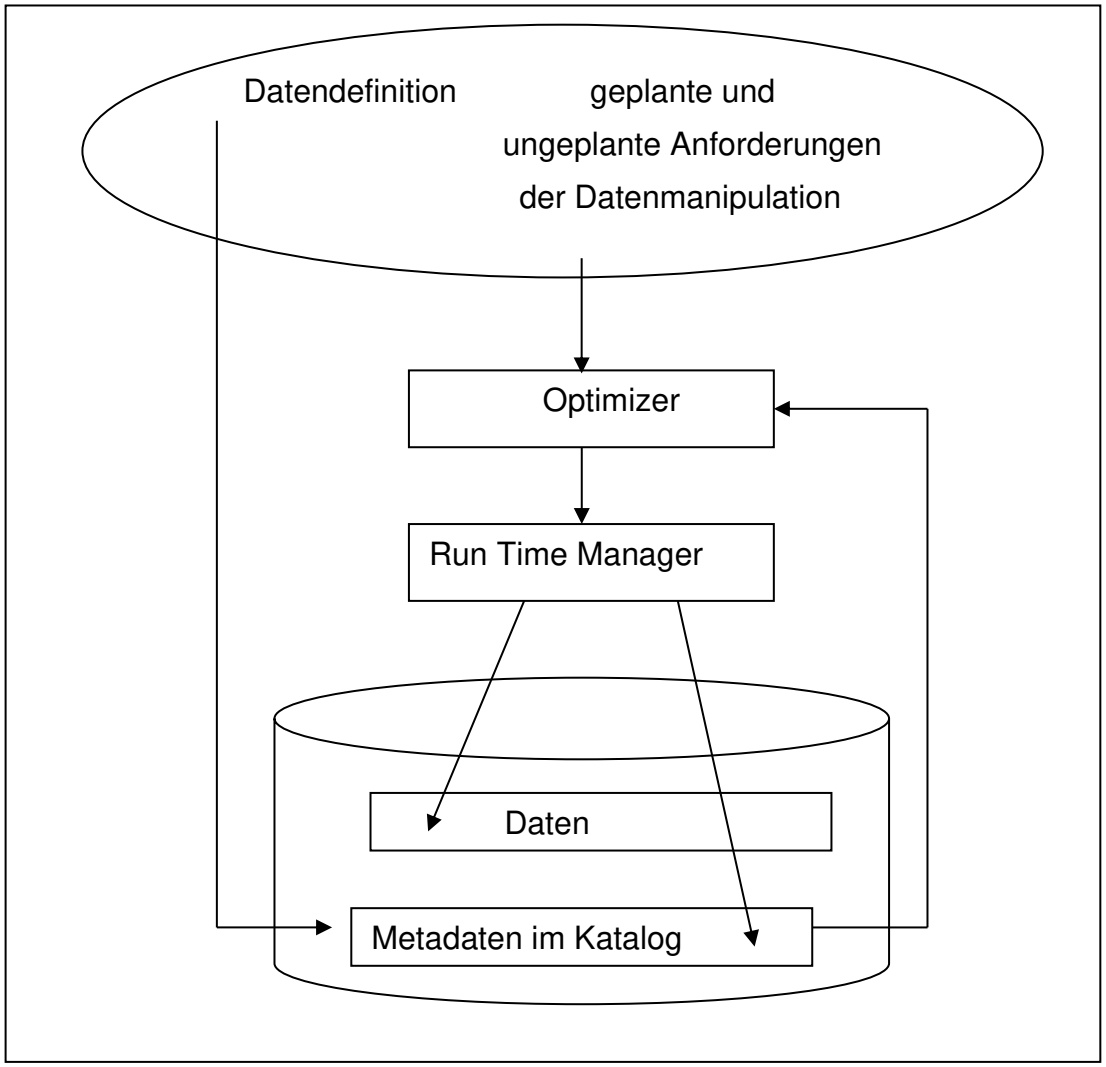
Auf der **Konzeptionellen Ebene präsentiert** die Datenbank Daten über Bücher. Jedes einzelne Buch hat buchnr, erschj, preis, verlagnr, titel. Die Daten werden präsentiert als Zeilen einer Table.

Auf der **Internen Ebene** können die Zeilen der Table Tbuch zum Beispiel als Sätze in einer Datei gespeichert sein. Die physische Speicherung der Daten kann aber auch in anderer Weise erfolgen. Eine Zeile aus Tbuch muss nicht unbedingt direkt einem physischen Satz entsprechen.

Auf der **Externen Ebene** hat der COBOL-User eine externe Sicht der Daten. Jedes Buch wird repräsentiert durch einen COBOL-Satz. Im Beispiel interessiert sich der COBOL-User nur für die Buchnummer bunr und den Preis preis.

Auf der **Externen Ebene** hat der Endbenutzer mit Hilfe einer
VIEW Vspezialpreis Zugriff auf Daten, er interessiert sich zum Beispiel nur für die Buchnummer buchnr und den Spezialpreis = Preis*0.90

1.12.5 Funktionen des Datenbankmanagementsystems



- Datendefinition
- Metadaten im Katalog
- Datenmanipulation (geplant, ungeplant)
- Interpreter, Compiler
- Optimierung zu Laufzeit, Optimierung beim Compile
- Datenschutz, Security
- Datenintegrität (Integritätsbedingungen, Integrity Constraints)
- Transaktionsverarbeitung (Concurrency Control, Recovery)
- Performance

Das Datenbankmanagementsystem (DBMS) kontrolliert den Zugang zur Datenbank, es stellt die **Schnittstelle für den Benutzer** (das User Interface) bereit.

Anforderungen der Datenmanipulation (Select, Insert, Update, Delete) können geplant oder ungeplant (ad hoc) sein. Ungeplante Anforderungen sind charakteristisch für Decision Support Anwendungen. Geplante Anforderungen sind charakteristisch für Operationale Anwendungen.

Ungeplante Anforderungen benötigen einen **Interpreter**, die **Optimierung des Zugriffs** erfolgt zu Laufzeit. Geplante Anforderungen können schon beim Umwandeln des Programms (**Compiler**) optimiert werden.

Das DBMS weist alle Versuche zurück, die vom Datenbankadministrator definierten Integritätsbedingungen und Berechtigungen zu verletzen.

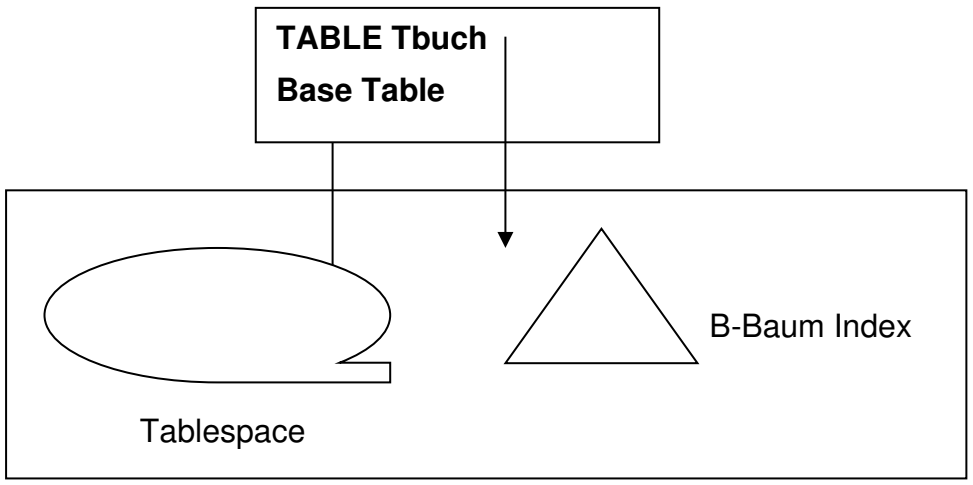
Ein **Transaktionsmanager** (TP-Monitor) muss die Daten gegen Verlust und Beschädigung schützen (z. B. bei Absturz des Systems, bei Absturz eines Programms) und konkurrierende Zugriffe mehrerer Programme kontrollieren.

Das DBMS sollte alle Aufgaben natürlich möglichst effizient ausführen.

1.12.6 „Was und wie" – SELECT und der Optimizer

```
SELECT Buchnr, Titel
FROM Tbuch
WHERE Buchnr = 5;
```

```
SELECT Buchnr, Preis
FROM Tbuch
WHERE Erschj = 1990;
```



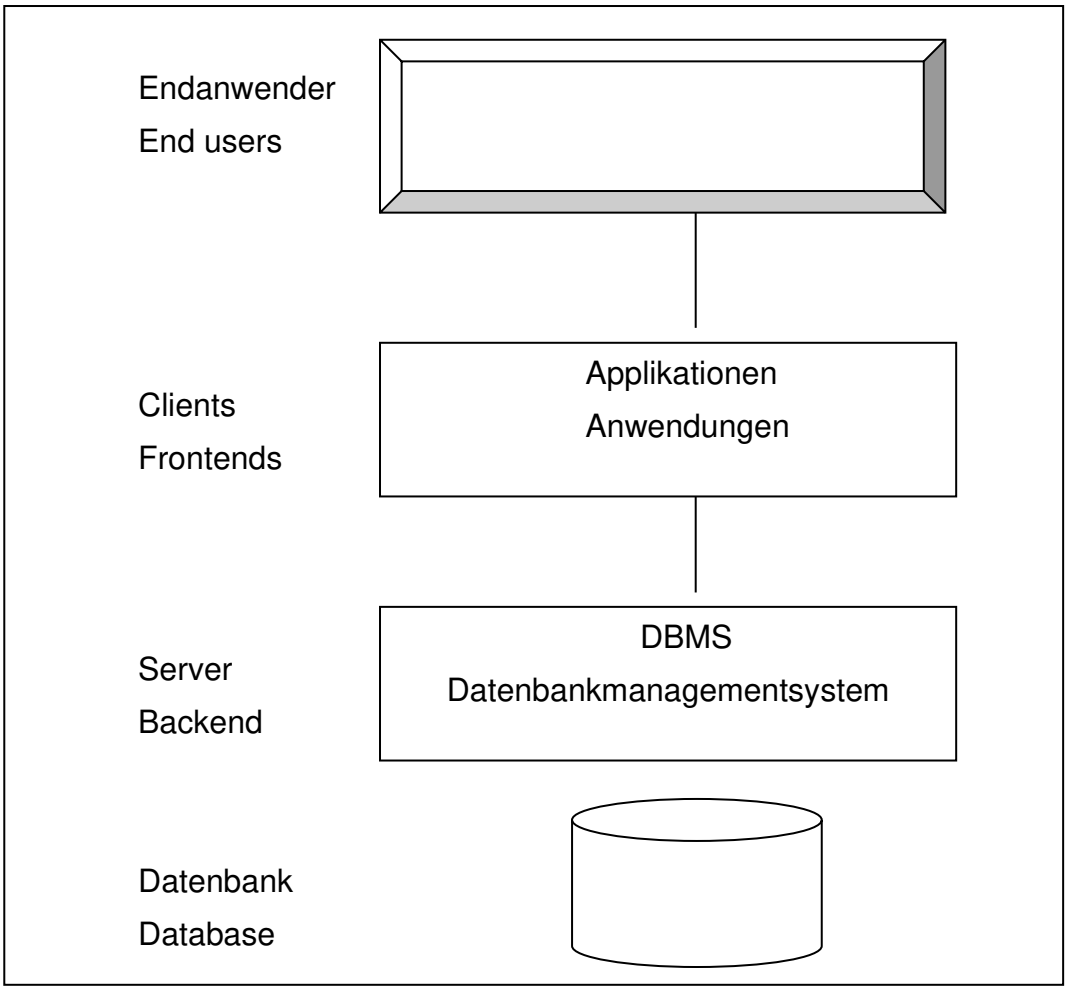
Informationen über die Objekte einer Datenbank werden im Katalog gespeichert.

Wir nehmen an, dass die Zeilen der Base Table physisch gespeichert sind als Sätze in einer Datei (Tablespace) und dass das System in der Lage ist, diese Datei sequentiell zu lesen. Zusätzlich nehmen wir an, dass auf dem Primärschlüssel ein B-Baum Index generiert wurde. Die meisten Systeme am Markt unterstützen diese Art der physischen Speicherung der Daten.

- Bei einem SELECT **(was**) kann sich der Optimizer für einen **sequentiellen Zugriff (wie)** oder – falls ein geeigneter Index vorhanden ist – für einen **Indexzugriff (wie)** entscheiden.
- Ein Datenbanksystem verwendet Indizes zur Optimierung von Zugriffen. Ob ein Index verwendet wird, oder welcher Index verwendet wird, muss nicht explizit in der SELECT-Anweisung angegeben werden. **Der Optimizer entscheidet.**
- Der Optimizer ermittelt aufgrund von **Statistikdaten im Katalog** den "schnellsten" Zugriffspfad. Dazu benötigt der Optimizer möglichst aktuelle Statistikdaten. Sofern das Datenbanksystem die Statistikdaten (bei INSERTs, UPDATEs und DELETEs) nicht automatisch aktualisiert können sie mit Hilfe von Hilfsprogrammen/Utilities (z. B. ORACLE ANALYZE, DB2 RUNSTATS) aktualisiert werden.
- Die Anwendungsentwicklung und Datenbankadministration muss aber die Zugriffpfade mit Hilfe der **EXPLAIN-Funktion** überprüfen. Nur so besteht die Gewissheit, dass z. B. ein vorhandener Index auch wirklich für den Zugriff verwendet wird.

1.13 Client/Server Architektur

1.13.1 Frontend, Backend, Database



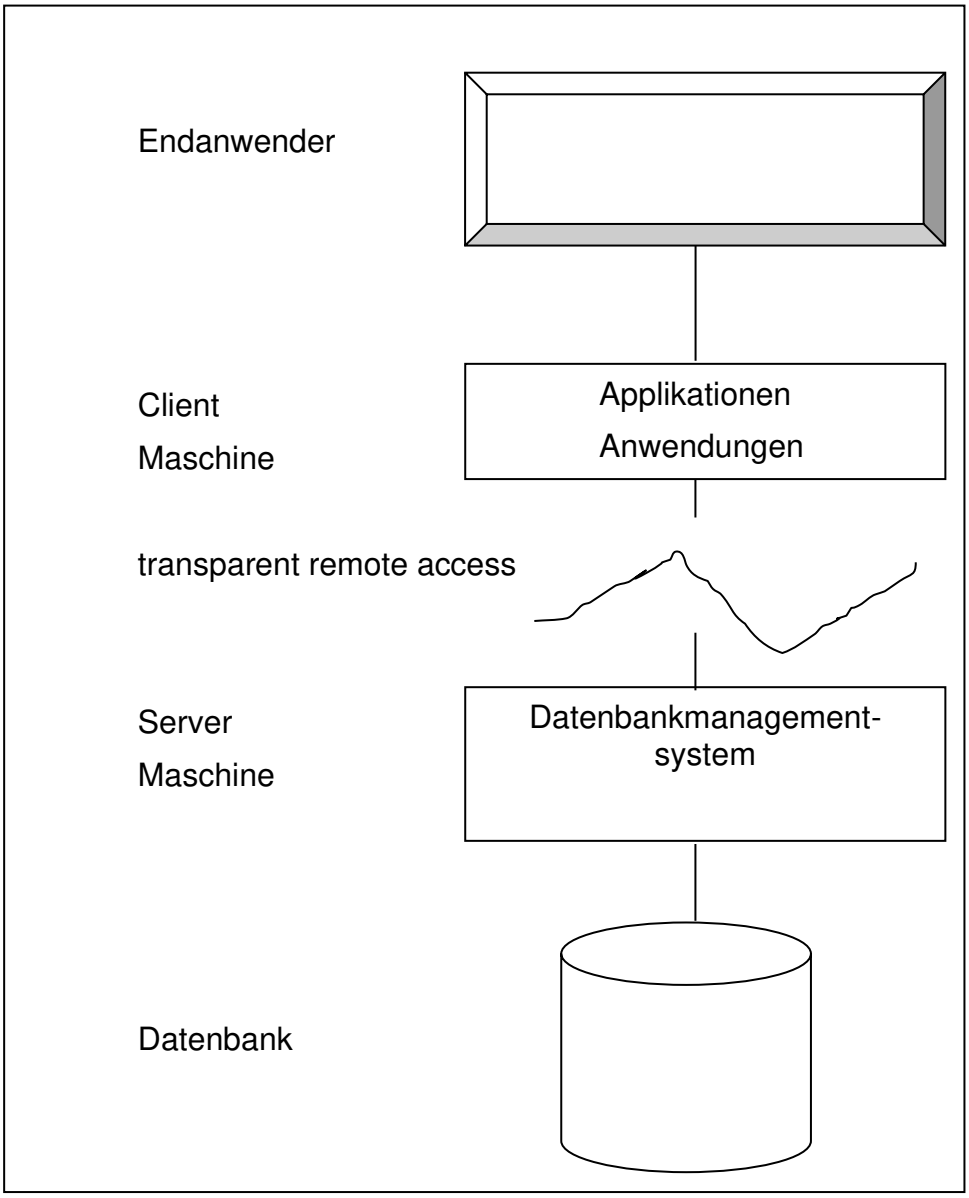
Wir können ein Datenbanksystem auch aus einer etwas anderen Perspektive, aus der Sicht der Anwendungen betrachten.

Von einem high-level Standpunkt aus besteht dann ein Datenbanksystem aus zwei Teilen, aus dem Server (auch Backend genannt) und einer Menge von Clients (auch Frontends genannt).

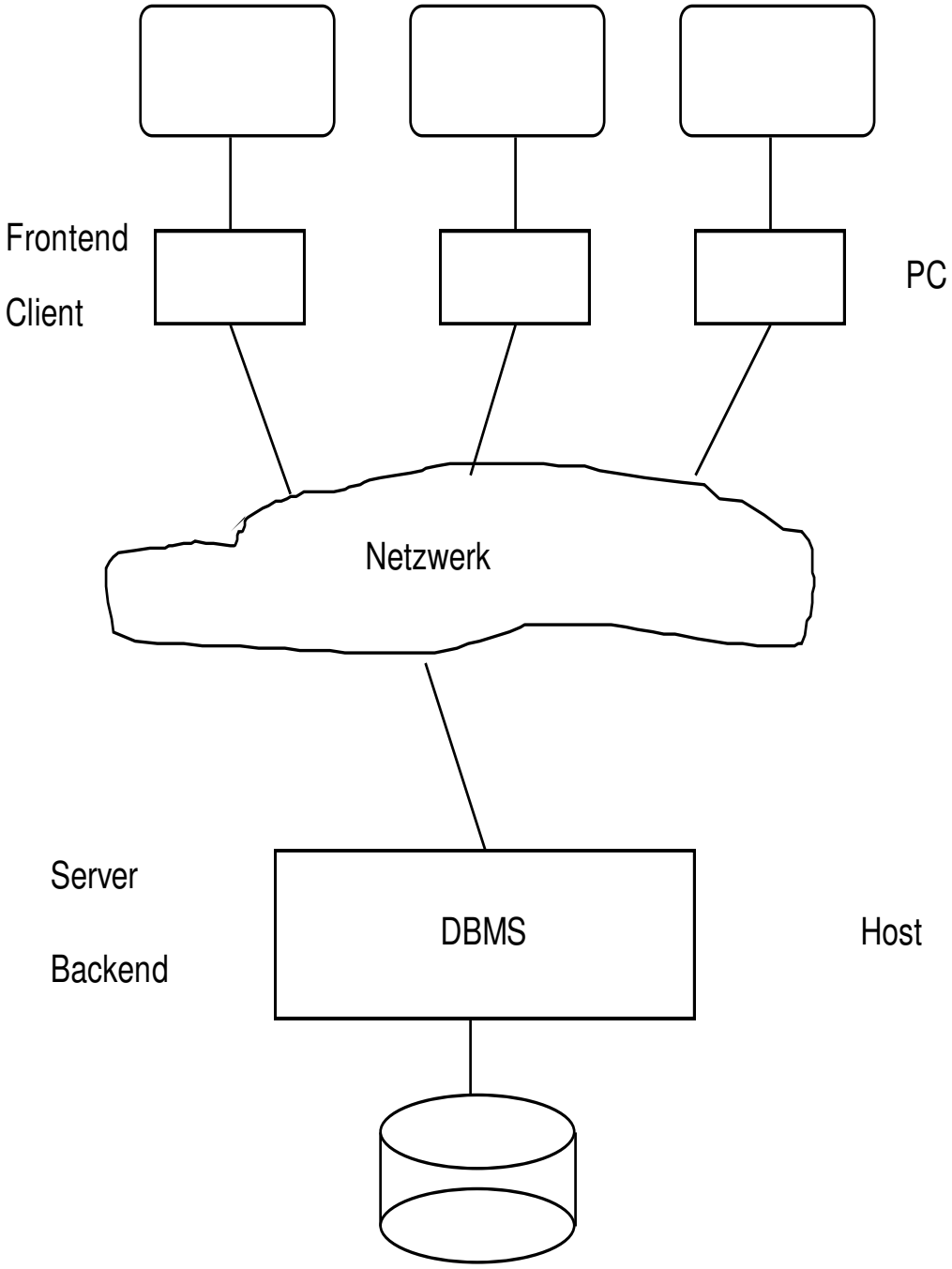
Der Server ist gerade das DBMS. Er unterstützt alle Funktionen eines Datenbanksystems – Datendefinition, Datenmanipulation, Datenschutz, Integrität usw.

Die Clients sind Anwendungen. Diese Anwendungen sind entweder selber geschrieben (user-written) oder im Server eingebaut (builtin) oder von einem sonstigen Anbieter gekauft (z.B. Tools).

1.13.2 Verteilte Verarbeitung – Distributed Processing

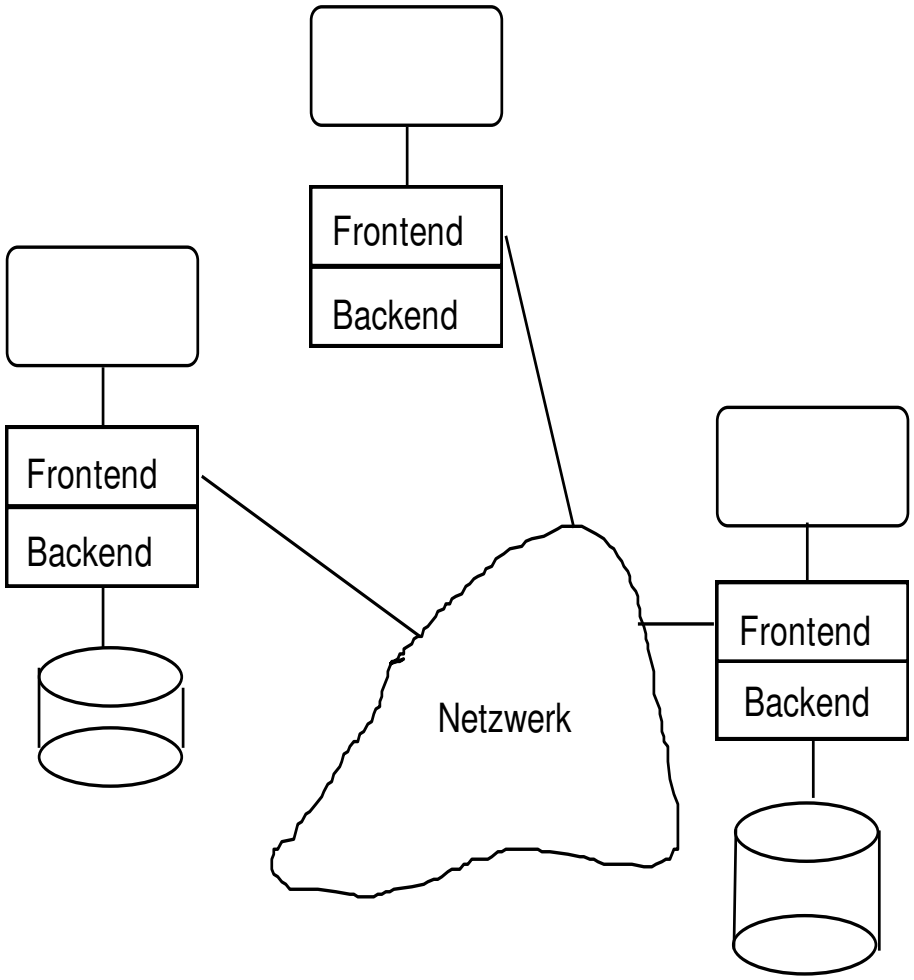


1.13.3 Eine Server Maschine, mehrere Client Maschinen



1.13.4 „Eine einzige Datenbank“

Eine Anwendung (Frontend) kann auf verschiedene Datenbanken auf verschiedenen Rechnern (Backend) zugreifen. Aus logischer Sicht "eine einzige Datenbank" (distributed database system).



2

SELECT mit einer TABLE

2 SELECT mit einer TABLE

2.1 query specification

```
<query specification> ::=
            SELECT  [ ALL | DISTINCT ]  ...
            FROM  ...
            WHERE  <search condition>
            GROUP BY ...
            HAVING  <search condition>

<search condition> ::=
            (<predicate> OR <predicate>)
             AND (<predicate>)
             AND NOT (<predicate>)

 <predicate> ::=
            ... <comp op> ...
            ...     BETWEEN ... AND ...
            ... NOT BETWEEN ... AND ...
            ...     IN (...)
            ... NOT IN (...)
            ...     LIKE ... ESCAPE ...
            ... NOT LIKE ... ESCAPE ...
            ... IS     NULL
            ... IS NOT NULL

<comp op>::=
            =   >   <   <=   >=  <>
```

Die folgenden Sprachelemente werden von den meisten Produkten am Markt nicht unterstützt:

```
<predicate> ::=
            ... IS TRUE
            ... IS FALSE
            ... IS UNKNOWN
            ... IS NOT TRUE
            ... IS NOT FALSE
            ... IS NOT UNKNOWN
```

2.2 SELECT

```
SELECT *        FROM Tbuch
;
SELECT Tbuch.* FROM Tbuch
;
SELECT
  buchnr, erschj, preis, titel, verlagnr
FROM Tbuch
;
SELECT
  buchnr
, erschj
, preis
, titel
, verlagnr
FROM Tbuch
;
```

Buchnr	Erschj	Preis	Titel	Verlagnr
5	1988	3.50	Ansichten eines Clowns	NULL
27	NULL	99.99	die Jüdin von Toledo	NULL
6	1988	20.50	die Blechtrommel	NULL
7	1989	99.99	der Name der Rose	NULL
8	1977	0.50	der Butt	1111
9	1990	55.00	DB2 fuer Sie	1111
11	1990	NULL	Elvis in Heidelberg	NULL
12	1989	NULL	a guide to db2	NULL
18	1989	99.99	Database Systems	NULL
1	NULL	NULL	C	NULL
2	NULL	NULL	C	NULL

Bei "SELECT *" wird die beim CREATE TABLE gewählte Reihenfolge der Spalten benützt. Die Bedeutung von "SELECT * " ändert sich, wenn zum Beispiel die TABLE um eine Spalte erweitert wird.

Für das Anlisten der Daten gibt es keine vordefinierte Ordnung der Spalten (links, rechts). Der Endanwender entscheidet über die Anordnung der Spalten:

```
SELECT Buchnr, Erschj, Preis, Titel
FROM Tbuch
;
SELECT Erschj, Titel, Buchnr
FROM Tbuch
;
```

Den Spalten kann explizit ein Name gegeben werden.

```
SELECT
       Erschj  AS Erscheinungs_Jahr
     , Titel   AS Titel
     , Buchnr  AS Buchnummer
FROM Tbuch
;
SELECT
       Erschj  AS "Erscheinungs-Jahr"
     , Titel   AS "Titel des Buches"
     , Buchnr  AS "Buch-Nummer"
FROM Tbuch
;
```

2.3 SELECT ... ORDER BY

2.3.1 ORDER BY und die NULL

--aufsteigend sortiert nach Erscheinungsjahr und Buchnr

```
SELECT Buchnr      AS Bnr
     , Erschj      AS Erschj
     , Preis       AS Preis
     , Verlagnr    AS Vnr
FROM Tbuch
ORDER BY erschj ASC, bnr ASC
;
```

Beispiel Oracle und DB2 (NULL am Schluss bei ASC)

BNR	ERSCHJ	PREIS	VNR
8	1977	,5	1111
5	1988	3,5	null
6	1988	20,5	null
7	1989	99,99	null
12	1989	null	null
18	1989	99,99	null
9	1990	55	1111
11	1990	null	null
1	**null**	**null**	**null**
2	**null**	**null**	**null**
27	**null**	**99,99**	**null**

Beispiel SQL Server (NULL am Anfang bei ASC)

Bnr	Erschj	Preis	Vnr
1	**NULL**	**NULL**	**NULL**
2	**NULL**	**NULL**	**NULL**
27	**NULL**	**99.99**	**NULL**
8	1977	0.50	1111
5	1988	3.50	NULL
6	1988	20.50	NULL
7	1989	99.99	NULL
12	1989	NULL	NULL
18	1989	99.99	NULL
9	1990	55.00	1111
11	1990	NULL	NULL

- **Nur mir Hilfe der Option ORDER BY ist sichergestellt, dass die Daten in der gewünschten Reihenfolge angelistet werden!**
- **Wenn in der ORDER BY-Klausel kein Schlüssel angegeben ist (d.h. alle Spalten eines Primärschlüssels bzw. Alternativschlüssels, PRIMARY KEY bzw. UNIQUE), dann werden die Zeilen eine systemabhängige Ordnung haben („der Zufall regiert“).**
- Sortieren nach mehreren Spalten (aufsteigend ASC bzw. absteigend DESC) ist möglich.

```
SELECT Buchnr, Erschj, Preis FROM Tbuch
ORDER BY Erschj ASC, Preis DESC, Buchnr;
```

- "ORDER BY 2" Identifizierung der Spalte durch eine Ziffer; ORDER BY mit Hilfe einer Ziffer ist ein "deprecated feature" und wird deshalb in Zukunft möglicherweise nicht mehr unterstützt.

```
SELECT Buchnr, Erschj, Preis FROM Tbuch
ORDER BY 2 ASC, 3 DESC, 1 ASC;
```

- **„Zum Zwecke der Ordnung werden alle Nullmarken entweder so betrachtet, als seien sie größer als alle Werte ungleich der Nullmarke oder aber, als seien sie kleiner als alle Werte ungleich der Nullmarke; welche von den beiden Möglichkeiten allerdings zutrifft, ist durch die Implementierung definiert.“ DATE DARWEN 1998, Seite 278**

2.3.2 ORDER BY und Zeichenketten, Zeichenordnung

- **Beim numerischen Vergleich entspricht das Ergebnis der mathematischen Ordnung (der mathematischen Ordnungsrelation).**
- Die Zeichenordnung (collating sequence) einer Zeichenmenge (character set) regelt den Vergleich von Zeichenketten (character strings). Die Zeichenmenge und ihre Zeichenordnung ist durch die Implementierung definiert (implementation-defined) und wird bei den meisten Produkten beim Generieren der Datenbank spezifiziert (CREATE DATABASE, diese Anweisung ist produktabhängig, **der SQL Standard kennt den Begriff DATABASE nicht!).**
- **Verschiedene Zeichen können von einer Zeichenordnung durchaus als gleich eingestuft werden (z.B. ‘a‘=‘A‘) und sogar ‘ß‘=‘ss‘ ist möglich. Wir können aber zunächst nur davon ausgehen, dass ‘A‘ < ‘B‘ < ‘C‘ < ... < ‘Z‘ und ‘a‘ <‘b‘ < ‘c‘ < ... < ‘z‘ und ‘0‘ < ‘1‘ < ‘2‘ < ... < ‘9‘.**

2.3.3 Testdaten

```
drop table testnls;
CREATE TABLE testnls
( schl integer   NOT NULL,
  spaltechar char(5) NOT NULL,
  spaltevar  varchar(5),
  PRIMARY KEY(schl)
);
INSERT Into TESTNLS (schl, spaltechar) values ( 0  , '0');
INSERT Into TESTNLS (schl, spaltechar) values ( 1  , '1');
INSERT Into TESTNLS (schl, spaltechar) values ( 2  , '11');
INSERT Into TESTNLS (schl, spaltechar) values ( 3  , '2');
INSERT Into TESTNLS (schl, spaltechar) values ( 4  , '22');
INSERT Into TESTNLS (schl, spaltechar) values ( 5  , '3');
INSERT Into TESTNLS (schl, spaltechar) values ( 6  , '9');
INSERT Into TESTNLS (schl, spaltechar) values ( 7  , 'A');
INSERT Into TESTNLS (schl, spaltechar) values ( 8  , 'AA');
INSERT Into TESTNLS (schl, spaltechar) values ( 9  , 'B');
INSERT Into TESTNLS (schl, spaltechar) values (10  , 'BB');
INSERT Into TESTNLS (schl, spaltechar) values (11  , 'Z');
INSERT Into TESTNLS (schl, spaltechar) values (12  , 'a');
INSERT Into TESTNLS (schl, spaltechar) values (13  , 'aa');
INSERT Into TESTNLS (schl, spaltechar) values (14  , 'b');
INSERT Into TESTNLS (schl, spaltechar) values (15  , 'bb');
INSERT Into TESTNLS (schl, spaltechar) values (16  , 'z');
INSERT Into TESTNLS (schl, spaltechar) values (17  , 'Ä');
INSERT Into TESTNLS (schl, spaltechar) values (18  , 'Ü');
INSERT Into TESTNLS (schl, spaltechar) values (19  , 'Ö');
INSERT Into TESTNLS (schl, spaltechar) values (20  , 'ä');
INSERT Into TESTNLS (schl, spaltechar) values (21  , 'ü');
INSERT Into TESTNLS (schl, spaltechar) values (22  , 'ö');
INSERT Into TESTNLS (schl, spaltechar) values (23  , 'S');
INSERT Into TESTNLS (schl, spaltechar) values (24  , 'SS');
INSERT Into TESTNLS (schl, spaltechar) values (25  , 'ß');
INSERT Into TESTNLS (schl, spaltechar) values (26  , 'ßß');
INSERT Into TESTNLS (schl, spaltechar) values (27  , 's');
INSERT Into TESTNLS (schl, spaltechar) values (28  , 'ss');
INSERT Into TESTNLS (schl, spaltechar) values (29  , '#');
INSERT Into TESTNLS (schl, spaltechar) values (30  , '/');
INSERT Into TESTNLS (schl, spaltechar) values (31  , 'd');
INSERT Into TESTNLS (schl, spaltechar) values (32  , 'D');
INSERT Into TESTNLS (schl, spaltechar) values (33  , 'dd');
INSERT Into TESTNLS (schl, spaltechar) values (34  , 'DD');
INSERT Into TESTNLS (schl, spaltechar) values (35  , 'dD');
INSERT Into TESTNLS (schl, spaltechar) values (36  , 'Dd');
INSERT Into TESTNLS (schl, spaltechar) values (37  , 'de');
INSERT Into TESTNLS (schl, spaltechar) values (38  , 'DE');
INSERT Into TESTNLS (schl, spaltechar) values (39  , 'dE');
INSERT Into TESTNLS (schl, spaltechar) values (40  , 'De');
COMMIT;
```

2.3.4 Beispiel SQL Server

CREATE DATABASE mit COLLATE Latin1_General_CI_AS

mit Parameter CI case insensitive

```
select spaltechar, schl
from testnls
----where spaltechar > 'a'
order by spaltechar ASC, schl ASC
;
 spaltechar schl
---------- -----------
#          29
/          30
0          0
1          1
11         2
2          3
22         4
3          5
9          6
A          7
a          12
Ä          17
ä          20
AA         8
aa         13
B          9
b          14
BB         10
bb         15
d          31
D          32
dd         33
DD         34
dD         35
Dd         36
de         37
DE         38
dE         39
De         40
Ö          19
ö          22
S          23
s          27
SS         24
ß          25
ss         28
ßß         26
Ü          18
ü          21
Z          11
z          16
(41 Zeile(n) betroffen)
```

mit Parameter CS case sensitive auf Anweisungsebene:

```
select spaltechar, schl
from testnls
----where spaltechar > 'a' COLLATE Latin1_General_CS_AS
order by spaltechar COLLATE Latin1_General_CS_AS ASC , schl ASC
;
```

2.3.5 Beispiel DB2

Zeichenordnung in einer Windows-Umgebung (Sortierfolge der Datenbank System_1252, Codepage für Datenbank 1208).

```
select spaltechar, schl

from testnls

--where spaltechar > 'A'

order by spaltechar ASC, schl ASC

;

SPALTECHAR SCHL
---------- -----------
/                    30
#                    29
0                     0
1                     1
11                    2
2                     3
22                    4
3                     5
9                     6
a                    12
aa                   13
A                     7
AA                    8
ä                    20
Ä                    17
b                    14
bb                   15
B                     9
BB                   10
d                    31
dd                   33
dD                   35
de                   37
dE                   39
D                    32
Dd                   36
DD                   34
De                   40
DE                   38
ö                    22
Ö                    19
s                    27
ss                   28
S                    23
SS                   24
ß                    25
ßß                   26
ü                    21
Ü                    18
z                    16
Z                    11
  41 Satz/Sätze ausgewählt.
```

Beachten Sie bitte: ‚aa' < ‚A' bzw. ‚de' < ‚D'

Parameter auf Anweisungsebene
(a culturally correct German collation**):**

```
select spaltechar, schl
from testnls
----where COLLATION_KEY_BIT(spaltechar, 'UCA400R1_LDE')
----    > COLLATION_KEY_BIT('A', 'UCA400R1_LDE')
ORDER BY
COLLATION_KEY_BIT(spaltechar, 'UCA400R1_LDE'),schl
;
```

2.3.6 Beispiel Oracle

Zeichenordnung nach ASCII in einer Windows-Umgebung.

```
alter session set nls_sort = binary;
alter session set nls_comp = binary;
select spaltechar, schl
from testnls
where spaltechar > 'ö'
order by spaltechar ASC , schl ASC
;
SPALT      SCHL
----- ----------
#             29
/             30
0              0
1              1
11             2
2              3
22             4
3              5
9              6
A              7
AA             8
B              9
BB            10
D             32
DD            34
DE            38
Dd            36
De            40
S             23
SS            24
Z             11
a             12
aa            13
b             14
bb            15
d             31
dD            35
dE            39
dd            33
de            37
s             27
ss            28
z             16
Ä             17
Ö             19
Ü             18
ß             25
ßß            26
ä             20
ö             22
ü             21
41 rows selected.
```

Parameter auf Anweisungsebene:

```
select spaltechar, schl
from testnls
WHERE NLSSORT(spaltechar , 'NLS_SORT = binary' )
    > NLSSORT('ö'        , 'NLS_SORT = binary' )
order by
      NLSSORT(spaltechar , 'NLS_SORT = binary' ) ASC
     ,schl ASC
;
```

Beispiel Oracle ‚linguistic sort sequence German' in einer Windows-Umgebung.

```
alter session set nls_sort = german;
alter session set nls_comp = linguistic;
select spaltechar, schl
from testnls
where spaltechar > 'ö'
order by spaltechar, schl
;
SPALT      SCHL
----- ----------
#             29
/             30
a             12
A              7
ä             20
Ä             17
aa            13
AA             8
b             14
B              9
bb            15
BB            10
d             31
D             32
dd            33
dD            35
Dd            36
DD            34
de            37
dE            39
De            40
DE            38
ö             22
Ö             19
s             27
S             23
ß             25
ss            28
SS            24
ßß            26
ü             21
Ü             18
z             16
Z             11
0              0
1              1
11             2
2              3
22             4
3              5
9              6
41 rows selected.
```

Beachten Sie bitte: ‚A' < ‚aa' bzw. ‚D' < ‚de'

Parameter auf Anweisungsebene:

```
select spaltechar, schl
from testnls
WHERE NLSSORT(spaltechar , 'NLS_SORT = GERMAN' )
    > NLSSORT('ö'        , 'NLS_SORT = GERMAN' )
order by
      NLSSORT(spaltechar , 'NLS_SORT = GERMAN' )
     ,schl
;
```

2.3.7 Beispiel Oracle SQL Developer, Voreinstellungen im Client

Überprüfen Sie bitte die Voreinstellungen im Oracle SQL Developer unter

Extras/Voreinstellungen/Datenbank/NLS

```
Sortieren GERMAN
Vergleich binary
```

das entspricht

```
alter session set nls_sort = GERMAN;
alter session set nls_comp = binary;
```

Diese Voreinstellungen, wenn denn so vorhanden, müssen Sie ändern!

Diese Kombination ist nicht sinnvoll!

Was liefern bei diesen Voreinstellungen die SELECT-Anweisungen?

```
select spaltechar, schl
from testnls
order by spaltechar, schl  --sort german
;
  ---------------------------------
select spaltechar, schl
from testnls
where spaltechar > 's'     --comp binary
order by spaltechar, schl  --sort german
;
```

```
SPALT       SCHL
----- ----------
#             29
/             30
a             12
A              7
ä             20
Ä             17
aa            13
AA             8
b             14
B              9
bb            15
BB            10
d             31
D             32
dd            33
dD            35
Dd            36
DD            34
de            37
dE            39
De            40
DE            38
ö             22
Ö             19
s             27
S             23
ß             25
ss            28
SS            24
```

```
ßß             26
ü              21
Ü              18
z              16
Z              11
0               0
1               1
11              2
2               3
22              4
3               5
9               6

41 Zeilen gewählt.


  ------------------------------------

SPALT       SCHL
----- ----------
ä              20
Ä              17
ö              22
Ö              19
ß              25
ss             28
ßß             26
ü              21
Ü              18
z              16

10 Zeilen gewählt.
```

2.4 SELECT DISTINCT

```
-- die verschiedenen Werte in der Spalte Preis
SELECT DISTINCT Preis FROM Tbuch;
```

Preis
3.50
20.50
99.99
0.50
55.00
NULL

- Beachten Sie, dass die NULL bei Preis nur einmal angelistet wird.

```
-- die verschiedenen Kombinationen in Erschj,Preis
SELECT DISTINCT Erschj,Preis FROM Tbuch;
```

Erschj	Preis
1988	3.50
NULL	99.99
1988	20.50
1989	99.99
1977	0.50
1990	55.00
1990	NULL
1989	NULL
NULL	NULL

- Die **Projekt-Operation** (auch **Projektion** genannt): Auswahl von Spalten und entfernen von Duplikat-Zeilen (mehrfach vorhandene Zeilen werden, sofern sie nach dem Entfernen von Spalten auftreten, bis auf je eine gestrichen). Das Ergebnis ist wieder eine Table/Relation.
- DISTINCT bezieht sich auf die ganze Zeile (alle ausgewählten Spalten) und bedeutet "entferne Duplikate“.
- DISTINCT benötigt eventuell einen internen Sort!
- Leider ist der Default nicht DISTINCT, sondern ALL!

Die folgenden SELECT-Anweisungen sind syntaktisch korrekt, sie implementieren aber keine relationalen Projektionen und liefern Listen mit Duplikaten.

```
SELECT ALL Preis FROM Tbuch;

SELECT     Preis FROM Tbuch;

SELECT ALL Erschj,Preis FROM Tbuch;

SELECT     Erschj,Preis FROM Tbuch;
```

2.5 SELECT ... WHERE ... IS NULL

```
SELECT buchnr, erschj FROM Tbuch
WHERE Erschj = 1990
;
buchnr      erschj
----------- ---------------------------------------
9           1990
11          1990

SELECT buchnr, erschj FROM Tbuch
WHERE NOT (Erschj = 1990)
;
buchnr      erschj
----------- ---------------------------------------
5           1988
6           1988
7           1989
8           1977
12          1989
18          1989

SELECT buchnr, erschj FROM Tbuch
WHERE Erschj IS NULL
;
buchnr      erschj
----------- ---------------------------------------
1           NULL
2           NULL
27          NULL
```

Beim SELECT werden die Zeilen angelistet, für die die Suchbedingung wahr ist.

Suchbedingung	NOT Suchbedingung
wahr/TRUE	falsch/FALSE
falsch/FALSE	wahr/TRUE
unbekannt/UNKNOWN	unbekannt/UNKNOWN

Achtung bei Oracle, SQL Server:

```
SELECT * FROM Tbuch WHERE PREIS = NULL;
SELECT * FROM Tbuch WHERE NOT (PREIS = NULL);
```

Beide Anweisungen liefern jeweils die leere Menge!

Achtung bei DB2 for z/OS:

```
SELECT * FROM Tbuch WHERE PREIS = NULL;
SELECT * FROM Tbuch WHERE NOT (PREIS = NULL);
```

Beide Anweisungen sind syntaktisch nicht korrekt!

Achtung bei DB2 for Windows:

Testen Sie ihre aktuelle Version, ihre Installation.

- „Erschj=1990“ wird auch Suchbedingung/search condition genannt.
- Eine Suchbedingung kann **wahr** sein, **falsch** sein oder **unbekannt.**
- „a search condition specifies a condition that is **true**, **false** or **unknown** about a given row"
- wahr, falsch, unbekannt sind Wahrheitswerte, unbekannt ist ein dritter Wahrheitswert!
- **unbekannt** kann nicht ohne weiteres wie falsch behandelt werden.
- **unbekannt** ist ein Wahrheitswert der dreiwertigen Logik, ist also etwas anderes als NULL.

NOT wahr entspricht falsch

NOT falsch entspricht wahr

NOT unbekannt entspricht unbekannt

Bücher mit wohldefiniertem Erscheinungsjahr:

```
SELECT Buchnr, Erschj
FROM Tbuch
WHERE Erschj IS NOT NULL;
Buchnr      Erschj
----------- ------
5           1988
6           1988
7           1989
8           1977
9           1990
11          1990
12          1989
18          1989
```

2.6 SELECT ... WHERE Suchbedingung

```
-- Buchnr und Titel aller Bücher
-- von 1990
SELECT
  Buchnr, Titel
FROM Tbuch
WHERE Erschj = 1990
;
```

```
Buchnr      Titel
----------- ----------------------------------------
9           DB2 fuer Sie
11          Elvis in Heidelberg
```

```
-- Erscheinungsjahr, Preis, Buchnr aller Bücher
-- die 1980 oder später erschienen sind
SELECT
Erschj, Preis, Buchnr
FROM Tbuch
WHERE Erschj >= 1980
;
```

```
Erschj     Preis                                  Buchnr
---------- -------------------------------------- -----------
1988       3.50                                   5
1988       20.50                                  6
1989       99.99                                  7
1990       55.00                                  9
1990       NULL                                   11
1989       NULL                                   12
1989       99.99                                  18
```

- Auswahl von Zeilen mit Hilfe der Suchbedingung in der WHERE-Klausel.
- Vergleichsoperatoren sind = > < <= >= <>

2.7 CHAR oder VARCHAR, Oracle und der Datentyp VARCHAR2

```
select verlagnr, verlag from tverlag
where verlag = 'Forkel'
;
select verlagnr, verlag from tverlag
where verlag = 'Forkel       '
;
select buchnr, titel from tbuch
where Titel  = 'der Butt'
;

  VERLAGNR VERLAG
---------- --------------------
      1111 Forkel


  VERLAGNR VERLAG
---------- --------------------
      1111 Forkel


    BUCHNR       TITEL
--------------  ----------------------
         8       der Butt
```

--DB2 bzw. SQL Server

```
select buchnr, titel from tbuch
where Titel  = 'der Butt     '
;
    BUCHNR       TITEL
--------------  ----------------------
         8       der Butt
```

--Oracle mit VARCHAR2

```
select buchnr, titel from tbuch
where Titel  = 'der Butt     '
;
```

Es wurden keine Zeilen ausgewählt.

Der Vergleich von Zeichenketten (character strings) erfolgt gemäß einer bestimmten Zeichenordnung (collating sequence, collation) paarweise von links nach rechts. Wenn die Zeichenketten von gleicher Länge sind, dann bestimmt der erste paarweise Vergleich, der nicht „gleich“ ergibt, das Ergebnis. Bei Zeichenketten unterschiedlicher Länge und einer **Zeichenordnung mit PAD SPACE** wird die kürzere Zeichenkette rechts mit Leerzeichen aufgefüllt, um sie auf die gleiche Länge wie die längere zu bringen; die Regeln zum Vergleich von Zeichenketten gleicher Länge werden dann angewandt.

`SELECT * from Tbuch WHERE` **`Titel ='C  ';`** **liefert bei einer Zeichenordnung mit PAD SPACE** auch die Titel die gerade aus dem Charakterstring ‚C‘ bestehen (DB2, SQL Server).

`SELECT * from Tbuch WHERE` **`Titel ='C  ';`** **liefert bei einer Zeichenordnung mit NO PAD** die Titel die gerade aus dem Charakterstring ‚C‘ bestehen aber **nicht.**

```
X = 'ijk'
Y = 'ijk  '
Z = 'ij'
```

Vergleich	PAD SPACE	NO PAD	DB2 CHAR VARCHAR	ORACLE CHAR	**ORACLE VARCHAR2**
X=Y	true	false	true	true	**Falsch**
X<Y	false	true	false	false	**Wahr**
X>Z	true	true	true	true	True

Achtung Oracle und der Datentyp VARCHAR2:

- Eine in der CREATE-Anweisung mit VARCHAR definierte Spalte wird von ORACLE mit dem proprietären Datentyp VARCHAR2 generiert.
- `WHERE Titel ='C  '` **Oracle liefert** die Titel die gerade aus dem Charakterstring ‚C‘ bestehen **bei CHAR, aber nicht bei VARCHAR2.**

Achtung VARCHAR und VARCHAR2:

- **DB2 und SQL Server** kennen einen **VARCHAR**-String der Länge 0.
- **Oracle** interpretiert einen **VARCHAR2**-String ohne Daten als NULL mit der Länge NULL.

2.8 Dreiwertige Aussagenlogik, TRUE, FALSE, UNKNOWN

```
-- Buchnr und Titel aller Bücher
-- nach 1980 und Preis unter 90.00
SELECT Buchnr, Erschj, Preis
FROM Tbuch
WHERE Erschj > 1980 AND PREIS < 90.00;
```

```
    BUCHNR ERSCHJ     PREIS
---------- ------ ---------
         5   1988      3.50
         6   1988     20.50
         9   1990     55.00
```

```
-- Buchnr und Titel aller Bücher
-- nach 1980 oder Preis unter 90.00
SELECT Buchnr, Erschj, Preis
FROM Tbuch
WHERE Erschj > 1980 OR PREIS < 90.00;
```

```
    BUCHNR ERSCHJ     PREIS
---------- ------ ---------
         5   1988      3.50
         6   1988     20.50
         7   1989     99.99
         8   1977      0.50
         9   1990     55.00
        11   1990 NULL
        12   1989 NULL
        18   1989     99.99
```

- Beim SELECT werden die Zeilen angelistet, für die die Suchbedingung wahr ist.
- Die Suchbedingung (search condition) in der Where-Klausel (where-clause) kann mit Hilfe der üblichen Operatoren und Klammern gebildet werden: AND NOT OR) (
- Die Alltagssprache kennt das exklusive oder und das inklusive oder. **Das oder der Aussagenlogik ist inklusive!**
- AND bindet stärker als OR

A	B	A AND B	A OR B
wahr	wahr	Wahr	wahr
wahr	falsch	falsch	wahr
falsch	wahr	falsch	wahr
falsch	falsch	falsch	falsch
wahr	unbekannt	unbekannt	wahr
unbekannt	wahr	unbekannt	wahr
unbekannt	unbekannt	unbekannt	unbekannt
falsch	unbekannt	falsch	unbekannt
unbekannt	falsch	falsch	unbekannt

A	NOT A
wahr	falsch
falsch	wahr
unbekannt	unbekannt

A	B	C	(A AND B) OR C A AND B OR C	A AND (B OR C)
wahr	wahr	wahr		
wahr	wahr	falsch		
wahr	falsch	wahr		
wahr	falsch	falsch		
falsch	wahr	wahr		
falsch	wahr	falsch		
falsch	falsch	wahr	wahr	falsch
falsch	falsch	falsch		
wahr	wahr	unbekannt		
usw.	usw.	usw.		

2.9 Logische Operatoren AND OR NOT

Aktuelle Daten von Tbuch:

```
buchnr      erschj                                preis
----------- ------------------------------------- -------
100         1989                                  99.99
200         1989                                  22.22
300         2000                                  99.99
400         2000                                  22.22
----------- ------------------------------------- -------
500         1989                                  NULL
600         NULL                                  99.99
700         NULL                                  NULL
800         2000                                  NULL
900         NULL                                  22.22
```

B1: Tbuch.Erschj = 1989 B2: Tbuch.Preis = 99.99

Welche Zeilen qualifizieren?

Bedingung für die Base-Table Tbuch	IST	SOLL
B1 erschj=1989	100 200 500	
NOT B1 **IST** **(not erschj=1989)** **SOLL** **(not erschj=1989 OR erschj is null)**	**300** **400** **800**	**300** **400** **800** **700** **600** **900**
B2		
NOT B2		
B1 AND B2 1989 und 99.99		
NOT(B1 AND B2) äquivalent (NOT B1) OR (NOT B2)		
B1 OR B2 1989 oder 99.99		
NOT(B1 OR B2) äquivalent (NOT B1) AND (NOT B2)		
B1 AND (NOT B2)		
(NOT B1) AND B2		
B1 OR (NOT B2)		
(NOT B1) OR B2		
(B1 OR B2) AND NOT (B1 AND B2) äquivalent (B1 AND (NOT B2)) OR (B2 AND (NOT B1)) ??>>entweder 1989 oder 99.99 aber nicht beides gleichzeitig<<??		

Liefern die SELECT-Anweisungen die von Ihnen gewünschten Zeilen?

2.9.1 Beispiel NOT B1

--Ist: ohne besondere Berücksichtigung der NULL

```
select buchnr, erschj , preis
from tbuch
where
(NOT Erschj = 1989)
;
```

--Soll: mit besonderer Berücksichtigung der NULL

```
select buchnr, erschj , preis
from tbuch
where
((NOT Erschj = 1989) OR (erschj IS NULL))
;
```

--Soll: mit besonderer Berücksichtigung der NULL

--Alternative EXCEPT Oracle MINUS

```
  select buchnr, erschj , preis from tbuch
EXCEPT
  select buchnr, erschj , preis from tbuch
  WHERE Erschj = 1989
;
```

```
delete from tbuch
;
INSERT INTO tbuch(buchnr,erschj, preis, titel) VALUES
 ( 100, 1989, 99.99, 'xyz');
INSERT INTO tbuch(buchnr,erschj, preis, titel) VALUES
 ( 200, 1989, 22.22, 'xyz');
INSERT INTO tbuch(buchnr,erschj, preis, titel) VALUES
 ( 300, 2000, 99.99, 'xyz');
INSERT INTO tbuch(buchnr,erschj, preis, titel) VALUES
 ( 400, 2000, 22.22, 'xyz');
INSERT INTO tbuch(buchnr,erschj, preis, titel) VALUES
 ( 500, 1989, NULL, 'xyz');
INSERT INTO tbuch(buchnr,erschj, preis, titel) VALUES
 ( 600, NULL, 99.99, 'xyz');
INSERT INTO tbuch(buchnr,erschj, preis, titel) VALUES
 ( 700, NULL, NULL, 'xyz');
INSERT INTO tbuch(buchnr,erschj, preis, titel) VALUES
 ( 800, 2000, NULL, 'xyz');
INSERT INTO tbuch(buchnr,erschj, preis, titel) VALUES
 ( 900, NULL, 22.22, 'xyz');
select buchnr, erschj, preis from tbuch order by buchnr
;
```

2.9.2 Beispiel NOT (B1 AND B2)

--Ist: ohne besondere Berücksichtigung der NULL

```
select buchnr, erschj , preis
from tbuch
where
NOT (Erschj = 1989 AND Preis=99.99)
;
select buchnr, erschj , preis
from tbuch
where
(NOT Erschj = 1989) OR (NOT Preis=99.99)
;
```

--Soll: mit besonderer Berücksichtigung der NULL

```
select buchnr, erschj , preis
from tbuch
where
(NOT Erschj = 1989 or erschj is null)
OR
(NOT Preis=99.99 or preis is null)
;
```

Ersetze überall

```
(NOT Erschj   = 1989)
```

durch

```
((NOT Erschj  = 1989)  OR (erschj IS NULL))
```

und

```
(NOT Preis    = 99.99)
```

durch

```
((NOT Preis   = 99.99) OR (preis IS NULL))
```

--Soll: mit besonderer Berücksichtigung der NULL

--Alternative EXCEPT Oracle MINUS, INTERSECT

```
  select buchnr, erschj , preis from tbuch
EXCEPT
(
  select buchnr, erschj , preis from tbuch
  WHERE Erschj = 1989
  INTERSECT
  select buchnr, erschj , preis from tbuch
  WHERE PREIS = 99.99
)
;
```

2.9.3 Beispiel NOT (B1 OR B2)

--Ist: ohne besondere Berücksichtigung der NULL

```
select buchnr, erschj , preis
from tbuch
where
NOT (Erschj = 1989 OR Preis=99.99)
;
select buchnr, erschj , preis
from tbuch
where
(NOT Erschj = 1989) AND (NOT Preis=99.99)
;
```

--Soll: mit besonderer Berücksichtigung der NULL

```
select buchnr, erschj , preis
from tbuch
where
(NOT Erschj = 1989 or erschj is null)
AND
(NOT Preis=99.99 or preis is null)
;
```

Ersetze überall

```
(NOT Erschj   = 1989)
```

durch

```
((NOT Erschj  = 1989)  OR (erschj IS NULL))
```

und

```
(NOT Preis    = 99.99)
```

durch

```
((NOT Preis   = 99.99) OR (preis IS NULL))
```

--Soll: mit besonderer Berücksichtigung der NULL

--Alternative EXCEPT Oracle MINUS, UNION

```
  select buchnr, erschj , preis from tbuch
EXCEPT
(
  select buchnr, erschj , preis from tbuch
  WHERE Erschj = 1989
  UNION
  select buchnr, erschj , preis from tbuch
  WHERE PREIS = 99.99
)
;
```

2.9.4 Beispiel (NOT B1) OR B2 Logische Implikation

- Hole alle Bücher, wenn aber Erschj=1989 dann muss Preis=99.99 sein.
- Liste aller Bücher, aber von den 1989-ern nur die mit Preis=99.99.
- Wenn erschj=1989 dann preis=99.99.
- „Wenn B1 dann B2“ ist äquivalent mit „NOT B1 OR B2“.

B1	B2	(NOT B1) OR B2 wenn B1 dann B2	(B1 and B2) OR NOT B1
wahr	wahr	wahr	wahr
wahr	falsch	falsch	falsch
falsch	wahr	wahr	wahr
falsch	falsch	wahr	wahr
wahr	unbekannt		
unbekannt	wahr	wahr	unbekannt
unbekannt	unbekannt		
falsch	unbekannt		
unbekannt	falsch		

--Ist: ohne besondere Berücksichtigung der NULL

```
select
buchnr, erschj, preis
from Tbuch
where 1=1
AND ( erschj <> 1989 OR preis=99.99 )
;
buchnr      erschj                                  preis
----------- --------------------------------------- ---------
100         1989                                    99.99
300         2000                                    99.99
400         2000                                    22.22
600         NULL                                    99.99
800         2000                                    NULL
select
buchnr, erschj, preis
from Tbuch
where 1=1
AND ( (erschj = 1989 AND preis=99.99) OR Erschj <> 1989 )
;
buchnr      erschj                                  preis
----------- --------------------------------------- ---------
100         1989                                    99.99
300         2000                                    99.99
400         2000                                    22.22
800         2000                                    NULL
```

--Soll: mit besonderer Berücksichtigung der NULL

```
select
buchnr, erschj, preis
from Tbuch
where 1=1
AND (
     (erschj <> 1989 OR Erschj is null) OR preis=99.99
    )
;
select
buchnr, erschj, preis
from Tbuch
where 1=1
AND (
          (erschj = 1989 AND preis=99.99 )
      OR
           (Erschj <> 1989 or Erschj is null)
     )
;
```

buchnr	erschj	preis
100	1989	99.99
300	2000	99.99
400	2000	22.22
600	NULL	99.99
700	NULL	NULL
800	2000	NULL
900	NULL	22.22

--Soll: mit besonderer Berücksichtigung der NULL

--Alternativen…

```
(
select buchnr, erschj, preis from Tbuch
EXCEPT
select buchnr, erschj, preis from Tbuch
WHERE erschj = 1989
)
UNION
select buchnr, erschj, preis from Tbuch
where Preis=99.99
;
-----------------------
(
select buchnr, erschj, preis from Tbuch
WHERE erschj = 1989
INTERSECT
select buchnr, erschj, preis from Tbuch
where Preis=99.99
)
UNION
(
select buchnr, erschj, preis from Tbuch
EXCEPT
select buchnr, erschj, preis from Tbuch
where Erschj=1989
)
;
```

2.10 SELECT ... WHERE ... BETWEEN

```
-- Preis zwischen 3.50 und 55.00
SELECT Buchnr, Erschj, Preis , Titel
FROM Tbuch
WHERE Preis BETWEEN  3.50 AND 55.00;

SELECT Buchnr, Erschj, Preis , Titel
FROM Tbuch
WHERE 3.50 <= Preis  AND Preis <= 55.00;
```

```
    BUCHNR ERSCHJ     PREIS TITEL
---------- ------ --------- ----------------------
         5   1988      3.50 Ansichten eines Clowns
         6   1988     20.50 die Blechtrommel
         9   1990     55.00 DB2 fuer Sie
```

- BETWEEN entspricht größer/gleich und kleiner/gleich

DB2 mit codeset IBM-1252, codepage 1252

```
select spaltechar, schl
from testnls
where spaltechar between 'B' and 'D'
order by spaltechar, schl;
SPALTECHAR SCHL
---------- -----------
B                    9
BB                  10
d                   31
dd                  33
dD                  35
de                  37
dE                  39
D                   32
  8 Satz/Sätze ausgewählt.
```

Oracle mit

alter session set nls_sort = german;

alter session set nls_comp = linguistic;

```
select spaltechar, schl
from testnls
where spaltechar between 'B' and 'D'
order by spaltechar, schl;
SPALT       SCHL
----- ----------
B              9
bb            15
BB            10
d             31
D             32
5 rows selected.
```

SQL Server mit
CREATE DATABASE mit COLLATE Latin1_General_CI_AS

```
select spaltechar, schl
from testnls
where spaltechar between 'B' and 'D'
order by spaltechar, schl;
spaltechar schl
---------- -----------
B          9
b          14
BB         10
bb         15
d          31
D          32
(6 Zeile(n) betroffen)
```

2.11 SELECT ... WHERE ... IN

```
-- Bücher von 1990, 1991 oder 1992
SELECT Buchnr, Erschj
FROM Tbuch
WHERE Erschj IN (1990, 1991, 1992)
;
SELECT Buchnr, Erschj
FROM Tbuch
WHERE
      Erschj =1990
   OR Erschj =1991
   OR Erschj =1992
;
Buchnr      Erschj
----------- ----------------------------------------
9           1990
11          1990

SELECT Buchnr, Erschj
FROM Tbuch
WHERE Erschj NOT IN (1990, 1991, 1992)
;
SELECT Buchnr, Erschj
FROM Tbuch
WHERE NOT
 (    Erschj =1990
   OR Erschj =1991
   OR Erschj =1992
 )
;
Buchnr      Erschj
----------- ----------------------------------------
5           1988
6           1988
7           1989
8           1977
12          1989
18          1989
```

- Welche Zeilen werden angelistet?

```
SELECT Buchnr, Erschj
FROM Tbuch
WHERE Erschj IN (1990, 1991, 1992, NULL)
;
Buchnr      Erschj
----------- ----------------------------------------
9           1990
11          1990

(2 Zeile(n) betroffen)

SELECT Buchnr, Erschj
FROM Tbuch
WHERE Erschj NOT IN (1990, 1991, 1992, NULL)
;
Buchnr      Erschj
----------- ----------------------------------------

(0 Zeile(n) betroffen)
```

- Welche Zeilen werden angelistet?

```
SELECT *
FROM Tbuch
WHERE     ( buchnr, erschj )
     IN   ( select buchnr, erschj from tbuch )
;
```

2.12 SELECT ... WHERE ... LIKE, Reguläre Ausdrücke

2.12.1 LIKE

Titel die mit dem Buchstaben D beginnen

```
SELECT buchnr, titel
FROM Tbuch
WHERE Titel LIKE 'D%'
;
```

sofern case-sensitive bei der Zeichenordnung (collating sequence)

```
buchnr      titel
----------- --------------------
9           DB2 fuer Sie
18          Database Systems
```

sofern case-insensitive bei der Zeichenordnung (collating sequence)

```
buchnr      titel
----------- ----------------------------
6           die Blechtrommel
7           der Name der Rose
8           der Butt
9           DB2 fuer Sie
18          Database Systems
27          die Jüdin von Toledo
```

Angabe von case-sensitive auf Anweisungsebene (Beispiel SQL Server)

```
SELECT buchnr, titel
FROM Tbuch
WHERE titel LIKE 'D%' COLLATE Latin1_General_CS_AS
```

- LIKE ist einsetzbar für CHAR und VARCHAR
- **"%"** das Prozentzeichen steht für eine beliebige Anzahl von Zeichen, die Anzahl kann auch 0 sein
- "_" das Unterstreichungszeichen steht für ein einzelnes Zeichen
- **Titel LIKE '%ABC%**' ist wahr, wenn Titel irgendwo den String 'ABC' enthält
- **Titel LIKE 'ABC_**' ist wahr, wenn Titel genau 4 Zeichen lang ist und die ersten 3 Zeichen aus 'ABC' bestehen
- **Titel LIKE '%ABC_'** ist wahr, wenn Titel aus 4 oder mehr Zeichen besteht und das viertletzte Zeichen A, das drittletzte Zeichen B und das vorletzte Zeichen C ist.
- ESCAPE

  ```
  ... WHERE Titel LIKE '10 /% ZINS%' ESCAPE '/'
  ```

  ist wahr, wenn die ersten 9 Zeichen aus dem String `'10 % ZINS'` bestehen (setze % bzw. _ außer Kraft).
- Im folgenden Beispiel (der kritische Fall) ist der Wahrheitswert abhängig von der Spaltendefinition!

  ```
  ... SET   Titel =    'ANSICHTEN'
  ... WHERE Titel LIKE '%ANSICHTEN'
  ```

Definition Titel	Wahrheitswert
CHAR(9)	wahr
CHAR(10)	falsch
VARCHAR(100)	wahr

2.12.2 Reguläre Ausdrücke, SQL:2008 LIKE_REGEXP und Oracle's REGEXP_LIKE

Oracle's REGEXP_LIKE prüft ob in einer Zeichenkette eine Teilkette enthalten ist die einem gegebenen Muster entspricht.

```
'...-....'
repräsentiert ein Muster nach dem gesucht werden soll. Die
Teilkette soll aus 3 Zeichen bestehen, gefolgt von einem Bin-
destrich (engl. dash) und weiteren 4 Zeichen (Achtung: ein
Leerzeichen/Blank ist auch ein Zeichen).
```

```
WHERE REGEXP_LIKE (titel, '...-....')
```

```
'[0-9]{3}[-.][0-9]{4}'
repräsentiert ein Muster nach dem gesucht werden soll. Die
Teilkette soll aus 3 Ziffern bestehen, gefolgt entweder von
einem Bindestrich (engl. dash) oder einem Punkt (engl. period
bzw. dot) und dann weiteren 4 Ziffern.
```

```
WHERE REGEXP_LIKE ( titel, '[0-9]{3}[-.][0-9]{4}' )
```

```
'^[[:digit:]]{3}[-.][[:digit:]]{4}$'
repräsentiert ein Muster nach dem gesucht werden soll. Die
Teilkette soll aus 3 Ziffern bestehen, gefolgt entweder von
einem Bindestrich (engl. dash) oder einem Punkt (engl. period
bzw. dot) und dann weiteren 4 Ziffern.
Die Teilkette muss aber am Anfang der Zeichenkette sein ^ und
am Ende der Zeichenkette $.
```

```
WHERE REGEXP_LIKE
     ( titel,
       '^[[:digit:]]{3}[-.][[:digit:]]{4}$'
     )
```

Im SQL:2008 Standard ist von LIKE_REGEXP die Rede.

Reguläre Ausdrücke (engl. regular expressions) sind ein leistungsstarkes Werkzeug zur Verarbeitung von Zeichenketten (engl. character strings).

In der UNIX Welt sind Reguläre Ausdrücke seit langem verbreitet und werden von vielen Programmier-Sprachen, Skript-Sprachen und Text-Editoren unterstützt.

Reguläre Ausdrücke ermöglichen es, Muster (engl. Pattern) in Zeichenketten zu definieren. Mit diesem Muster kann in Zeichenketten gesucht werden, Teile einer Zeichenkette können extrahiert bzw. verändert werden.

Mögliche Einsatzgebiete sind:

– das Suchen einer Teilkette mit einem bestimmten Muster in einer Zeichenkette
– das Extrahieren einer Teilkette mit einem bestimmtem Muster aus einer Zeichenkette
– das Ersetzen von Teilketten mit einem bestimmtem Muster durch andere Teilketten in einer Zeichenkette
– das Zählen von Teilketten mit einem bestimmtem Muster innerhalb einer Zeichenkette

Mit Regulären Ausdrücken können sehr komplexe Probleme elegant gelöst werden. Ein produktiver Einsatz ist für Ungeübte jedoch kein einfaches Unterfangen.

Reguläre Ausdrücke sind eine der Neuerungen von Oracle 10g und sind in den Manuals von Oracle beschrieben, zum Beispiel in der SQL Reference. Es stehen (analog zu LIKE, SUBSTR, REPLACE, INSTR) Funktionen zur Verfügung.

```
REGEXP_LIKE REGEXP_SUBSTR REGEXP_REPLACE REGEXP_INSTR
```

Mit Oracle 11g steht auch eine COUNT-Funktion zur Verfügung.

```
REGEXP_COUNT
```

Reguläre Ausdrücke können bei Oracle auf folgende Datentypen angewandt werden: CHAR VARCHAR2 CLOB NCHAR NVARCHAR NCLOB

2.12.3 Reguläre Ausdrücke und SQL Server

SQL Server unterstützt einige wenige Sprachelemente der Regulären Ausdrücke (Regular Expression).

```
select buchnr, titel
from tbuch

select buchnr, titel
from tbuch
where Titel like '[abc]%'
;
select buchnr, titel
from tbuch
where Titel like '[^abc]%'
;

select buchnr, titel
from tbuch
where Titel like '[a-c]%'
;
select buchnr, titel
from tbuch
where Titel like '[^a-c]%'
;
```

LIKE (Transact-SQL)

Bestimmt, ob eine bestimmte Zeichenfolge mit einem bestimmten Muster übereinstimmt. Ein Muster kann normale Zeichen und Platzhalterzeichen einschließen. Bei einem Mustervergleich müssen normale Zeichen exakt mit den angegebenen Zeichen in der Zeichenfolge übereinstimmen. Platzhalterzeichen können jedoch mit beliebigen Teilen der Zeichenfolge übereinstimmen. Das Verwenden der Vergleichsoperatoren für Zeichenfolgen = und != ist nicht so flexibel wie das Verwenden von Platzhalterzeichen mit dem LIKE-Operator. Wenn eines der Argumente kein Zeichenfolgen-Datentyp ist, wird es ggf. von SQL Server Database Engine (Datenbankmodul) in einen Zeichenfolgen-Datentyp konvertiert.

match_expression [NOT] LIKE pattern [ESCAPE escape_character]

match_expression

Ist ein gültiger Ausdruck eines beliebigen Zeichendatentyps.

pattern

Ist die bestimmte Zeichenfolge, nach der in *match_expression* gesucht werden soll, und kann die folgenden gültigen Platzhalterzeichen enthalten. *pattern* kann maximal 8.000 Bytes umfassen.

Platzhalter	Beschreibung	Beispiel
%	Eine Zeichenfolge aus null oder mehr Zeichen	WHERE title LIKE '%Computer%' findet alle Buchtitel, die das Wort 'Computer' enthalten.
_ (Unterstrich)	Ein einzelnes Zeichen	WHERE au_fname LIKE '_ean' findet alle Vornamen mit vier Buchstaben, die auf ean enden (Dean, Sean usw.).
[ ]	Beliebiges einzelnes Zeichen im angegebenen Bereich ([a-f]) oder in der angegebenen Menge ([abcdef]).	WHERE au_lname LIKE '[C-P]arsen' findet alle Autorennachnamen, die auf 'arsen' enden und mit einem einzelnen Zeichen zwischen C und P beginnen, z. B. Carsen, Larsen, Karsen usw. In Bereichssuchvorgängen unterscheiden sich die im Bereich enthaltenen Zeichen möglicherweise je nach den Sortierungsregeln für die Sortierung.
[^]	Beliebiges einzelnes Zeichen, das sich nicht im angegebenen Bereich ([^a-f]) oder in der angegebenen Menge ([^abcdef]) befindet.	

2.13 Nested Table Expression bzw. Derived Table, Common Table Expression

```
SELECT
  preis*1.07   AS preisbrutto
, buchnr       AS buchnr
, preis        AS preisnetto
FROM tbuch
WHERE (preis * 1.07) > 100.00
ORDER BY preisbrutto, buchnr
;
  ------------------------------------------
  --Nested Table Expression bzw. Derived Table
SELECT
 zwi.preisbrutto  AS preisbrutto
,zwi.buchnummer       AS buchnr
,zwi.preisnetto   AS preisnetto
FROM
(
  SELECT
   buchnr        AS buchnummer
  ,preis         AS preisnetto
  ,preis*1.07    AS preisbrutto
  FROM tbuch
) zwi
WHERE zwi.preisbrutto > 100.00
ORDER BY preisbrutto, buchnr
;
  ------------------------------------------
  --Common Table Expression
WITH
zwi AS
(
  SELECT
   buchnr        AS buchnummer
  ,preis         AS preisnetto
  ,preis*1.07    AS preisbrutto
  FROM tbuch
)
SELECT
 zwi.preisbrutto  AS preisbrutto
,zwi.buchnummer       AS buchnr
,zwi.preisnetto   AS preisnetto
FROM zwi
WHERE zwi.preisbrutto > 100.00
ORDER BY preisbrutto, buchnr
;
```

Eine query_table_expression (Oracle Terminologie) bzw. ein fullselect (IBM DB2 Terminologie) kann mit Hilfe von Klammern in die FROM-Klausel eingebettet werden.

Achtung: bei DB2 und SQL Server muss ein Name vergeben werden, AS ist optional.

```
... FROM ( … ) zwi
... FROM ( … ) AS zwi
```

Achtung: bei Oracle ist ein Name nicht zwingend erforderlich, die syntaktische Variante mit dem Syntaxelement AS ist aber nicht möglich!

```
... FROM ( … ) zwi
... FROM ( … ) AS zwi
... FROM ( … )
```

Dieses Konstrukt wird nested table expression (IBM DB2 Terminologie) bzw. derived_table (Microsoft SQL Server Terminologie) genannt.

Mit Hilfe einer Common Table Expression kann die TABLE zwi auch vor der eigentlichen SELECT-Anweisung codiert werden.

```
WITH
zwi AS
(
   ...
)
SELECT * FROM zwi
;
```

Beachten Sie bitte die Formulierungen in den WHERE-Klauseln.

```
WHERE (abc.preis * 1.07) > 100.00


WHERE zwi.preisbrutto > 100.00
```

`abc.preis * 1.6` ist ein numerischer Ausdruck (engl. Expression).

Eine ausführliche Diskussion zu arithmetischen Operatoren, Funktionen und Expressions finden Sie im **Kapitel über Skalare Operatoren und Funktionen, Expression, CASE, CAST**.

2.14 Aggregatfunktionen SUM AVG MAX MIN COUNT

-- Summe und Durchschnitt aller Preise

```
SELECT SUM(Preis) AS Su , AVG(Preis) AS Av
FROM Tbuch;
Su                                Av
------------------------------ --------------------
379.47                            54.210000
```

-- maximaler Preis von 1990

```
SELECT MAX(Preis) AS Ma
FROM Tbuch WHERE Erschj = 1990;
Ma
---------
55.00
```

-- Anzahl der wohldefinierten Werte in der Spalte Erschj

-- Anzahl der verschiedenen wohldefinierten Werte in der Spalte Erschj

-- Anzahl der Zeilen in der Table

-- Anzahl der wohldefinierten Werte in der Spalte Buchnr

```
SELECT
      COUNT (Erschj)           AS sp1
     ,COUNT (DISTINCT ERschj) AS sp2
     ,COUNT (*)               AS sp3
     ,COUNT (Buchnr)          AS sp4
FROM Tbuch;
sp1         sp2         sp3         sp4
----------- ----------- ----------- -----------
8           4           11          11
```

-- Wieviel Bücher haben kein Erscheinungsjahr?

```
SELECT COUNT (*) AS anzernull
FROM Tbuch WHERE erschj IS NULL;
anzernull
-----------
3
```

- SUM und AVG für numerische Argumente, NULL-Marken werden nicht berücksichtigt.
- MAX und MIN ist auch für CHAR und VARCHAR anwendbar.
- COUNT (DISTINCT Erschj) "Anzahl der verschiedenen wohldefinierten Erscheinungsjahre".
- COUNT (Erschj) "Anzahl der wohldefinierten Erscheinungsjahre".
- COUNT(*) ist Spezialfunktion um die Anzahl von Zeilen zu zählen.
- Statt COUNT(*) kann auch der Primärschlüssel Buchnr beim Zählen eingesetzt werden!

```
-- Anzahl der Bücher teurer als 10.00
SELECT COUNT (*) AS An
FROM Tbuch WHERE Preis > 10.00;
SELECT COUNT (buchnr) AS An
FROM Tbuch WHERE Preis > 10.00;
An
-----------
5
```

SQL hat ein ziemlich seltsames Verhalten bei den Aggregatfunktionen! Auch wenn alle Spalten der Table mit NOT NULL definiert sind kann SQL NULLs präsentieren!

```
SELECT
   max(tbuch.preis)     AS maxpreis
  ,min(tbuch.preis)     AS minpreis
  ,sum(tbuch.preis)     AS sumpreis
  ,count(tbuch.preis)   AS countpreis
  ,avg(tbuch.preis)     AS avgpreis
FROM tbuch
WHERE
          tbuch.preis IS NOT NULL
      AND tbuch.erschj = 2000

maxpreis  minpreis  sumpreis countpreis  avgpreis
--------- --------- -------- ----------- --------
NULL      NULL      NULL     0           NULL
```

„Die Summe über eine leere Menge von Preisen müsste eigentlich die Zahl 0 ergeben!“

2.15 SELECT ... GROUP BY …

pro Erscheinungsjahr die Summe der Preise, der maximale Preis und die Anzahl der Bücher

```
SELECT Erschj          AS Erschj
     , SUM(Preis)      AS Supr
     , MAX(Preis)      AS Mapr
     , COUNT(Buchnr)   AS Anzbue
FROM Tbuch GROUP BY Erschj
;
```

```
Erschj Supr                          Mapr      Anzbue
------ ------------------------------ --------- -----
NULL   99.99                          99.99     3
1977   .50                            .50       1
1988   24.00                          20.50     2
1989   199.98                         99.99     3
1990   55.00                          55.00     2
```

Anzahl der Bücher mit gleichem Jahr und Preis

```
SELECT Erschj, Preis, COUNT(Buchnr) AS Anzerpr
FROM Tbuch
GROUP BY Erschj, Preis
;
```

```
Erschj Preis     Anzerpr
------ --------- -----------
NULL   NULL      2
NULL   99.99     1
1977   .50       1
1988   3.50      1
1988   20.50     1
1989   NULL      1
1989   99.99     2
1990   NULL      1
1990   55.00     1
```

- die „NULL-Marken“ der Spalte Tbuch.Erschj bilden eine eigene Gruppe
- die Spalten für die verdichteten Daten erhalten einen Namen
- COUNT(*) ... mit GROUP BY zählt die Zeilen in einer Gruppe
- ***alle Spalten nach SELECT werden mit Spaltenfunktionen ermittelt, mit Ausnahme der GROUP BY Spalte(n).***
- die folgenden SELECT-Anweisungen liefern dasselbe:

```
SELECT DISTINCT Erschj FROM Tbuch;
```

```
SELECT Erschj FROM Tbuch GROUP BY Erschj;
```

- Die folgende Formulierung ist **syntaktisch illegal:**

```
SELECT Erschj, SUM(Preis) FROM Tbuch
       WHERE Erschj = 1990;
```

Auch bei GROUP BY ist nur bei Verwendung von ORDER BY sichergestellt, dass die Daten in einer gewünschten Sortierfolge angelistet werden!

2.16 GROUP BY und HAVING-Klausel, abgeleitete Table und WHERE-Klausel

Pro Erscheinungsjahr der maximale Preis, die Anzahl der Bücher, aber nur der Jahre mit 2 Büchern

```
SELECT zwitab.Erschj AS Erschj
      ,zwitab.Mapr   AS Mapr
      ,zwitab.Anzbue AS Anzbue
FROM
 (
  SELECT Erschj           AS Erschj
       , MAX(Preis)       AS Mapr
       , COUNT(buchnr)    AS Anzbue
  FROM Tbuch GROUP BY Erschj
 ) Zwitab
WHERE Zwitab.Anzbue = 2
;
Erschj Mapr      Anzbue
------ --------- -----------
1988   20.50     2
1990   55.00     2
```

alternative Formulierung:

```
SELECT Erschj           AS Erschj
     , MAX(Preis)       AS Mapr
     , COUNT(buchnr)    AS Anzbue
FROM Tbuch GROUP BY Erschj
HAVING COUNT(buchnr) = 2
;
```

In der FROM-Klausel taucht jetzt eine abgeleitete Table auf (oft auch Nested-Table-Expression genannt). Dieser Table mit den verdichteten Daten wird ein Name gegeben. Mit Hilfe der WHERE-Klausel können jetzt Gruppen ausgewählt werden.

```
SELECT ....
FROM
   ( SELECT ... FROM ... GROUP BY  )
   Zwitab
WHERE
```

Alternative Formulierung mit HAVING! Mit HAVING können Gruppen ausgewählt werden.

```
SELECT ... FROM ... GROUP BY HAVING
```

Beachten Sie bitte bei den folgenden SELECT-Anweisungen, dass im Rahmen einer HAVING-Variante eine gewisse Redundanz in der Formulierung der Anweisung notwendig ist!

```
select * from
(
  SELECT Erschj                   AS Erschj
       , MAX(Preis)               AS Mapr
       , MIN(preis)               AS Mipr
       , MAX(preis) - MIN(preis) AS MA_MI
  FROM Tbuch GROUP BY Erschj
) zwitab
where zwitab.MA_MI > 10.00
;
SELECT Erschj                   AS Erschj
     , MAX(Preis)               AS Mapr
     , MIN(preis)               AS Mipr
     , MAX(preis) - MIN(preis) AS MA_MI
FROM Tbuch GROUP BY Erschj
HAVING Max(preis) - Min(preis) > 10.00
;
Erschj Mapr      Mipr      MA_MI
------ --------- --------- ----------
1988   20.50     3.50      17.00
```

2.17 Variationen mit GROUP BY

„Was kosten alle Bücher von 1990?“

--Variante1 erst verdichten, dann einschränken

```
select
       zwitab.erschj as erschj
     , zwitab.supr   as supr
FROM
(
SELECT
       Tbuch.Erschj     AS Erschj
     , SUM(Tbuch.Preis) AS Supr
FROM Tbuch
GROUP BY Tbuch.Erschj
) zwitab
WHERE zwitab.Erschj = 1990
;
```

--Variante2 erst verdichten, dann einschränken

```
SELECT
       Tbuch.Erschj     AS Erschj
     , SUM(Tbuch.Preis) AS Supr
FROM Tbuch
GROUP BY Tbuch.Erschj
HAVING Tbuch.Erschj = 1990
;
```

--Variante3 erst einschränken, dann verdichten

```
SELECT
       Tbuch.Erschj     AS Erschj
     , SUM(Tbuch.Preis) AS Supr
FROM Tbuch
WHERE tbuch.Erschj = 1990
GROUP BY Tbuch.Erschj
;
```

--Variante4 erst einschränken, dann verdichten

```
SELECT
      Zwitab.erschj     as erschj
     ,SUM(zwitab.preis) as supr
FROM
(
SELECT
       Tbuch.Erschj     AS Erschj
     , Tbuch.Preis     AS Preis
FROM Tbuch
WHERE tbuch.Erschj = 1990
) zwitab
GROUP BY zwitab.Erschj
;
```

"Buchnr von Büchern mit genau einer wohldefinierten Praemie"

```
SELECT
tvautor.buchnr as buchnr
FROM tvautor
GROUP BY tvautor.buchnr
HAVING COUNT(Tvautor.praemie) = 1
;
SELECT
tvautor.buchnr as buchnr
FROM tvautor
WHERE Tvautor.Praemie IS NOT NULL
GROUP BY tvautor.buchnr
HAVING COUNT(tvautor.autornr) = 1
;
SELECT
tvautor.buchnr as buchnr
FROM tvautor
WHERE Tvautor.Praemie IS NOT NULL
GROUP BY tvautor.buchnr
HAVING COUNT(*) = 1
;
buchnr
-----------
8
12
```

„Welche Fremdschlüsselwerte sind genau zweimal vorhanden?“

"Buchnr von Büchern mit genau zwei Autoren"

```
SELECT
tvautor.buchnr as buchnr
FROM Tvautor
GROUP BY tvautor.buchnr
HAVING COUNT(tvautor.autornr) =2
;
```

"Buchnr von Büchern mit genau zwei ISBNs"

```
SELECT
tisbn.buchnr as buchnr
FROM tisbn
GROUP BY tisbn.buchnr
HAVING COUNT(tisbn.isbn) =2
;
```

2.18 NULL-Werte sind gleich – NULL-Werte sind nicht gleich

	Preisalt	Preisneu	
	3.50	3.50	
	99.99	3.50	
	NULL	3.50	
	NULL	NULL	

```
WHERE
  Preisalt = Preisneu
  OR
 (Preisalt IS NULL AND Preisneu IS NULL)
```

x ope y hat den Wahrheitswert **UNKNOWN**
wenn x NULL ist oder y NULL ist oder beide NULL sind
(wobei ope Vergleichsoperator = < > <= ...)

x = x hat den Wahrheitswert **TRUE** oder **UNKNOWN**
("that 'null=null' does not evaluate to true ... lies at the root of most of the problems that arise over null"
C. J. DATE Relational Database: Selected Writings 1986 s 316)

NULL-Werte sind nicht gleich bezüglich

- WHERE-Klausel
- HAVING-Klausel

NULL-Werte sind aber gleich bezüglich

- DISTINCT Duplikateliminierung
- ORDER BY
- GROUP BY
- UNION
- INTERSECT
- EXCEPT bzw. MINUS Oracle

2.19 NULL ist etwas anderes als UNKNOWN

	Name	Antwort	
	Otto	wahr	
	Emil	falsch	
	Fritz	unbekannt	
	Franz	NULL	

NULL (der Wahrheitswert ist im System **nicht definiert**)
ist etwas anderes als
unbekannt (der Wahrheitswert ist im System **definiert**).

Der Befragte geht bei der nächsten Wahl ins Wahllokal.
Otto: Ja, ich gehe ins Wahllokal.
Wahrheitswert **wahr**, im System definiert!

Emil: Nein, ich gehe nicht ins Wahllokal.
Wahrheitswert **falsch**, im System definiert!

Fritz: Ich weiß es noch nicht.
Wahrheitswert **unbekannt**, im System definiert!

Franz:
Wahrheitswert ist im System nicht definiert, d. h. **NULL**
"Franz konnte nicht befragt werden."

2.20 IS TRUE, IS FALSE, IS UNKNOWN wird von wenigen Produkten am Markt unterstützt

NOT (Erschj = 1990)
ist false oder true oder unknown,

und entspricht damit also auf keinen Fall

(Erschj = 1990 IS *NOT* TRUE)
ist false oder true

NOT ist eine andere Negation als ***NOT***

NOT NOT NOT NOT NOT NOT NOT NOT NOT NOT NOT NOT NOT

```
... WHERE Erschj = 1990
```
ist true oder false oder unknown

```
... WHERE NOT (Erschj = 1990)
```
ist false oder true oder unknown

NOT NOT NOT NOT NOT NOT NOT NOT NOT NOT NOT NOT NOT

SQL kennt zusätzlich die folgenden Sprachelemente:

NOT NOT NOT NOT NOT NOT NOT NOT NOT NOT NOT NOT NOT

```
... WHERE Erschj = 1990 IS TRUE
... WHERE Erschj = 1990 IS FALSE
... WHERE Erschj = 1990 IS UNKNOWN
```
sind immer nur true oder false

```
... WHERE     Erschj = 1990  IS NOT UNKNOWN
```
entspricht

```
... WHERE NOT (Erschj = 1990 IS     UNKNOWN)
```
entspricht

```
... WHERE     (Erschj = 1990 IS TRUE)
          OR (Erschj = 1990 IS FALSE)
```
NOT NOT NOT NOT NOT NOT NOT NOT NOT NOT NOT NOT NOT

- Missing Information and Nulls: "CAVEAT: It is only fair to the reader to explain right at the outset that the topic of this chapter CANNOT be described in a manner that is simultaneously comprehensive and comprehensible. The reason is that both the theoretical ideas on which the relevant SQL features are based – even more so – those SQL features per se, are themselves not consistent ..." C. J. Date with Hugh Darwen, A Guide to The SQL Standard, 1993, Seite 219
- SQL kennt **drei Wahrheitswerte**: true, false, unknown
- **Die entsprechenden Literale sind TRUE, FALSE, UNKNOWN.**
- Der von SQL:1999 beschriebene neue Datentyp BOOLEAN mit seinen Operatoren NOT, AND, OR sieht nur zwei Wahrheitswerte vor. Der Wahrheitswert *unknown* wird – seltsamerweise, fälschlicherweise – durch NULL repräsentiert!

  "... the type BOOLEAN includes only two values, not three; the unknown truth value is represented – quite incorrectly! – by *null* (for example, assigning UNKNOWN to a variable of type BOOLEAN will actually set that variable to null)..." C. J. DATE An Introduction to Database Systems, Addison-Wesley Publishing Company Seventh Edition 2000, Seite 901
- **Überprüfen Sie die Implementierung Ihres Servers, sofern dieser schon einen boolschen Datentyp kennt!**
- Bei ORACLE PL/SQL kann eine Variable des Datentyps BOOLEAN die Werte **TRUE** und **FALSE** annehmen (also zwei Wahrheitswerte). Zusätzlich ist "Zustand nicht bestimmt – Wert NULL" möglich. Der dritte Wahrheitswert **UNKNOWN** kann nicht repräsentiert werden!

2.21 SCHEMA und TABLE, schema-name und table-name

-- Buchnr und Titel aller Bücher von 1990

```
SELECT Buchnr, Titel
FROM Tbuch
WHERE Erschj = 1990
;
Buchnr      Titel
----------- ----------------------------------------
9           DB2 fuer Sie
11          Elvis in Heidelberg
```

Ein Schema ist eine mit einem Namen versehene Menge von Dingen wie zum Beispiel Tables. Der Name einer Table muss innerhalb eines Schemas eindeutig sein.

In den Beispielen sind skis41t bzw. schema07 jeweils Schema-Namen. Es kann mehrere Tables geben mit dem Table-Namen Tbuch.

Die Referenz auf eine Spalte sollte mit dem Korrelationsnamen, oft auch Aliasname genannt, qualifiziert werden. Im den Beispielen ist „abc“ bzw. „xyz“ ein Korrelationsname. **„AS abc“** ist möglich bei DB2 und SQL Server, aber nicht bei bei Oracle

```
SELECT xyz.Buchnr AS Buchnr
     , xyz.Titel  AS Titel
FROM skis41t.Tbuch xyz
WHERE xyz.Erschj = 1990
;
SELECT abc.Buchnr AS Buchnr
     , abc.Titel  AS Titel
FROM schema07.Tbuch abc
WHERE abc.Erschj = 1990
;
```

CREATE DATABASE ist im SQL Standard nicht definiert. Eine Database hat in den verschiedenen Datenbanksystemen unterschiedliche Bedeutung.

Situation 1 **schema-name.table-name**

Bei Oracle und Db2 for z/OS ist auf der Ebene einer SELECT-Anweisung kein database-name sichtbar!

Das heißt:: im Rahmen eines laufenden Systems ist eine Table eindeutig identifizierbar mit Hilfe von **schema-name.table-name**

```
SELECT abc.Buchnr AS Buchnr
     , abc.Titel  AS Titel
FROM schema07.Tbuch abc
WHERE abc.Erschj = 1990
;
```

Situation 2 database-name.schema-name.table-name

Bei SQL Server und Db2 for Windows, Linux, Unix ist auf der Ebene einer SELECT-Anweisung ein database-name sichtbar!

Das heißt:: im Rahmen eines laufenden Systems ist eine Table eindeutig identifizierbar mit Hilfe von database-name.schema-name.table-name

```
SELECT abc.Buchnr AS Buchnr
     , abc.Titel  AS Titel
FROM sample01.schema07.Tbuch abc
WHERE abc.Erschj = 1990
;
--dieses Coding ist etwas "umständlich"
SELECT sample01.schema07.Tbuch.Buchnr AS Buchnr
     , sample01.schema07.Tbuch.Titel  AS Titel
FROM   sample01.schema07.Tbuch
WHERE  sample01.schema07.Tbuch.Erschj = 1990
;
```

3

Skalare Operatoren und Funktionen, Expression, CASE, CAST

3 Skalare Operatoren und Funktionen, Expression, CASE, CAST

3.1 SELECT ... expression AS name ...

```
SELECT Buchnr         AS Buchnr
      ,Preis          AS Netto
      ,'Bruttopreis' AS xxx
      ,Preis * 1.2    AS Brutto
FROM Tbuch
ORDER BY Buchnr
;
```

```
Buchnr      Netto     xxx          Brutto
----------- --------- ------------ ------------
1           NULL      Bruttopreis NULL
2           NULL      Bruttopreis NULL
5           3.50      Bruttopreis 4.200
6           20.50     Bruttopreis 24.600
7           99.99     Bruttopreis 119.988
8           .50       Bruttopreis .600
9           55.00     Bruttopreis 66.000
11          NULL      Bruttopreis NULL
12          NULL      Bruttopreis NULL
18          99.99     Bruttopreis 119.988
27          99.99     Bruttopreis 119.988

(11 row(s) affected)
```

3.2 Expression Ausdruck

```
Preis
(Preis+100)/75.2
CAST(Preis + 0.50 AS DECIMAL(6,0))
CAST (Preis AS CHAR(12))

COALESCE (Preis, 0.00)
COALESCE (Preis, -97599.12)
COALESCE (  CAST (Preis AS CHAR(12)), 'nixda' )
Oracle NVL
DB2 VALUE

50 - AVG(Preis)/100
(SELECT AVG(Preis) FROM Tbuch)

Titel
SQL Standard
    SUBSTRING (TITEL FROM 1 FOR 5)
SQL Server
    SUBSTRING (TITEL, 1, 5)
DB2
    SUBSTRING(Titel, 1, 5, CODEUNITS16)
    SUBSTRING(Titel, 1, 5, CODEUNITS32)
    SUBSTRING(Titel, 1, 5, OCTETS     )
    SUBSTR (TITEL, 1, 5),
Oracle
    SUBSTR (TITEL, 1, 5), SUBSTRB, SUBSTRC, SUBSTR2, SUBSTR4

SQL Standard
    'neu: ' || Titel
DB2
    'neu: ' ||     Titel
    CONCAT( 'neu: ' ,  Titel )
    'neu: ' CONCAT Titel
Oracle
    'neu: ' ||     Titel
    CONCAT( 'neu: ' ,  Titel )
SQL Server
    'neu: ' + Titel
    CONCAT( 'Heinrich' , ' Böll')

LOWER('der Butt')
TRIM ('  Test     ')
UPPER('der Butt')
USER

(SELECT MAX(Preis) FROM Tbuch)

( SELECT COUNT(*) FROM Tisbn
  WHERE Tisbn.Buchnr = Tbuch.Buchnr )

( select sum(xyz.praemie) from tvautor xyz
  where xyz.buchnr = abc.buchnr )
```

Arithmetische Operatoren mit Klammern) bzw. (

+ Addition

- Subtraktion

*** Multiplikation**

/ Division

Ist einer der Operanden NULL, dann ist auch das Ergebnis der arithmetischen Operation NULL.

- Mit Hilfe der Operatoren und Funktionen können Expressions, Ausdrücke konstruiert werden. Die Operanden einer Expression sind einfache Skalarwerte und eine Expression liefert selbst wieder einen Skalarwert.
- **Mit gewissen Einschränkungen kann eine solche Expression überall auftreten, wo auch ein Literal des entsprechenden Typs möglich ist (z.B. in der Select-Klausel, Where-Klausel bzw. Having-Klausel, in der Set-Klausel beim UPDATE, in der Set-Klausel beim MERGE, beim INSERT).**
- Auch die Spaltenfunktionen können innerhalb von gewissen Expressions benützt werden, da sie einen Skalarwert zurückgeben.
- **Eine Subquery, die einen Skalarwert zurückgibt, ist auch eine Expression!** Eine Subquery ist (syntaktisch betrachtet) vereinfacht gesagt eine Select-From-Where-Anfrage in Klammern.
- **Achtung: Expressions in der Where-Klausel führen zu einem sequentiellen Zugriff, die optimale Nutzung eines eventuell vorhandenen Indices ist nicht möglich!**

```
SELECT * FROM Tbuch WHERE Buchnr + 0  = 8
;
SELECT * FROM Tbuch WHERE Preis/2 = 5.20
;
SELECT * from Tbuch WHERE LOWER(Titel) = 'der butt'
;
```

3.3 Oracle und DUAL, DB2 und SYSIBM.SYDUMMY1, SQL Server und ???

Beim SQL SERVER muss die FROM-Klausel nicht spezifiziert werden!

```
SELECT 12345 AS Wert

Wert

-----------

12345

(1 row(s) affected)
```

Oracle und DUAL

Die Table DUAL ist eine Data Dictionary Table von Oracle mit nur einer Spalte DUMMY CHAR(1) und einer Zeile mit dem Wert "X".

```
SELECT * FROM DUAL;

D

-

X

1 row selected.
```

DB2 und SYSIBM.SYDUMMY1

Analoges wie für Oracle und die Table DUAL gilt für DB2 und die Table SYSIBM.SYSDUMMY1.

```
SELECT * FROM SYSIBM.SYSDUMMY1
IBMREQD
-------
Y
1 Satz/Sätze ausgewählt.
```

```
SELECT GETDATE()
;
SELECT CURRENT_DATE
FROM DUAL
;
SELECT CURRENT_DATE
FROM SYSIBM.SYDUMMY1
;
```

3.4 CASE

```
SELECT
       Buchnr
      ,Titel
      ,CASE
             WHEN Preis = 5.00   THEN 'fünf'
             WHEN Preis = 6.00   THEN 'sechs'
             WHEN Preis = 7.00   THEN 'sieben'
             ELSE 'weder 5 noch 6 noch 7'
       END AS Kommentar
FROM Tbuch
;
```

Bücher, die noch mehr als 10.00 kosten, wenn entsprechender Rabatt abgezogen ist.

```
SELECT
        Buchnr
      , Titel
      , Erschj
      , Preis AS Normalpreis
      , CASE
             WHEN Erschj < 1990 THEN Preis - 3.00
             WHEN Erschj = 1990 THEN Preis - 2.00
             ELSE                    Preis - 1.00
        END AS Sonderpreis
FROM TBUCH
WHERE (
        CASE
           WHEN Erschj < 1990 THEN (Preis - 3.00)
           WHEN Erschj = 1990 THEN (Preis - 2.00)
            ELSE                   (Preis - 1.00)
        END
       )         > 10.00
;
```

NULLIF(x,y)

ist äquivalent zu

CASE WHEN x=y THEN NULL ELSE x END

COALESCE(x,y)

ist äquivalent zu

CASE WHEN x IS NOT NULL THEN x ELSE y END

Oracle kennt DECODE:

```
SELECT
   buchnr               as buchnr
  ,substr(titel, 1, 5) as titel
  ,DECODE ( tbuch.preis , 5.00 , 'fünf'
                        , 6.00 , 'sechs'
                        , 7.00 , 'sieben'
                        , 'weder 5 noch 6 noch 7'
          )             as kommentar
FROM tbuch
;
```

CASE in der WHERE-Klausel oder Trickprogrammierung

```
drop table taball
;
create table taball
( schl integer not null
 ,col1 integer not null
 ,col2 decimal(32) not null
 ,PRIMARY KEY(schl)
 )
 ;
 INSERT INTO taball (schl, col1, col2)
 values             ( 11  ,  2   , 5.)
 ;
 INSERT INTO taball (schl, col1, col2)
 values             ( 22  ,  0   , 5.)
 ;
 INSERT INTO taball (schl, col1, col2)
 values             ( 33  , 20   , 5.)
 ;
 select
 schl, col1, col2
 from taball
 ;
schl        col1        col2
----------- ----------- ---------
11          2           5
22          0           5
33          20          5
```

Aufgabe:

Es sollen alle Zeilen angelistet werden, für die col2/col1 > 2

```
schl        col1        col2
----------- ----------- ---------
11          2           5
```

```
 --divide-by-zero error
 select
 schl, col1, col2, col2/col1
 from taball
 ;
```

```
--Versuch:
--ein divide-by-zero error ist möglich
--die Reihenfolge der Verarbeitung
--der Bedingungen in der WHERE-Klausel
--ist nicht festgelegt
select
schl, col1, col2,  col2/col1 as col3
from taball
where col1<>0 and col2/col1 >2
-- where col2/col1 >2 and col1<>0
;
--Lösung1:
--ein divide-by-zero error ist nicht möglich
--die Reihenfolge der Verarbeitung
--der Bedingungen im CASE-Block
--ist festgelegt
select
schl, col1, col2, col2/col1 as col3
from taball
WHERE
  CASE
    WHEN col1 = 0       THEN 'no'
    WHEN col2/col1 >2   THEN 'yes'
ELSE 'no'
END                        = 'yes'
;
--Lösung2:
select
schl, col1, col2, col2/col1 as col3
from taball
WHERE
   (col1 > 0 AND col2 > 2*col1)
OR
   (col1 < 0 AND col2 < 2*col1)
;
```

3.5 CAST

Beispiel SQL Server

```
SELECT 8/3
-----------
2

SELECT 8/3.0
----------
2.666666

SELECT
 6.6666                          AS s1
,cast(6.6666 as decimal (3,2))   AS s2

S1         S2
---------  --------
6.6666     6.67
```

Division von INTEGER-Zahlen

DB2: 8/3 = 2

DB2: 8/3.0 = 2,6666666666666666

ORACLE: 8/3 = 2,66666667

SQL Server: 8/3 = 2

SQL Server: 8/3.0 = 2.666666

CAST

ORACLE rundet

DB2 rundet nicht

SQL Server rundet

```
SELECT CAST(Buchnr/3 AS DECIMAL(6,4))
FROM Tbuch WHERE Buchnr = 8
;
ORACLE     2,6667
DB2        2.0000
SQL Server 2.0000
```

```
SELECT CAST(Buchnr AS DECIMAL(6,4)) / 3
FROM Tbuch WHERE Buchnr = 8
;
ORACLE      2,66666667
DB2         2,666666666666666666666666666666
SQL Server  2.666666
```

```
SELECT CAST (Preis/2 AS DECIMAL(8,6))
FROM TBUCH WHERE Buchnr = 7
;
ORACLE      49,995
DB2         49,995000
SQL Server  49.995000

SELECT CAST (Preis/2 AS DECIMAL(7,2))
FROM TBUCH WHERE Buchnr = 7
;
ORACLE      50,00
DB2         49,99
SQL Server  50.00
```

```
SELECT CAST (Preis/2 AS DECIMAL(8,7))
FROM TBUCH WHERE Buchnr = 7
;
SQL Server
```
Server: Nachr.-Nr. 8115, Schweregrad 16, Status 8, Zeile 1

Arithmetischer Überlauffehler beim Konvertieren von numeric in den Datentyp numeric.

3.5.1 CAST AS DECIMAL

CAST AS DECIMAL SQL Server

```
SELECT CAST ('1234.56'     AS DECIMAL(7,2) )
1234.56

SELECT CAST ('12345'       AS DECIMAL(7,2) )
12345.00

SELECT CAST ('123456'      AS DECIMAL(7,2) )
Arithmetischer Überlauffehler beim Konvertieren von
numeric in den Datentyp numeric.

SELECT CAST ('1234.5678'   AS DECIMAL(7,2) )
1234.57
```

CAST AS DECIMAL Oracle

```
SELECT CAST ('1234.56'     AS DECIMAL(7,2) )
      FROM DUAL;
*
ERROR at line 1:
ORA-01722: invalid number

SELECT CAST ('12345'       AS DECIMAL(7,2) )
FROM DUAL;
12345

SELECT CAST ('123456'       AS DECIMAL(7,2) )
FROM DUAL;
*
ERROR at line 1:
ORA-01438: value larger than specified precision al-
lows for this column
```

CAST AS DECIMAL DB2

```
SELECT CAST ('1234.56'     AS DECIMAL(7,2) )
       FROM SYSIBM.SYSDUMMY1;
1234,56

SELECT CAST ('12345'     AS DECIMAL(7,2) )
       from sysibm.sysdummy1
12345,00

SELECT CAST ('123456'     AS DECIMAL(7,2) )
       from sysibm.sysdummy1
SQL0413N  Datenüberlauf während der Umsetzung numeri-
scher Datentypen.
SQLSTATE=22003

SELECT CAST ('1234.5678'   AS DECIMAL(7,2) )
FROM SYSIBM.SYSDUMMY1;
1234,56
```

3.5.2 CAST AS CHAR, CONVERT und “eine fette Wanze“ SQL Server

CAST SQL Server

„CAST und CONVERT Konvertiert einen Ausdruck explizit von einem Datentyp in einen anderen. Die Funktionalität von CAST und CONVERT ist ähnlich.“

```
SELECT
  CAST(1234567890  AS CHAR(8) )   as 'was?'
,len
 (CAST(1234567890  AS CHAR(8) ) ) as 'laenge?'

was?     laenge?
-------- -----------
*        1
 (1 row(s) affected)


SELECT
  CONVERT(char(8), 1234567890 )  as 'was?'

was?
--------
*
 (1 row(s) affected)


SELECT CAST ('1234567890'  AS CHAR(8) ) as 'das!'
12345678

SELECT CONVERT( CHAR(8), '1234567890' ) as 'das!'
12345678
```

CAST AS CHAR Oracle

```
SELECT CAST (1234567890  AS CHAR(8) )
FROM DUAL;
*
ERROR at line 1:
ORA-25137: Data value out of range
```

CAST AS CHAR DB2

```
SELECT CAST (1234567890  AS CHAR(8) )
FROM SYSIBM.SYSDUMMY1;
12345678
  1 Satz/Sätze ausgewählt, dabei 1 Warnungen unter-
drückt.
```

TO_NUMBER Oracle konvertiert eine Zeichenkette in eine Zahl

```
SELECT TO_NUMBER ('1234.56'   , '99999.99')
FROM DUAL;
1234,56

SELECT TO_NUMBER ('1234.5678' , '99999.99')
FROM DUAL
*
ERROR at line 1:
ORA-01722: invalid number
```

TO_CHAR Oracle konvertiert ein Zahl in eine Zeichenkette.

```
SELECT TO_CHAR (1234567890) FROM DUAL;
1234567890
```

CAST Oracle

```
SELECT CAST (1234567890  AS CHAR(8) ) FROM DUAL
*
ERROR at line 1:
ORA-25137: Data value out of range
```

3.6 CAST und COALESCE

```
SELECT
 Buchnr                      AS Buchnr
,Preis                       AS Preis
,CAST (Preis AS CHAR(9))     AS Preiscast
FROM Tbuch
ORDER BY Buchnr
;
Buchnr      Preis     Preiscast
----------- --------- ---------
1           NULL      NULL
2           NULL      NULL
5           3.50      3.50
6           20.50     20.50
7           99.99     99.99
8           .50       0.50
9           55.00     55.00
11          NULL      NULL
12          NULL      NULL
18          99.99     99.99
27          99.99     99.99

(11 row(s) affected)
```

```
SELECT
 Buchnr                    AS Buchnr
,Preis                     AS Preis
,COALESCE (
           CAST (Preis AS CHAR(10)) , 'kein Preis'
          )                AS Preiscoacast
FROM Tbuch
ORDER BY Buchnr
;
Buchnr      Preis     Preiscoacast
----------- --------- ------------
1           NULL      kein Preis
2           NULL      kein Preis
5           3.50      3.50
6           20.50     20.50
7           99.99     99.99
8           .50       0.50
9           55.00     55.00
11          NULL      kein Preis
12          NULL      kein Preis
18          99.99     99.99
27          99.99     99.99

(11 row(s) affected)
```

Oracle kennt neben COALESCE auch NVL

DB2 kennt neben COALESCE auch VALUE

3.7 Character-Semantik und Byte-Semantik

Das Ermitteln einer Stringlänge in (Anzahl von) Bytes bezeichnen wir als **Byte-Semantik.** Das Ermitteln einer Stringlänge in (Anzahl von) Zeichen bezeichnen wir als **Character-Semantik.**

Die Endbenutzer von SQL sind, was die String-Funktionen betrifft, an der Character-Semantik interessiert!

Was ist die Semantik der entsprechenden String-Funktionen bei **SQL Server:** LEN, DATALENGTH, SUBSTRING etc.

```
Byte-Semantik     : DATALENGTH
Character-Semantik: SUBSTRING LEN
```

Zitat Anfang

Returns part of a character, binary, text, or image expression.
SUBSTRING (value_expression , start_expression, length_expression)
value_expression
Is a character, binary, text, ntext, or image expression.
start_expression
Is an integer or bigint expression that specifies where the returned characters start....
length_expression
Is a positive integer or bigint expression that specifies how many characters of the *value_expression* will be returned.
Zitat Ende
Zitat aus der SQL Server Dokumentation

Wie ermitteln wir die Länge einer Zeichenkette? Mit Hilfe von Trickprogrammierung!

```
select
 verlagnr, verlag
,LEN(Verlag)              as length_ohneblanksrechts
,LEN(verlag + '***') - 3  as length_des_strings
from tverlag
```

Demonstration DATALENGTH, LEN, SUBSTRING

```
DECLARE @string_nchar    NCHAR(20)
DECLARE @string_nvarchar NVARCHAR(20)

SET @string_nchar    =  N'Å1Å2Å3456'
SET @string_nvarchar =  N'Å1Å2Å3456'

print @string_nchar + '***'
print @string_nvarchar + '***'

select
 @string_nchar                   as kette_nchar
,LEN(@string_nchar)              as kette_nchar_len
,DATALENGTH(@string_nchar )      as kette_nchar_datalength
,SUBSTRING( @string_nchar, 1, 3) as kette_nchar_substring

select
 @string_nvarchar                   as kette_nvarchar
,LEN(@string_nvarchar)              as kette_nvarcharlen
,DATALENGTH(@string_nvarchar )      as kette_nvarchardatalength
,SUBSTRING( @string_nvarchar, 1, 3) as kette_nvarchar_substring
```

*****) Byte-Semantik und Character-Semantik**

„In the case of a single-byte character encoding scheme, a single byte constitutes a character and the length of a single byte string is the same as the byte length of the string…However, in the case of a multi-byte encoding, the length of the character in bytes varies according to the encoding used, and each character can be one or more bytes in length.“ (Zitat aus: Character–based string functions in DB2 9, Kiran Nair, 24 May 2007)

DB2: *LENGTH*, SUBSTR, OCTET_LENGTH, SUBSTRING, CHAR_LENGTH, POSITION, LOCATE etc.

Byte-Semantik: ***LENGTH***, **SUBSTR**, OCTET_LENGTH
Character-Semantik: SUBSTRING, CHAR_LENGTH, POSITION

Oracle: *LENGTH*, LENGTHB, LENGTH2, LENGTH4, LENGTHC, SUBSTR, SUBSTRB, SUBSTR2, SUBSTR4, SUBSTRC etc.

Byte-Semantik: LENGTHB, SUBSTRB
Character-Semantik: ***LENGTH, SUBSTR***
“INSTR, LENGTH, and SUBSTR Functions and Character Sets
When using string manipulation functions, you can get different results depending upon the database character set. In particular, the INSTR, LENGTH, and SUBSTR functions can return incorrect results because of the difference between single and multibyte character sets. To guarantee correct results, you should use variants of these functions designed for multibyte character sets.” Zitat aus der Oracle Dokumentation

3.8 Syntax und Semantik

- Die einzelnen Produkte implementieren – abhängig von der jeweiligen Version – unterschiedliche Mengen von Operatoren und Funktionen.
- Die einzelnen Produkte implementieren einige der Funktionen des Standards. **Die Syntax ist normiert, die Semantik ist aber oft implementation-defined (vergleichen Sie dazu bitte als Beispiel die CAST-Funktion).**
- Die einzelnen Produkte implementieren einige der Funktionen die im Standard definiert sind, benutzen dabei aber eine andere Syntax!
- Die einzelnen Produkte implementieren viele Funktionen die nicht im Standard definiert sind!
- **Testen Sie Ihren Server! Testen Sie die neue Version Ihres Servers!**

SQL Server Beispiel REPLACE

```
SELECT REPLACE ('Böll', 'l', 'm')
Bömm
```

Oracle Beispiel TRANSLATE

```
FROM l TO m
FROM m TO l
SELECT TRANSLATE ('Böll', 'l', 'm') FROM DUAL;
Bömm
SELECT TRANSLATE ('Böll', 'm', 'l') FROM DUAL;
Böll
```

DB2 Beispiel TRANSLATE

```
TO m FROM l
TO l FROM m
SELECT TRANSLATE ('Böll', 'l', 'm')
       FROM sysibm.sysdummy1;
Böll
SELECT TRANSLATE ('Böll', 'm', 'l')
       from sysibm.sysdummy1;
Bömm
```

Beachten Sie bitte den Unterschied:

Oracle: FROM TO bzw. DB2: TO FROM

3.9 Skalare Operatoren und Funktionen, der SQL Standard,

Die Beschreibung der skalaren Operatoren und Funktionen ist teilweise wörtlich dem folgenden, ausgezeichneten Buch entnommen:

SQL – Der Standard SQL/92 mit den Erweiterungen CLI und PSM Deutsche Ausgabe des amerikanischen Klassikers Ausblick auf SQL3 Chris J. Date Hugh Darwen Addison Wesley Longman GmbH, 1998

*) Für die genaue Beschreibung von COLLATE, CONVERT USING, TRANSLATE USING und den damit zusammenhängenden Begriffen CHARACTER SET (Zeichenmenge) und COLLATION (Zeichenordnung) wird der Leser auf dieses Buch, auf die offiziellen Dokumente bzw. auf die Manuals des jeweiligen Produkts verwiesen.

Operator/Funktion	Beschreibung
ABS	ABS gibt den absoluten Wert eines numerischen Ausdrucks zurück. SQL:1999
Arithmetische Operatoren + – * /	Die üblichen arithmetischen Operatoren +, -, * und / werden mit ihrer jeweils üblichen Bedeutung unterstützt. Seien x und y jeweils von einem gewissen numerischen Datentyp. Dann wir, wenn x NULL ist, jede Auswertung der Ausdrücke +x und –x als NULL definiert. Ist x NULL oder y NULL oder beide, dann wird jede Auswertung der Ausdrücke x+y x-y x*y x/y ebenfalls als NULL definiert.
BIT_LENGTH	BIT_LENGTH gibt die Länge einer gegebenen Zeichenkette oder Bitkette in Bits zurück. Die fragliche Kette ist mit Hilfe eines beliebigen Kettenausdrucks angegeben.
CARDINALITY	CARDINALITY liefert die Anzahl der Elemente eines Arrays zurück. SQL:1999
CASE	Eine CASE-Operation gibt, abhängig von einer Bedingung, einen Wert aus einer angegebenen Menge von Skalarwerten zurück.
CAST	CAST wandelt einen gegebenen Skalarwert in einen gegebenen skalaren Datentyp um.
CEIL CEILING	CEIL bzw. CEILING gibt die kleinste ganze Zahl zurück die größer oder gleich ist als ein gegebener numerischer Ausdruck. SQL:2003

Operator/Funktion	Beschreibung
CHAR_LENGTH	CHAR_LENGTH gibt die Länge einer gegebenen Kette in Zeichen oder in „Oktetten“ zurück. Die fragliche Kette ist mit Hilfe eines beliebigen Kettenausdrucks angegeben; das Ergebnis ist die Länge in Zeichen, wenn die Kette eine Zeichenkette ist, ansonsten die Länge in Oktetten.
COALESCE	Der Ausdruck COALESCE(x,y) wird als äquivalent zu dem Ausdruck CASE WHEN x IS NOT NULL THEN x ELSE y END definiert. Allgemeiner ausgedrückt: der Ausdruck COALESCE (x, y, ..., z) liefert NULL genau dann, wenn seine Operanden alle zu NULL ausgewertet werden; ansonsten gibt er den Wert des ersten Operanden ungleich NULL zurück.
COLLATE *)	
CONVERT USING *)	
CURRENT_USER	CURRENT_USER gibt eine Zeichenkette mit dem aktuellen Berechtigungsschlüssel (authID) zurück.
EXP	Exponentialfunktion SQL:2003
FLOOR	FLOOR gibt die größte ganze Zahl zurück die kleiner oder gleich ist als ein gegebener numerischer Ausdruck. SQL:2003
LN	natürlicher Logarithmus SQL:2003
LOWER	LOWER und UPPER gibt eine Zeichenkette zurück, die mit der gegebenen Zeichenkette identisch ist bis auf: alle Großbuchstaben sind durch ihre entsprechenden Kleinbuchstaben ersetzt (bei LOWER) bzw. umgekehrt (bei UPPER).

MOD	MOD gibt den Rest einer Division zweier Integer-Werte zurück, der erste Wert sei größer oder gleich 0, der zweite Wert sei echt größer 0.
	`MOD( 11, 4)  ergibt  3`
	`MOD(  0, 4)  ergibt  0`
	MOD verhält sich aber sonst bei den verschiedenen Produkten jeweils in einer proprietären Weise (systemabhängig, implementation-dependent)!
	`MOD( 11, -4)  ergibt ?`
	`MOD(-11,  0)  ergibt ?`
	SQL:1999

Operator/Funktion	Beschreibung
NULLIF	Der Ausdruck NULLIF(x,y) wird als äquivalent zu dem Ausdruck CASE WHEN x = y THEN NULL ELSE x END definiert.
OCTET_LENGTH	OCTET_LENGTH gibt die Anzahl einer gegebenen Kette in „Oktetten“ an. Die fragliche Kette kann mit Hilfe eines beliebigen Kettenausdrucks angegeben werden; Ergebnis ist die Bitlänge der Kette dividiert durch 8 (wobei jeglicher Rest der Division ignoriert wird).
OVERLAY	OVERLAY ersetzt innerhalb einer Zeichenkette eine angegebene Teilzeichenkette durch eine andere Zeichenkette. SQL:1999
POSITION	POSITION gibt die Position einer gegebenen Kette innerhalb einer anderen Kette zurück.
POWER	Gibt den Wert eines spezifizierten numerischen Ausdrucks erhoben zu einer spezifizierten Potenz zurück. SQL:2003
SESSION_USER	SESSION_USER gibt eine Zeichenkette zurück, die den Berechtigungsschlüssel (authID) der SQL-Sitzung enthält.
SQRT	Gibt die Quadratwurzel eines spezifizierten numerischen Ausdrucks zurück. SQL:2003
SUBSTRING	SUBSTRING nimmt eine Teilkette aus einer gegebenen Kette heraus.
SYSTEM_USER	SYSTEM_USER gibt eine Zeichenkette mit der ID des am Betriebssystem angemeldeten Benutzers zurück, der das Modul aufgerufen hat, welches die Referenz auf SYSTEM_USER enthält.
TRANSLATE USING *)	TRANSLATE USING übersetzt eine angegebene Zeichenkette Zeichen für Zeichen in eine andere Zeichenkette gleicher Länge, indem eine vordefinierte Zeichenübersetzung (translation) verwendet wird, die eine Quellzeichenmenge in eine Zielzeichenmenge abbildet.
TRIM	TRIM gibt eine Zeichenkette zurück, die bis auf alle führenden und/oder angehängten Füllzeichen (pad characters) mit der gegebenen Zeichenkette identisch ist.
UPPER	LOWER und UPPER gibt eine Zeichenkette zurück, die mit der gegebenen Zeichenkette identisch ist bis auf: alle Großbuchstaben sind durch ihre entsprechenden Kleinbuchstaben ersetzt (bei LOWER) bzw. umgekehrt (bei UPPER).

Operator/Funktion	Beschreibung
USER	USER gleich mit CURRENT_USER.
Verkettung Konkatenation ||	Der Verkettungsoperator || kann zur Verkettung zweier Zeichen- oder Bitketten verwendet werden.

Operatoren und Funktionen des Standards für Datums- und Zeitangaben	Datentypen des Standards für Datums- und Zeitangaben
CAST Datenumwandlung	DATE
CURRENT_DATE	TIME
CURRENT_TIME	TIME WITH TIME ZONE
CURRENT_TIMESTAMP	TIMESTAMP
EXTRACT	TIMESTAMP WITH TIME ZONE
Aggregatfunktionen	INTERVAL
Arithmetische Operatoren für bestimmte Berechnungen	
Vergleichsoperatoren	
Vergleichsoperator OVERLAPS um zu testen, ob sich zwei Zeiträume überlappen	

Jedes RDBMS am Markt hat seine besonderen Eigenarten in Bezug auf Datums- und Zeitangaben. Vergleichen Sie bitte dazu das Kapitel dieser Broschüre zum Thema Datums- und Zeitangaben.

3.10 Skalare Operatoren und Funktionen, der Standard und die Dialekte

SQL Standard	DB2	ORACLE	SQL Server
ABS	X	X	X
Arithmetische Operatoren + – * /	X	X	X
BIT_LENGTH			
CARDINALITY			
CASE	X	X DECODE	X
CAST	X	X	X CONVERT
CEIL CEILING	CEIL CEILING	CEIL	CEILING
CHAR_LENGTH ***)	X ***LENGTH mit*** **Byte-Semantik**	LENGTH	LEN ohne nachfolgende Leerzeichen
COALESCE	X VALUE	X NVL	X
COLLATE			
CONVERT USING		CONVERT	CONVERT
CURRENT_USER	USER	USER	X
EXP	X	X	X
FLOOR	X	X	X
LN	X LOG	X	LOG
LOWER	X LCASE	X	X
MOD	X	X	%
NULLIF	X		
OCTET_LENGTH ***)	X *LENGTH*	LENGTHB LENGTHC LENGTH2 LENGTH4	DATALENGTH

SQL Standard	DB2	ORACLE	SQL Server
OVERLAY			
POSITION ***)	X LOCATE POSSTR	INSTR	CHARINDEX PATINDEX
POWER	X	X	X
	RPAD, LPAD	RPAD, LPAD	REPLICATE
SESSION_USER			
SQRT	X	X	X
SUBSTRING ***) FROM FOR	SUBSTRING **SUBSTRING** **(Titel, 1, 5, OCTETS)** ***mit B*yte-Semantik** ***SUBSTR***	SUBSTR ***SUBSTRB*** ***mit B*yte-Semantik** ***SUBSTRC*** ***SUBSTR2*** ***SUBSTR4***	SUBSTRING
SYSTEM_USER			
TRANSLATE USING für charactersets		x CHAR_CS NCHAR_CS	
	TRANSLATE für Strings to from	TRANSLATE für Strings from to	
TRIM LEADING TRAILING BOTH	RTRIM LTRIM STRIP	X RTRIM LTRIM	RTRIM LTRIM
	TRUNC	TRUNC	ROUND(…,…,11)
UPPER	X UCASE	X	X
USER	X	X	X
|| Verkettung Konkatenation	X CONCAT(a,b) a CONCAT b	X CONCAT(a,b)	+ CONCAT(a,b) 2012

3.11 Zeichenfolgenfunktionen im Detail

3.11.1 CHARINDEX bzw. PATINDEX

SQL Server

```
SELECT CHARINDEX( 'c', 'abcde')
3
SELECT PATINDEX( '%AmE%', 'der Name der Rose')
6
```

Oracle

```
SELECT INSTR('abcde','c') FROM DUAL;
3
```

DB2

```
SELECT POSSTR('abcde','c') FROM SYSIBM.SYSDUMMY1
3
```

3.11.2 FORMAT

SQL Server 2012

FORMAT Returns a value formatted with the specified format and optional culture in SQL Server 2012. Use the FORMAT function for locale-aware formatting of date/time and number values as strings. For general data type conversions, use CAST or CONVERT.

```
SELECT FORMAT ( 1234 , 'd10')
0000001234
```

3.11.3 LEFT RIGHT

SQL Server

LEFT und RIGHT sind Abkürzungen der SUBSTRING Funktion

```
SELECT LEFT('ABCDE', 3)
ABC

SELECT RIGHT('ABCDE', 3)
CDE
```

3.11.4 LENGTH, LEN bzw. DATALENGTH

SQL Server

```
SELECT LEN ('abcde  ')
5
SELECT DATALENGTH ('abcde  ')
7
```

LEN „Gibt die Anzahl der Zeichen im gegebenen Zeichenfolgenausdruck zurück (nicht die Anzahl an Bytes), wobei nachfolgende Leerzeichen ausgeschlossen werden.“

DATALENGTH „Gibt die Anzahl von Bytes zurück, die zum Darstellen eines Ausdrucks verwendet werden.“

Oracle LENGTH

```
SELECT LENGTH ('abcde') FROM DUAL;
5
```

DB2 LENGTH byte orientiert

```
SELECT LENGTH ('abcde') FROM SYSIBM.SYSDUMMY1;
5
```

3.11.5 LOWER und UPPER

LOWER(*string*) gibt *string* in Kleinbuchstaben aus.

```
SELECT LOWER('deR bUTT')
der butt

SELECT LOWER('deR bUTT') FROM DUAL;
der butt
SELECT LOWER('deR bUTT') FROM SYSIBM.SYSDUMMY1;
der butt
```

UPPER(*string*) gibt *string* in Großbuchstaben aus.

```
SELECT UPPER('deR bUTT')
DER BUTT

SELECT UPPER('deR bUTT') FROM DUAL;
DER BUTT
SELECT UPPER('deR bUTT') FROM SYSIBM.SYSDUMMY1;
DER BUTT
```

3.11.6 REPLACE, TRANSLATE

SQL Server

```
SELECT REPLACE ('Böll', 'l', 'm')
Bömm
SELECT REPLACE('abcdefghicde','cde','xxx')
abxxxfghixxx
```

Oracle

```
SELECT REPLACE('aiMaiMai', 'ai', 'ei') FROM DUAL;
eiMeiMei
SELECT REPLACE('aiMaiMai', 'ai') FROM DUAL;
MM


FROM l TO m
FROM m TO l
SELECT TRANSLATE ('Böll', 'l', 'm') FROM DUAL;
Bömm
SELECT TRANSLATE ('Böll', 'm', 'l') FROM DUAL;
Böll

SELECT TRANSLATE ('Böll') FROM DUAL
       *
ERROR at line 1:
ORA-00909: invalid number of arguments
----------------------------------------------------------
```

Beachten Sie bitte den Unterschied:

Oracle: FROM TO bzw. DB2: TO FROM

```
----------------------------------------------------------
```

DB2

```
TO m FROM l
TO l FROM m
SELECT TRANSLATE ('Böll', 'l', 'm')
       FROM sysibm.sysdummy1;
Böll
SELECT TRANSLATE ('Böll', 'm', 'l')
       from sysibm.sysdummy1;
Bömm
SELECT TRANSLATE ('Böll') from sysibm.sysdummy1;
BÖLL
```

3.11.7 REPLICATE, RPAD, LPAD

SQL Server

```
SELECT REPLICATE( 'abcd' ,3)
Abcdabcdabcd
```

Oracle

```
SELECT LPAD( 'abcd' ,8,  'x')  FROM DUAL;
SELECT RPAD( 'abcd' ,8,  'x')  FROM DUAL;

SELECT SUBSTR(LPAD( ' ' ,8,  'x'), 2, 8) FROM DUAL;
SELECT SUBSTR(RPAD( ' ' ,8,  'x'), 2, 8) FROM DUAL;
```

DB2 Testen Sie.

3.11.8 RTRIM und LTRIM

SQL Server

```
SELECT LTRIM('abc  ')  , LEN(LTRIM('abc  ')  )
abc   3

SELECT RTRIM('  efgh') ,LEN ( RTRIM('efgh' ))
  efgh 4
```

DB2 unterstützt syntaktische Varianten von RTRIM, LTRIM bezüglich Blanks und STRIP für ein anderes vorgegebenes Zeichen.

Oracle

```
SELECT LTRIM('VVabcdeVVV' , 'V') FROM DUAL;
abcdeVVV

SELECT RTRIM('VVabcdeVVV' , 'V') FROM DUAL;
VVabcde


SELECT LTRIM('  abcde   ' , ' ')
       ,LENGTH ( LTRIM('  abcde   ' , ' '))
       FROM DUAL;
abcde    8
SELECT RTRIM('  abcde   ' , ' ')
       ,LENGTH ( RTRIM('  abcde   ' , ' '))
       FROM DUAL;
  abcde  7
```

Oracle

TRIM LEADING TRAILING BOTH O

TRIM gibt eine Zeichenkette zurück, die bis auf alle führenden und/oder angehängten Füllzeichen (pad characters) mit der gegebenen Zeichenkette identisch ist.

```
SELECT TRIM(LEADING 'V' FROM 'VVabcdeVVV')
       FROM DUAL;
abcdeVVV

SELECT TRIM(TRAILING 'V' FROM 'VVabcdeVVV')
       FROM DUAL;
VVabcde

SELECT TRIM(BOTH 'V' FROM 'VVabcdeVVV') FROM DUAL;
abcde

SELECT TRIM('V' FROM 'VVabcdeVVV') FROM DUAL;
abcde

SELECT TRIM(' ' FROM '  abcde   ') FROM DUAL;
abcde

SELECT TRIM(' ' FROM '  abcde   ')
       ,LENGTH ( TRIM(' ' FROM '  abcde   ') )
       FROM DUAL;
abcde        5

SELECT TRIM('  abcde   ')
       ,LENGTH ( TRIM('  abcde   ') )
       FROM DUAL;
abcde        5
```

3.11.9 STUFF

SQL Server

STUFF ersetzt einen Teilstring durch einen anderen.

```
SELECT STUFF ('0987654321' , 2, 4, 'ab')
0ab54321
```

3.11.10 SUBSTRING

SUBSTRING nimmt eine Teilkette aus einer gegebenen Kette heraus.

SQL Server

```
SELECT SUBSTRING('der Name der Rose', 5, 8)
Name der
```

Oracle

```
SELECT SUBSTR('der Name der Rose', 5, 8)
FROM DUAL;
Name der
```

DB2 byteorientiert

```
SELECT SUBSTR('der Name der Rose', 5, 8)
FROM SYSIBM.SYSDUMMY1
Name der
```

3.11.11 UPPER, LOWER, SUBSTRING und Verkettung

SQL Server

```
SELECT
UPPER(SUBSTRING('deR bUTT', 1, 1 ))
+
LOWER(SUBSTRING('deR bUTT', 2, LEN ('deR bUTT')-1))
Der butt
```

Oracle

```
SELECT
UPPER ( SUBSTR('deR bUTT', 1, 1                    ) )
||
LOWER ( SUBSTR('deR bUTT', 2, LENGTH ('deR bUTT' )-1 ) )
FROM DUAL;
Der butt
```

DB2

```
SELECT
UPPER ( SUBSTR('deR bUTT', 1, 1                    ) )
CONCAT
LOWER ( SUBSTR('deR bUTT', 2, LENGTH ('deR bUTT' )-1 ) )
FROM SYSIBM.SYSDUMMY1;
Der butt
```

3.11.12 Verkettung + CONCAT SQL Server 2012

SQL Server

```
SELECT 'Heinrich' + ' ' +' Böll'
Heinrich Böll

SELECT CONCAT( 'Heinrich' , ' Böll')
Heinrich Böll
```

Verhalten bei NULL?

```
set concat_null_yields_null off
select 'ABC' + null as s1
s1
----
ABC
set concat_null_yields_null ON
select 'ABC' + null as s1
s1
----
NULL
```

Verkettung || Oracle und DB2

Der Verkettungsoperator || kann zur Verkettung zweier Zeichen- oder Bitketten verwendet werden.

```
SELECT  'Heinrich' || ' ' || 'Böll'
FROM DUAL;
Heinrich Böll

SELECT  'Heinrich' || ' ' || 'Böll'
FROM sysibm.sysdummy1;
Heinrich Böll
```

Verkettung CONCAT Oracle und DB2

```
SELECT CONCAT
(
'Heinrich', CONCAT (' ', 'Böll')
)
FROM DUAL;
Heinrich Böll

SELECT CONCAT
(
'Heinrich', CONCAT (' ', 'Böll')
)
FROM sysibm.sysdummy1;
Heinrich Böll
```

DB2 analog wie Oracle und außerdem:

```
SELECT 'Heinrich' CONCAT ' ' CONCAT 'Böll'
FROM sysibm.sysdummy1;
Heinrich Böll
```

aber nicht Oracle:

```
SELECT 'Heinrich' CONCAT ' ' CONCAT 'Böll'
FROM dual;
SELECT 'Heinrich' CONCAT ' ' CONCAT 'Böll'
ERROR at line 1:
ORA-00923: FROM keyword not found where expected
```

Verhalten bei NULL?

Oracle entspricht nicht dem Standard, liefert 'ABC'

```
SELECT 'ABC' || NULL FROM DUAL;
'ABC'
```

DB2 entspricht dem Standard, liefert NULL sofern S1 die NULL präsentiert

```
select buchnr, titel || S1 from tbuch;
NULL
```

3.12 Mathematische Funktionen im Detail

3.12.1 ABS

ABS gibt den absoluten Wert eines Ausdrucks zurück.

```
SELECT ABS(-15)
15


SELECT ABS(-15) FROM DUAL;
15


SELECT ABS(-15) FROM SYSIBM.SYSDUMMY1;
15
```

3.12.2 CEIL

CEIL bzw. CEILING gibt die kleinste ganze Zahl zurück die größer oder gleich ist.

```
SELECT CEILING(15.7)
16
SELECT CEILING(-15.7)
-15

SELECT CEIL(15.7) FROM DUAL;
 16
SELECT CEIL(-15.7) FROM DUAL;
-15

SELECT CEIL(15.7) FROM SYSIBM.SYSDUMMY1;
+1,60000000000000E+001
SELECT CEIL(-15.7) FROM SYSIBM.SYSDUMMY1;
-1,50000000000000E+001
```

3.12.3 EXP

EXP Exponentialfunktion

```
select exp(1)
2.7182818284590451

select exp(1) from dual;
2,71828183

select exp(1) from sysibm.sysdummy1;
+2,71828182845905E+000
```

3.12.4 FLOOR

FLOOR gibt die größte ganze Zahl zurück die kleiner oder gleich ist.

```
SELECT FLOOR(15.7)
15
SELECT FLOOR(-15.7)
-16

SELECT FLOOR(15.7)  FROM DUAL;
15
SELECT FLOOR(-15.7) FROM DUAL;
-16

SELECT FLOOR(15.7) FROM SYSIBM.SYSDUMMY1
+1,50000000000000E+001
SELECT FLOOR(-15.7) FROM SYSIBM.SYSDUMMY1;
-1,60000000000000E+001
```

Achtung: FLOOR und Oracle, was ist 11/-4?

SQL Server

```
select 11/-4 ,FLOOR(11/-4)
-2          -2
```

Oracle

```
select  11/-4  ,FLOOR(11/-4)  from dual;
-2,75          -3
```

DB2

```
select 11/-4 ,FLOOR(11/-4) from sysibm.sysdummy1;
-2          -2
```

3.12.5 LN

LN natürlicher Logarithmus

```
select LOG(1)
0.0
select LOG(EXP (1))
1.0
select EXP (LOG(2.71828183))
2.71828183

select LN(1)                 from dual;
0
select LN(EXP(1))            from dual;
1
select EXP(LN(2.71828183))   from dual;
2,71828183

select LN(1)                 sysibm.sysdummy1;
0
select LN(EXP(1))            sysibm.sysdummy1;
1
select EXP(LN(2.71828183))   sysibm.sysdummy1;
2,71828183
```

3.12.6 % Modulo

% gibt den Rest zurück, wenn zwei Integer-Werte dividiert werden.

```
SELECT 7%5
2
SELECT 11%-4
3

SELECT MOD(7,5) FROM DUAL;
2
SELECT MOD(7,5)  FROM SYSIBM.SYSDUMMY1 ;
2
```

3.12.7 MOD und FLOOR, Oracle und die Mathematik!

Der Test mit SQL Server, Oracle bzw. DB2 liefert bei der Benutzung von FLOOR nicht dasselbe Ergebnis!

Test mit SQL Server:

```
SELECT
 11 AS M
,-4 AS N
,11%-4                     AS "MOD(M/N)"
,11 - (- 4)*FLOOR(11/-4) AS  "modulo klassisch?"

M           N           MOD(M/N)    modulo klassisch?
----------- ----------- ----------- -----------------
         11          -4           3                 3
```

Test mit Oracle:

```
SELECT
 11 AS M
,-4 AS N
,MOD(11, -4)                      AS "MOD(M/N)"
,11 - (- 4)*FLOOR(11/-4) AS  "modulo klassisch?"
from dual;

M          N   MOD(M/N) modulo klassisch?
---------- ---------- ---------- -----------------
        11         -4          3                -1

1 row selected.
```

Test mit DB2:

```
SELECT
 11 AS M
,-4 AS N
,MOD(11, -4)                      AS "MOD(M/N)"
,11 - (- 4)*FLOOR(11/-4) AS  "modulo klassisch?"
from sysibm.sysdummy1;

M           N           MOD(M/N)    modulo klassisch?
----------- ----------- ----------- -----------------
         11          -4           3                 3

  1 Satz/Sätze ausgewählt.
```

Trauen Sie nur Ihrem eigenen gesunden Menschenverstand!

Die folgenden Zeilen stammen aus der Oracle Documentation:

```
SELECT MOD(11,4) "Modulus" FROM DUAL;
   Modulus
----------
         3
```

This function behaves differently from the classical mathematical modulus function when m is negative. The classical modulus can be expressed using the MOD function with this formula:

m – n * FLOOR(m/n)

The following statement illustrates the difference between the MOD function and the classical modulus:

```
SELECT m, n, MOD(m, n),
       m - n * FLOOR(m/n) "Classical Modulus"
FROM test_mod_table;

         M          N   MOD(M,N) Classical Modulus
---------- ---------- ---------- ------------------
        11          4          3                  3
        11         -4          3                 -1
       -11          4         -3                  1
       -11         -4         -3                 -3
```

3.12.8 POWER

Gibt den Wert der Potenz zurück.

```
SELECT POWER(3, 2)
9
SELECT POWER(16, 0.5)
4

SELECT POWER(3, 2) FROM DUAL;
9
SELECT POWER(16, 0.5) FROM DUAL;
4

SELECT POWER(3, 2) FROM SYSIBM.SYSDUMMY1;
9
SELECT POWER(16, 0.5) FROM SYSIBM.SYSDUMMY1;
+4,00000000000000E+000


SELECT POWER(-16, 0.5)
Msg 3623, Level 16, State 1, Line 1
A domain error occurred.

SELECT POWER(-16, 0.5) FROM DUAL
             *
ERROR at line 1:
ORA-01428: argument '-16' is out of range

SELECT POWER(-16, 0.5) FROM SYSIBM.SYSDUMMY1;
SQLSTATE=38552
```

3.12.9 ROUND

SQL Server ROUND(*n*,*m*) Rundet *n* auf die Anzahl Stellen, die mit *m* angegeben wird. m kann auch 0 bzw. negativ sein.

```
SELECT ROUND(158.193 ,  2)
158.190
SELECT ROUND(158.193 ,  1)
158.200
SELECT ROUND(158.193 ,  0)
158.000
SELECT ROUND(158.193  )
Server: Nachr.-Nr. 189, Schweregrad 15, Status 1, Zeile 1
Die round-Funktion erfordert 2 bis 3 Argumente.
SELECT ROUND(158.193 , -1)
160.000
SELECT ROUND(158.193 , -2)
200.000
```

Oracle ROUND(*n*[,*m*]) Rundet *n* auf die Anzahl Stellen, die mit *m* angegeben wird. Wenn keine Zahl m angegeben wird, dann ist *m* = 0. *m* kann auch negativ sein.

```
SELECT ROUND(158.193 ,  2) FROM DUAL;
158,19
SELECT ROUND(158.193 ,  1) FROM DUAL;
158,2
SELECT ROUND(158.193 ,  0) FROM DUAL;
158
SELECT ROUND(158.193  ) FROM DUAL;
158
SELECT ROUND(158.193 , -1) FROM DUAL;
160
SELECT ROUND(158.193 , -2) FROM DUAL;
200
```

DB2 Testen Sie.

3.12.10 SIGN

SIGN(*n*) Wenn *n*<0 Rückgabewert = -1, wenn *n*=0 Rückgabewert = 0, wenn *n*>0 Rückgabewert 1.

```
SELECT SIGN(-15)
-1

SELECT SIGN(-15)  FROM DUAL;
-1

SELECT SIGN(-15)  FROM SYSIBM.SYSDUMMY1;
-1
```

3.12.11 SQRT

SQRT Quadratwurzel

```
SELECT SQRT(27)
5.196152422706632

SELECT SQRT(27) FROM DUAL;
5,19615242

SELECT SQRT(27) FROM SYSIBM.SYSDUMMY1
+5,19615242270663E+000
```

3.12.12 TRUNC und SQL Server ROUND(… , … , 111111)

SQL Server

```
TRUNC wird simuliert mit Hilfe von ROUND

SELECT ROUND(123456.789, 2 , 00000000);
123456.790

SELECT ROUND(123456.789, 2 , 1111111);
123456.780
```

ROUND (numeric_expression , length [,function])

When *function* is omitted or has a value of 0 (default), *numeric_expression* is rounded. When a value other than 0 is specified, *numeric_expression* is truncated.

Oracle TRUNC(*n*,[*m*]) Abschneiden der Dezimalstellen von *n* hinter *m*. Ist *m* nicht gesetzt, gilt der Wert 0. *m* kann auch negativ sein.

```
SELECT TRUNC(158.193 , 2) FROM DUAL;
158,19
SELECT TRUNC(158.193 , 1) FROM DUAL;
158,1
SELECT TRUNC(158.193 , 0) FROM DUAL;
158
SELECT TRUNC(158.193 ) FROM DUAL;
158
SELECT TRUNC(158.193 , -1) FROM DUAL;
150
SELECT TRUNC(158.193 , -2) FROM DUAL;
100
```

DB2 Testen Sie.

3.13 Dezimalarithmetik genauer betrachtet

3.13.1 AVG das arithmetische Mittel

```
select
preis, buchnr
from tbuch
order by preis, buchnr
;
```

```
preis                                buchnr
------------------------------------ -----------
NULL                                 1
NULL                                 2
NULL                                 11
NULL                                 12
0.50                                 8
3.50                                 5
20.50                                6
55.00                                9
99.99                                7
99.99                                18
99.99                                27
```

```
SELECT
AVG(preis) as avgpreis
FROM tbuch
where buchnr <>8
;
----SQL Server
avgpreis
---------------------------------------
63.161666

----DB2
avgpreis
---------------------------------------
63.16166666666666666666666666

----Oracle
avgpreis
---------------------------------------
63.1616667
```

3.13.2 Was ist AVG(sal) *1000?

```
SELECT
AVG(preis)*1000 as avgspreis1000
FROM tbuch
;

----SQL Server




----DB2
Error: DB2 SQL Error: SQLCODE=-802, SQLSTATE=22003, SQLERRMC=null,
DRIVER=4.18.60
SQLState:  22003
ErrorCode: -802




----Oracle
```

3.13.3 Runden, Abschneiden, Konvertieren einer Dezimalzahl, „entferne Zeroes“

```
----SQL Server CAST rundet
   12345.678
   12345.680
   12345.68
   12345.670
   12345.67

   WITH ZWI AS
   (select 12345.678 as wert
   )
   select
       wert                                    AS wert
    --,CAST(wert AS DECIMAL(07,2))             AS wertROUNDcastimplizit
   ------
    ,ROUND(wert,02,0)                          AS wertround
    ,CAST(ROUND(wert,02,0) AS DECIMAL(7,2))    AS wertroundcast
   ------
    ,ROUND(wert,02,1)                          AS werttrunc
    ,CAST(ROUND(wert,02,1) AS DECIMAL(7,2))    AS werttrunccast
   FROM zwi
   ;
  -----------------------------------------------------
----DB2 CAST schneidet ab
   12345.678
   12345.680
   12345.68
   12345.670
   12345.67

   WITH ZWI AS
   (select 12345.678 as wert FROM SYSIBM.SYSDUMMY1
   )
   select
      wert                                    AS wert
    --,CAST(wert AS DECIMAL(07,2))             AS wertTRUNCcastimplizit
   ----
    ,ROUND(wert,02)                            AS wertround
    ,CAST(ROUND(wert,02) AS DECIMAL(7,2))      AS wertroundcast
   ------
    ,TRUNC(wert,02)                            AS werttrunc
    ,CAST(TRUNC(wert,02) AS DECIMAL(7,2))      AS werttrunccast

   FROM zwi
   ;
  -----------------------------------------------------
```

```
----Oracle CAST rundet
   12345.678
   12345.68
   12345.68
   12345.67
   12345.67

   WITH ZWI AS
   (select 12345.678 as wert FROM DUAL
   )
   select
      wert                                  AS wert
    --,CAST(wert AS DECIMAL(07,2))          AS wertROUNDcastimplizit
   ----
    ,ROUND(wert,02)                         AS wertround
    ,CAST(ROUND(wert,02) AS DECIMAL(7,2))   AS wertroundcast
   ------
    ,TRUNC(wert,02)                         AS werttrunc
    ,CAST(TRUNC(wert,02) AS DECIMAL(7,2))   AS werttrunccast
   FROM zwi
   ;
```

3.13.4 Arithmetik Division und AVG

```
DROP TABLE tab123
;
CREATE TABLE tab123
(
  schl integer not null
 ,wert integer not null
 ,PRIMARY KEY (schl)
 )
;
insert into tab123(schl, wert) values (1,1);
insert into tab123(schl, wert) values (2,2);
insert into tab123(schl, wert) values (3,5);
select schl, wert from tab123;
```

Was erwarten Sie bei folgenden Abfragen, SQL Server, DB2, Oracle?

ERROR oder

2 oder 2.6 oder 2.666666 oder 2.66666667 oder 2.666666666666 oder 2.6666666666666666 oder 2.66666666666666666666666666664

--SQL Server

```
select
 SUM(wert)/count(wert)       as avg1
,AVG(wert)                   as avg2
,SUM(wert*1.0)/count(wert)   as avg3
,SUM(wert)*1.0/count(wert)   as avg4
,SUM(wert)*(1.0/count(wert)) as avg5
,AVG(wert*1.0)               as avg6
,(select (8/3)           )       as avg7
,(select (8/3.0)         )       as avg8
,(select (1+2+5)/3       )       as avg9
,(select (1+2+5)/3.0     )       as avg10
,(select (1+2+5.0)/3.0   )       as avg11
from tab123
;
  ----------------------------
```

--DB2

```
select
 SUM(wert)/count(wert)       as avg1
,AVG(wert)                   as avg2
,SUM(wert*1.0)/count(wert)   as avg3  --DB2 2,6
,SUM(wert)*1.0/count(wert)   as avg4  --DB2 2.6666666666666666
,SUM(wert)*(1.0/count(wert)) as avg5  --DB2 2.66666666666666666666666666664
,AVG(wert*1.0)               as avg6
,(select (8/3)         from sysibm.sysdummy1  )       as avg7
,(select (8/3.0)       from sysibm.sysdummy1  )       as avg8
,(select (1+2+5)/3     from sysibm.sysdummy1 )        as avg9
,(select (1+2+5)/3.0   from sysibm.sysdummy1 )        as avg10
,(select (1+2+5.0)/3.0 from sysibm.sysdummy1 )        as avg11
from tab123
;
  ----------------------------
```

--Oracle

```
select
 SUM(wert)/count(wert)       as avg1
,AVG(wert)                   as avg2
,SUM(wert*1.0)/count(wert)   as avg3
,SUM(wert)*1.0/count(wert)   as avg4
,SUM(wert)*(1.0/count(wert)) as avg5
,AVG(wert*1.0)               as avg6
--,(select (8/3)         from DUAL )        as avg7  --ERROR
--,(select (8/3.0)       from DUAL  )       as avg8  --ERROR
--,(select (1+2+5)/3     from DUAL )        as avg9  --ERROR
--,(select (1+2+5)/3.0   from DUAL )        as avg10 --ERROR
--,(select (1+2+5.0)/3.0 from DUAL )        as avg11 --ERROR
from tab123
;
  ----------------------------
```

3.13.5 Arithmetik und ROUND bzw. TRUNC

```
DROP TABLE tab123
;
CREATE TABLE tab123
(
  schl integer not null
 ,wert integer not null
 ,PRIMARY KEY (schl)
 )
;
insert into tab123(schl, wert) values (1,1);
insert into tab123(schl, wert) values (2,2);
insert into tab123(schl, wert) values (3,5);
select schl, wert from tab123;
```

gewüschte Liste

```
roundavg        truncavg
--------------- --------------- ------------------

2.667           2.666
```

--SQL Server

```
select
 CAST(ROUND(AVG(wert*1.0),3,0) AS DECIMAL(31,3))  AS roundavg
,CAST(ROUND(AVG(wert*1.0),3,1) AS DECIMAL(31,3))  AS truncavg
from tab123
;
  ----------------------------
```

--DB2

```
select
 CAST(ROUND(AVG(wert*1.0),3) AS DECIMAL(31,3))  AS roundavg
,CAST(TRUNC(AVG(wert*1.0),3) AS DECIMAL(31,3))  AS truncavg
from tab123
;
  ----------------------------
```

--Oracle

```
select
 CAST(ROUND(AVG(wert*1.0),3) AS DECIMAL(31,3))  AS roundavg
,CAST(TRUNC(AVG(wert*1.0),3) AS DECIMAL(31,3))  AS truncavg
from tab123
;
  ----------------------------
```

Empfehlung:

Nach AVG bzw. Division immer CAST (runden oder CAST (abschneiden

3.13.6 Was erwarten Sie bei diesen Abfragen?

```
SELECT
 buchnr
 ,preis
 ,preis *1.19                 as brutto1
 ,preis *((100+19)/100)       as "brutto2???"
 ,preis *((100+19)*1.0/100)   as brutto3
 ,preis *((100+19*1.0)/100)   as brutto4
 ,(preis/100)*119             as brutto5
 ,(preis*119)/100             as brutto6
 from tbuch
WHERE preis IS NOT NULL
;
SELECT
 SUM(preis)                                    as sumpreis
,AVG(preis)                                    as avgpreis1
,SUM(preis)/(select count(preis) from tbuch) as avgpreis2
from tbuch
;
SELECT
AVG(preis)*100
from tbuch
;
```

Was erwarten Sie bei "brutto2???"

SQL Server

DB2

Oracle

3.13.7 Precision und Scale, Addition und Subtraktion

Was ist +9 +3.987?

	SQL Server	DB2	Oracle
(37,0)+(38,3) 9+3.987	>>>+13.0		>>>+12.987
(30,0)+(31,3) 9+3.987	>>>+12.987	>>>+12.987	>>>+12.987

DB2 for DB2 for Linux, UNIX, and Windows
(p,s) + (p',s')

Addition and subtraction

The precision is min(31,max(p-s,p'-s') +max(s,s')+1). The scale of the result of

addition and subtraction is max (s,s').

SQL Server
(p1,s1) – (p2,s2)

Operation	Result precision	Result scale *
e1 + e2	max(s1, s2) + max(p1-s1, p2-s2) + 1	max(s1, s2)
e1 - e2	max(s1, s2) + max(p1-s1, p2-s2) + 1	max(s1, s2)

* The result precision and scale have an absolute maximum of 38. When a result precision is greater than 38, it is reduced to 38, and the corresponding scale is reduced to try to prevent the integral part of a result from being truncated. In some cases such as multiplication or division, scale factor will not be reduced in order to keep decimal precision, although the overflow error can be raised.

In addition and subtraction operations we need `max(p1 – s1, p2 – s2)` places to store integral part of the decimal number. If there is not enough space to store them i.e. `max(p1 – s1, p2 – s2) < min(38, precision) – scale`, the scale is reduced to provide enough space for integral part. Resulting scale is `MIN(precision, 38) - max(p1 – s1, p2 – s2)`, so the fractional part might be rounded to fit into the resulting scale.

Oracle

vgl. numeric Datatypes und SQL*PLUS Formatelemente

„All decimal data is converted to floating point format, both for storing and for calculations. In floating point format, a number is represented by means of a mantissa and an exponent. The mantissa and the exponent are stored as binary numbers. „

----SQL Server

```
select
 cast(9     as decimal(37,0))
+cast(3.987 as decimal(38,3))   as sp1
--from DUAL
;
select
 cast(9     as decimal(30,0))
+cast(3.987 as decimal(31,3))   as sp2
--from DUAL
;
sp1
---------------------------------------
13.0

(1 row(s) affected)

sp2
---------------------------------------
12.987

(1 row(s) affected)
```

----DB2

```
select
 cast(9     as decimal(30,0))
+cast(3.987 as decimal(31,3))   as sp1
from sysibm.sysdummy1
;
```

----Oracle

```
select
 cast(9     as decimal(37,0))
+cast(3.987 as decimal(38,3))   as sp1
from DUAL
;
select
 cast(9     as decimal(30,0))
+cast(3.987 as decimal(31,3))   as sp2
from DUAL
;
Oracle SQL*Plus Präsentation
COLUMN sp1 FORMAT 999.9;
select
 cast(9     as decimal(37,0))
+cast(3.987 as decimal(38,3))   as sp1
from DUAL
;
liefert 13.0
```

3.13.8 Precision und Scale, Multiplikation und Division

```
select
 cast(0.0000009000 as decimal(30,20))
*cast(1.0000000000 as decimal(30,20)) AS sel1
,cast(0.0000009000 as decimal(30,10))
*cast(1.0000000000 as decimal(30,10)) AS sel2
----FROM SYSIBM.SYSDUMMY1
----FROM DUAL
;
```

	SQL Server	DB2	Oracle
(30,20)*(30,20)	**(38,17)** 0.00000090000000000	**(31,31)** 0.000000900000000000000000000000000	0.0000009
(30,10)*(30,10)	**(38,6)** 0.000001	**(31,20)** 0.00000090000000000000	0.0000009

Beispiel DB2 for DB2 for Linux, UNIX, and Windows

(p,s) * (p',s')
Multiplication

The precision of the result of multiplication is min (31,p+p') and the scale is min(31,s+s').

SQL Server

(p1,s1) – (p2,s2)

Operation	Result precision	Result scale *
e1 * e2	p1 + p2 + 1	s1 + s2
e1 / e2	p1 - s1 + s2 + max(6, s1 + p2 + 1)	max(6, s1 + p2 + 1)

* The result precision and scale have an absolute maximum of 38. When a result precision is greater than 38, it is reduced to 38, and the corresponding scale is reduced to try to prevent the integral part of a result from being truncated. In some cases such as multiplication or division, scale factor will not be reduced in order to keep decimal precision, although the overflow error can be raised.

In multiplication and division operations we need `precision - scale` places to store the integral part of the result. The scale might be reduced using the following rules:

- The resulting scale is reduced to `min(scale, 38 - (precision-scale))` if the integral part is less than 32, because it cannot be greater than `38 - (precision-scale)`. Result might be rounded in this case.
- The scale will not be changed if it is less than 6 and if the integral part is greater than 32. In this case, overflow error might be raised if it cannot fit into decimal(38, scale)
- The scale will be set to 6 if it is greater than 6 and if the integral part is greater than 32. In this case, both integral part and scale would be reduced and resulting type is decimal(38,6). Result might be rounded to 6 decimal places or overflow error will be thrown if integral part cannot fit into 32 digits.

```
select
 cast(8 as decimal(31,00)) / cast(3 as decimal(31,02))  as scalea
,cast(8 as decimal(31,01)) / cast(3 as decimal(31,02))  as scaleb
,cast(8 as decimal(31,02)) / cast(3 as decimal(31,02))  as scalec
,cast(8 as decimal(31,03)) / cast(3 as decimal(31,02))  as scaled
,cast(8 as decimal(31,04)) / cast(3 as decimal(31,02))  as scalee
--from sysibm.sysdummy1
--from DUAL
;
```

	SQL Server	DB2	Oracle Präsentation abhängig vom Default-Format des Client
(31,00)/(31,02)	2.666666	ERROR scale negativ	2.66666667
(31,01)/(31,02)	2.666666	ERROR scale negativ	2.66666667
(31,02)/(31,02)	2.6666666	2	2.66666667
(31,03)/(31,02)	2.66666666	2,6	2.66666667
(31,04)/(31,02)	2.666666666	2,66	2.66666667

Oracle SQL*Plus Präsentation

```
COLUMN spalte1 FORMAT 99999999999999999999999999999.9999999;

select 8/3 as spalte1 from dual;

select cast(8 as number(38,7))/3 as spalte1 from dual;
```

```
COLUMN spalte1 FORMAT 9.9;

select 8/3 as spalte1 from dual;
```

4

Datenbankdesign

4 Datenbankdesign

4.1 Problem: „Finde das passende logische Design und die Integritätsbedingungen für die vorliegenden Daten“.

Datenbankdesign ist eher eine Kunst als eine Wissenschaft.

Es gibt einige wissenschaftliche Prinzipien. Aber es gibt viele Designfragen, die von diesen Prinzipien nicht behandelt werden. Es gibt deshalb wenig objektive Kriterien für manche zu fällende Entscheidung beim Design.

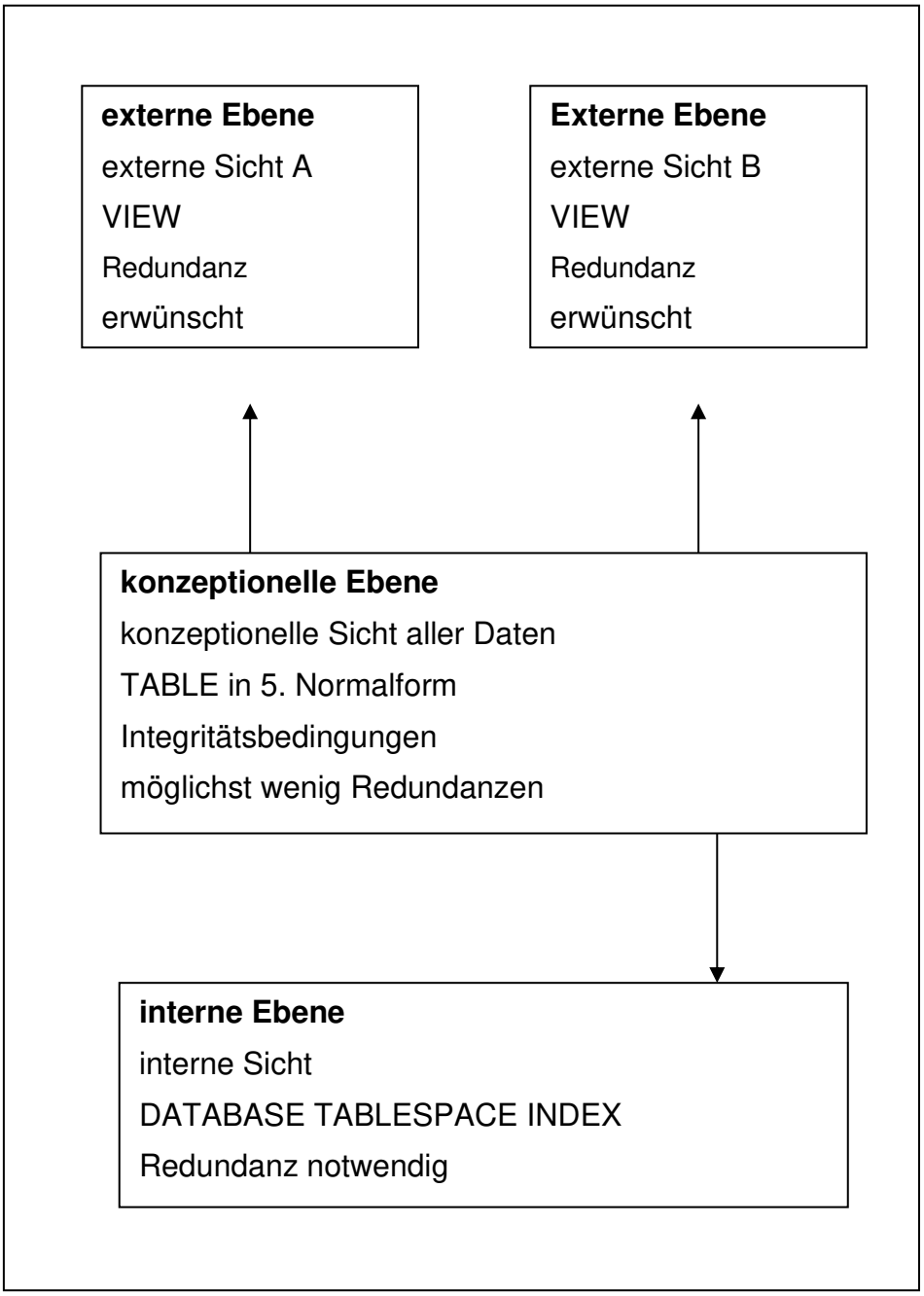
- Das Design benutzt eine **Top-Down Methode** um zu „großen“ Relationen/Tables zu kommen (die semantische Datenmodellierung bietet Zugang zum Problem). **Die bekannteste Vorgehensweise für die semantische Datenmodellierung ist die Entity/Relationship Methode und ihre Diagramme** (diese beruhen auf dem Entity/Relationship Modell von Chen).
- Das Konzept der Normalformen wird genutzt, um „große“ Relationen/Tables in kleinere zu zerlegen und gewisse Integritätsbedingungen auf einfache Art und Weise durchzusetzen. **Auf der Ebene der Relationen/Tables ist der Normalisierungsprozess für eine saubere Datenanalyse die wichtigste Vorgehensweise.**

Der Normalisierungsprozess ist nichts anderes als formalisierter gesunder Menschenverstand und zerlegt eine Relation/Table **ohne Verlust von Information** mit Hilfe von Projektionen in mehrere Relationen/Tables, so dass jede dieser Tables einen Entitätstyp repräsentiert und deshalb voll normalisiert ist (5. Normalform).

Der Normalisierungsprozess hat aber nichts damit zu tun, wie wir zuerst zu diesen Relationen/Tables kommen. Dazu benötigen wir eine Top-Down Methode. Normalisierung ist ein Prozess des Redesigns. In der Praxis ergänzen sich die beiden Wege.

Eine gute Top-Down Design Methode führt sowieso meist zu einem voll normalisierten Design!

4.2 Konzeptionelle Ebene, Integritätsbedingungen und Redundanzen



Externe Ebene/Logische Ebene für einen Benutzer – Redundanz erwünscht

- Die externe Ebene ist die Ebene des einzelnen Benutzers (Anwendungsprogramm, Endanwender). Sie kümmert sich darum, wie die Daten einem individuellen User präsentiert werden.
- Aus der Sichtweise der Benutzer (externe Ebene) ist Redundanzfreiheit nicht immer zweckmäßig.

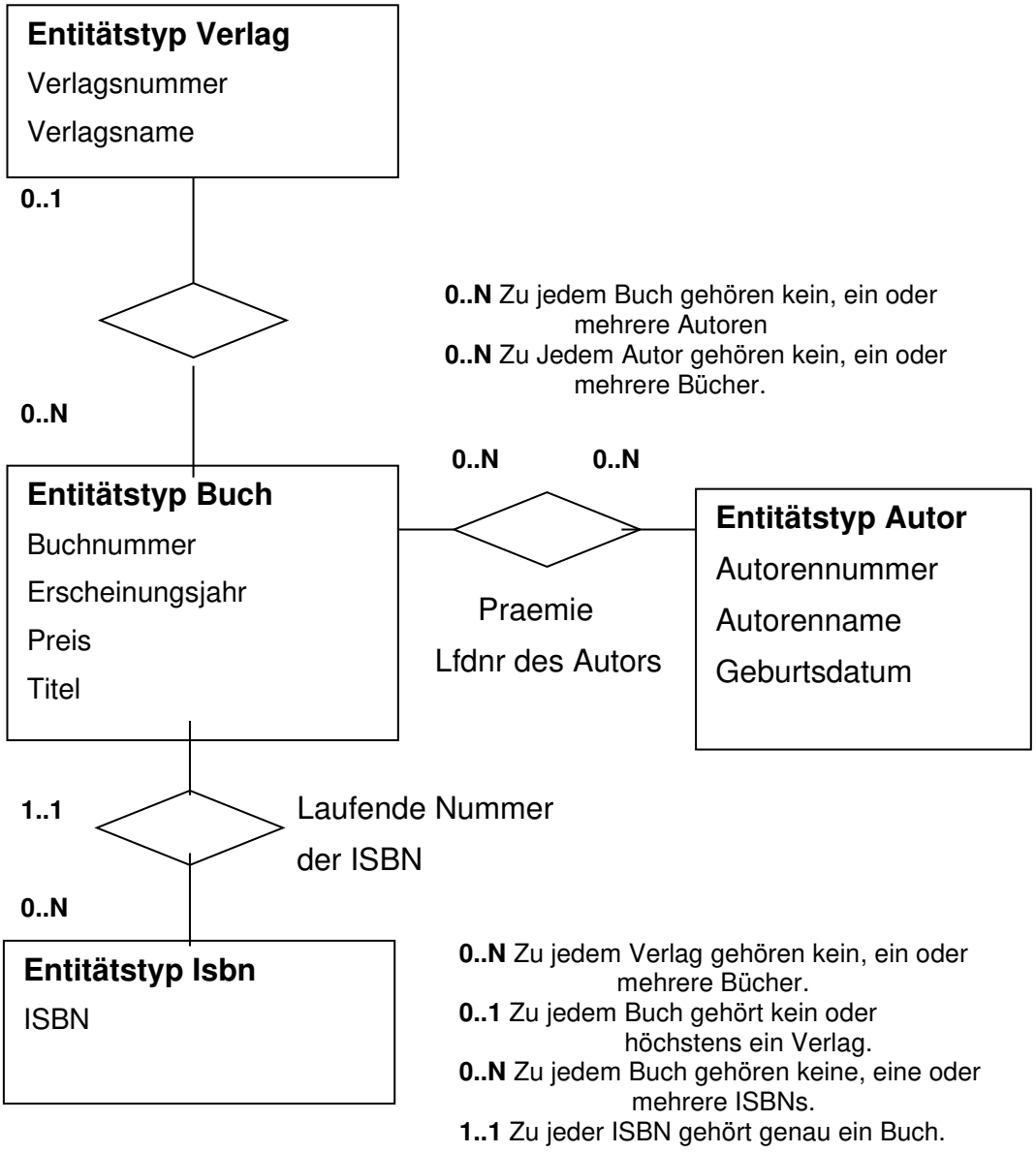
Konzeptionelle Ebene/Logische Ebene für alle Benutzer – möglichst wenig Redundanz

- Auf der konzeptionellen Ebene sollte ein Ausschnitt der Realität ohne Rücksicht auf physische Speicherung und Zugriffstechniken (physische Datenunabhängigkeit) bzw. ohne Rücksicht auf Überlegungen applikatorischer Art präsentiert werden. Dieser zentrale, stabile Bezugspunkt sollte nur dann einer Änderung unterliegen, wenn der bislang berücksichtigte Realitätsausschnitt erweitert werden muss.
- Die konzeptionelle Ebene präsentiert alle Daten der Datenbank. Auf der konzeptionellen Ebene werden auch die Integritätsbedingungen (Integrity Constraints) definiert.
- Im Normalisierungsprozess werden gewisse **Integritätsbedingungen** (sie beschreiben Zusammenhänge der Daten innerhalb einer Table) benutzt, um vor allem **Redundanzen** zu beseitigen und diese Integritätsbedingungen auf einfache Weise durchzusetzen. Im voll normalisierten Bestand muss dazu nur die Eindeutigkeit der Primärschlüssel und Alternativschlüssel durchgesetzt werden!

Interne Ebene/Physische Ebene – Redundanz notwendig

- Die interne Ebene kümmert sich darum, wie die Daten physisch gespeichert werden. Es versteht sich, dass dabei auf das Realitätsmodell der konzeptionellen Ebene Bezug zu nehmen ist.
- Redundanz auf der internen/physischen Ebene verbessert für gewisse Anwendungen die Performance!

4.3 Semantische Datenmodellierung, Entity/Relationship Diagramm



Das Entity/Relationship Diagramm orientiert sich an der realen Welt. Es beschreibt Entitäten, Eigenschaften von Entitäten, Beziehungen „zwischen“ Entitäten, Eigenschaften von Beziehungen, Identifikationsschlüssel.

Wir haben eine Menge von semantischen Konzepten um über die Realität zu reden und symbolische Objekte, um diese semantischen Konzepte darzustellen.

Entität (entity)

Ein Objekt, eine Person, eine Struktur der realen Welt oder der Vorstellungswelt ist eine Entität (individuelles Exemplar).

Entitätstyp (entity type)

Entitäten können eingeteilt werden (klassifiziert werden) in Entitätstypen. Eine Entität ist eine Instanz ihres Entitätstyps.

Entitätsmenge

Eine Entitätsmenge ist eine Menge von Entitäten des gleichen Entitätstyps.

Eigenschaft (property, Merkmal, Attribut)

Die Entitäten eines Entitytyps haben gemeinsame Eigenschaften. Eine Eigenschaft ist ein Stück Information, das eine Entität beschreibt. Für eine konkrete Entität nehmen die Eigenschaften konkrete Werte ihres zugrunde liegenden Datentyps an.

Identität (identity)

Jede Entität hat eine spezielle Eigenschaft, mit der sie eindeutig identifiziert werden kann.

Beziehung (relationship)

Jede Entität kann eine Beziehungen haben zu anderen Entitäten. Beziehungen zwischen Entitäten können selbst als Entitäten aufgefasst werden.

Subtyp (subtype)

Eine Entität ist Instanz von wenigstens einem Entitätstyp. Eine Entität kann aber auch gleichzeitig Instanz von mehreren Entitätstypen sein. Manche Mitarbeiter sind Anwendungsentwickler, manche sind Systemprogrammierer. Wir sagen, dass die Entitätstypen Anwendungsentwickler bzw. Systemprogrammierer beide Subtypen sind des Supertyps Mitarbeiter. Alle Eigenschaften des Supertyps Mitarbeiters sind automatisch auch Eigenschaften der Subtypen Anwendungsentwickler bzw. Systemprogrammierer.

4.4 Präsentation von Daten in Form von Tables

Tverlag

Verlagnr	Verlag
1111	Forkel

Tbuch

Buchnr	Erschj	Preis	Titel	Verlagnr
5	1988	3.50	Ansichten eines Clowns	NULL
27	NULL	99.99	die Jüdin von Toledo	NULL
6	1988	20.50	die Blechtrommel	NULL
7	1989	99.99	der Name der Rose	NULL
8	1977	0.50	der Butt	1111
9	1990	55.00	DB2 fuer Sie	1111
11	1990	NULL	Elvis in Heidelberg	NULL
12	1989	NULL	a guide to db2	NULL
18	1989	99.99	Database Systems	NULL
1	NULL	NULL	C	NULL
2	NULL	NULL	C	NULL

Tisbn

Buchnr	Isbn	Lfdnr
8	3472864303	1
12	0201501139	1
8	34728643yx	2

Tautor

Autornr	Autor	Geburtsdatum
1	Boell	NULL
2	Grass	NULL
3	Eco	NULL
6	Scheifele	NULL
10	Emil Hack	NULL
11	Frieda Holz	NULL
20	C. J. Date	NULL
21	Colin J. White	NULL
100	BUSCH	NULL
200	BUSCH	NULL

Tvautor

Buchnr	Autornr	Lfdnr	Praemie
5	1	0	NULL
6	2	0	NULL
7	3	0	NULL
8	2	0	20.00
9	10	1	10.00
9	11	2	498.00
12	20	1	30.00
12	21	2	NULL
1	100	1	NULL
1	200	2	NULL
2	200	0	NULL

- **Eine Table eignet sich sehr gut zur datenmäßigen Darstellung einer Entitätsmenge.**
- **Eine Table ist die graphische Darstellung einer Relation.**

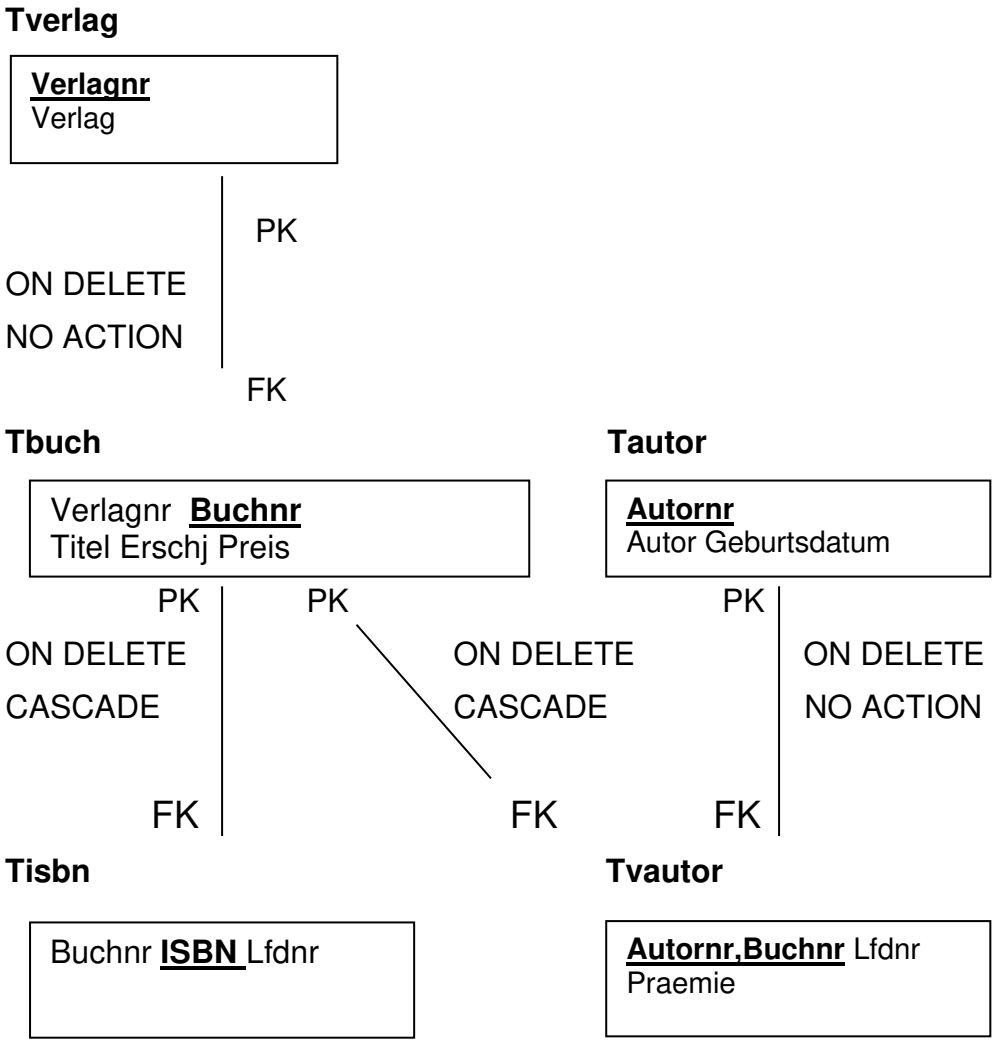
Folgende Integritätsbedingungen wurden festgelegt:

- Ein Buch mit genau einem Autor hat Tvautor.Lfdnr=0.
- Ein Buch mit genau einer Isbn hat Tisbn.Lfdnr=1.

Tvautor.Lfdnr muss vom Anwender gepflegt werden. Für die Festlegung, wer der erste, zweite ... Autor eines Buches ist, gibt es keinen formalen Algorithmus. Mit Hilfe von Tvautor.Lfdnr=0 ist feststellbar, ob ein Buch genau einen Autor hat.

Tisbn.Lfdnr muss vom Anwender gepflegt werden. Für die Festlegung, welches die erste, zweite ... Isbn eines Buches ist, gibt es keinen formalen Algorithmus. Mit Hilfe von Tisbn.Lfdnr=0 bzw. =1 kann nicht festgestellt werden, ob ein Buch genau eine Isbn hat.

4.5 Konzeptionelles Strukturdiagramm



Die Tables/Relationen sind in 5. Normalform.

4.6 Integritätsbedingungen

4.6.1 Primärschlüssel, Alternativschlüssel, Fremdschlüssel

Die **Primärschlüssel** der einzelnen Tables/Relationen sind unterstrichen:

- Verlagnr ist der Primärschlüssel von Tverlag.
- Buchnr ist der Primärschlüssel von Tbuch.
- Autornr ist der Primärschlüssel von Tautor.
- (Autornr,Buchnr) ist der Primärschlüssel von Tvautor.
- ISBN ist der Primärschlüssel von Tisbn.

Alternativschlüssel sind in zwei der Tables/Relationen vorhanden:

- (Buchnr, Lfdnr) ist ein Alternativschlüssel von Tisbn.
- (Buchnr, Lfdnr) ist ein Alternativschlüssel von Tvautor.

Drei Tables/Relationen haben **Fremdschlüssel**:

- Verlagnr ist Fremdschlüssel in Tbuch und referenziert Tverlag.
- Buchnr ist Fremdschlüssel in Tisbn und referenziert Tbuch.
- Buchnr ist Fremdschlüssel in Tvautor und referenziert Tbuch.
- Autornr ist Fremdschlüssel in Tvautor und referenziert Tautor.

Eine 1:N Beziehung wird mit Hilfe von Primärschlüssel bzw. Alternativschlüssel und Fremdschlüssel dargestellt.

Eine N:M Beziehung wird mit Hilfe einer eigenen Table/Relation dargestellt, diese enthält dann zwei Fremdschlüssel.

Im konzeptionellen Strukturdiagramm werden die dynamischen Zusammenhänge zwischen den Daten beschrieben.

- Was soll passieren, wenn wir in Tverlag die Zeile mit der Verlagnr 1111 löschen? DELETE FROM Tverlag WHERE Verlagnr=1111;
- Was soll passieren, wenn wir in Tbuch die Zeile mit der Buchnr 8 löschen? DELETE FROM Tbuch WHERE Buchnr=8;
- Was soll passieren, wenn wir in Tautor die Zeile mit der Autornr 1 löschen? DELETE FROM Tautor WHERE Autornr=1;

4.6.2 Integritätsbedingungen des Relationalen Modells

Das Relationale Modell beinhaltet die folgenden Integritätsbedingungen:

Entity Integrity Constraint: Keine Komponente eines Schlüsselkandidaten (Primärschlüssel bzw. Alternativschlüssel) darf ‚NULL-Werte' annehmen.

Referential Integrity Constraint: Jeder ‚nicht-NULL' Fremdschlüsselwert stimmt mit einem Wert des relevanten Schlüsselkandidaten der referenzierten Table überein.

Diese Integritätsbedingungen werden bei entsprechender Datendefinition vom Datenbankmanagementsystem durchgesetzt.

4.6.3 Fachliche Integritätsbedingungen

Der Titel eines Buches, der Name eines Autors, der Name eines Verlages muss im System vorhanden sein.

Preis eines Buches > 0

Erscheinungsjahr eines Buches > 0

Praemie eines Buches >= 0

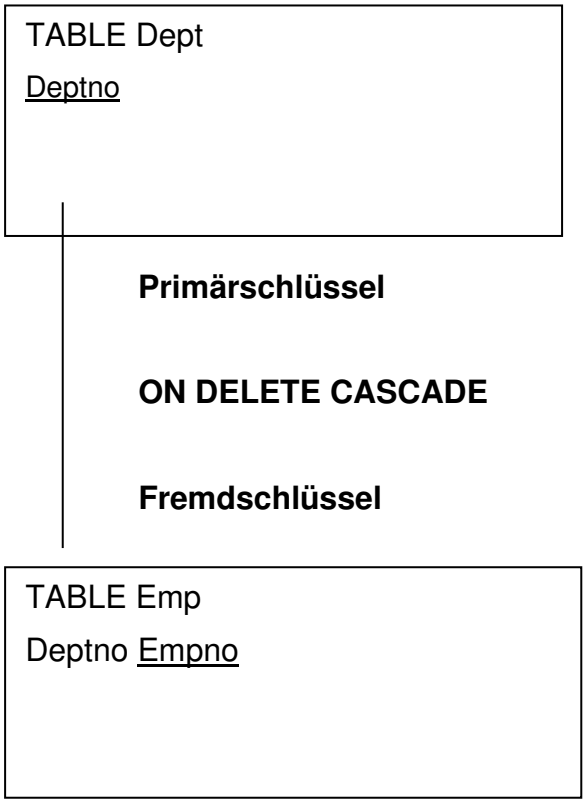
Tisbn.Lfdnr, die einzige Isbn hat Lfdnr = 1, zwischen 1 und 9, pro Buchnr aufsteigend ab 1 ohne Lücken wenn mehrere Isbns vorhanden sind

Tvautor.Lfdnr, der einzige Autor hat Lfdnr = 0, zwischen 0 und 9, pro Buchnr aufsteigend ab 1 ohne Lücken wenn mehr als ein Autor vorhanden ist

Ein Teil dieser Integritätsbedingungen kann bei entsprechender Datendefinition vom Datenbankmanagementsystem durchgesetzt.

4.7 ON DELETE CASCADE

Business Rule: Eine Abteilung wird aufgelöst, alle Mitarbeiter werden entlassen.



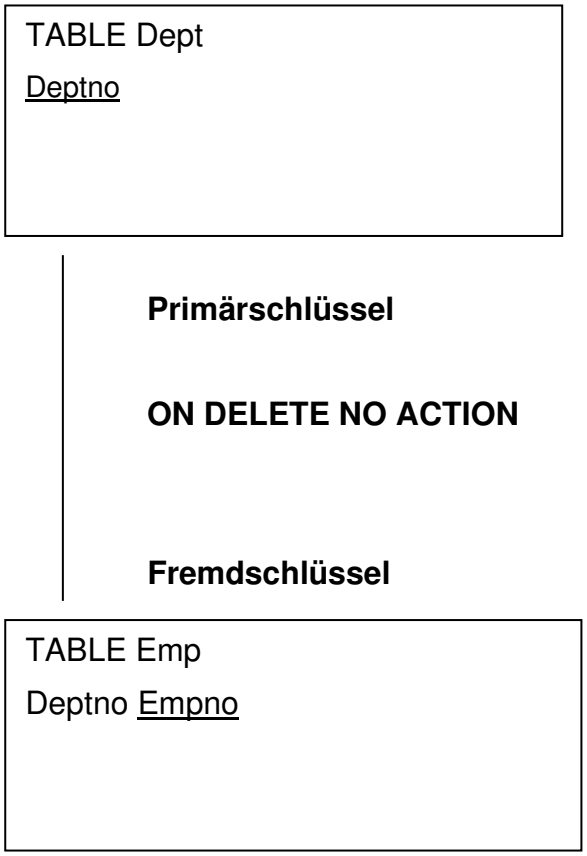
ON DELETE CASCADE definiert eine „Delete-Regel für den Fremdschlüssel“ und CASCADE ist die Referentielle Aktion in dieser Delete-Regel.

DELETE FROM Dept WHERE Deptno = 10;

Die Delete-Anweisung wird erfolgreich ausgeführt, alle Zeilen mit dem Fremdschüsselwert Emp.Deptno=10 (Daten aller Mitarbeiter der Abteilung 10) werden wegen ON DELETE CASCADE vom System mitgelöscht.

4.8 ON DELETE NO ACTION

Business Rule: Eine Abteilung kann nicht aufgelöst werden, sofern ihr noch Mitarbeiter zugeordnet sind.



ON DELETE NO ACTION definiert eine „Delete-Regel für den Fremdschlüssel“ und NO ACTION ist die Referentielle Aktion in dieser Delete-Regel.

DELETE FROM Dept WHERE Deptno = 10;

Der Delete-Anweisung wird wegen ON DELETE NO ACTION nur dann erfolgreich ausgeführt, wenn in der TABLE Emp keine Zeilen mit dem Fremdschlüsselwert Emp.Deptno=10 vorhanden sind.

DEPT

DEPTNO	DNAME	LOCNO
20	RESEARCH	4711
30	SALES	4812
40	OPERATIONS	4812

EMP

DEPTNO	ENAME	EMPNO	JOB	MGR	HIREDATE	SAL	COMM
20	ADAMS	7876	CLERK	7788	12.01.83	1100.00	NULL

DELETE FROM Dept;

Diese Delete-Anweisung wird wegen ON DELETE NO ACTION nur dann erfolgreich ausgeführt, wenn in der TABLE Emp keine Zeile mit einem wohldefinierten Fremdschlüsselwert Emp.Deptno vorhanden ist.

Alle Abteilungen können nur dann mit einem Schlag aufgelöst werden, wenn vorher alle Mitarbeiter entlassen wurden.

4.9 Referentielle Aktionen in der Delete-Regel für den Fremdschlüssel

DELETE von Primärschlüssel- bzw. Alternativschlüsselzeilen, die abhängigen Fremdschlüsselwerte dürfen aber nicht ‚in die Luft zeigen'. **Es bestehen folgende Alternativen bei der Datendefinition:**
ON DELETE NO ACTION ***) Ein Fremdschlüsselwert verhindert die erfolgreiche Ausführung der mengeorientierten DELETE-Anweisung.
ON DELETE CASCADE Die Fremdschlüsselzeilen werden mitgelöscht.
ON DELETE SET NULL Die Fremdschlüsselwerte werden auf NULL gesetzt. ON DELETE SET DEFAULT *) x) Die Fremdschlüsselwerte werden auf DEFAULT gesetzt.

***) bei Oracle impliziter Default

*) von Oracle nicht unterstützt

x) von DB2 nicht unterstützt

4.10 Referentielle Aktionen in der Update-Regel für den Fremdschlüssel

UPDATE von Primärschlüssel- bzw. Alternativschlüsselwerten, die abhängigen Fremdschlüsselwerte dürfen aber nicht ‚in die Luft zeigen'. **Es bestehen folgende Alternativen bei der Datendefinition:**
ON UPDATE NO ACTION ***) xxx) Ein Fremdschlüsselwert verhindert die erfolgreiche Ausführung der mengeorientierten UPDATE-Anweisung.
ON UPDATE CASCADE *) x) Die Fremdschlüsselwerte werden mitgeändert.
ON UPDATE SET NULL *) x) Die Fremdschlüsselwerte werden auf NULL gesetzt. ON UPDATE SET DEFAULT *) x) Die Fremdschlüsselwerte werden auf DEFAULT gesetzt.

***) bei Oracle impliziter Default

*) von Oracle nicht unterstützt

xxx) bei DB2 impliziter Default

x) von DB2 nicht unterstützt

4.11 Erstellen der Tables und der Constraints

```
DROP TABLE Tvautor;
DROP TABLE Tisbn;
DROP TABLE Tbuch;
DROP TABLE Tverlag;
DROP TABLE Tautor;
COMMIT;
-----------------------------------------------
  CREATE TABLE Tverlag
  (
   Verlagnr   INTEGER    NOT NULL
  ,Verlag     CHAR(20)   NOT NULL
  ,PRIMARY KEY (Verlagnr)
  )
  ;
  GRANT SELECT ON Tverlag TO PUBLIC
  ;
-----------------------------------------------
  CREATE TABLE Tbuch
  (
   Buchnr        INTEGER      NOT NULL
  ,Erschj        DECIMAL(4)
  ,Preis         DECIMAL(7,2)
  ,Verlagnr      INTEGER
  ,Titel         VARCHAR(127) NOT NULL
  ,PRIMARY KEY (Buchnr)
  )
  ;
  ALTER TABLE Tbuch ADD CONSTRAINT
                  FK_Tbuch_Tverlag
                  FOREIGN KEY (Verlagnr)
                  REFERENCES Tverlag(Verlagnr)
                  ;
                --ON DELETE NO ACTION

  GRANT SELECT ON Tbuch TO PUBLIC
  ;
-----------------------------------------------
  CREATE TABLE Tisbn
  (
   Buchnr    INTEGER NOT NULL
  ,Isbn      CHAR(10)  NOT NULL
  ,Lfdnr     DECIMAL(1) NOT NULL
  ,PRIMARY KEY (Isbn)
  ,CONSTRAINT Altkey_Tisbn UNIQUE (Buchnr,Lfdnr)
  ,FOREIGN KEY (Buchnr)
              REFERENCES Tbuch(Buchnr)
              ON DELETE CASCADE
  )
  ;
  GRANT SELECT ON Tisbn  TO PUBLIC
  ;
```

```
  ------------------------------------------------
  CREATE TABLE Tautor
  (
   Autornr      INTEGER      NOT NULL
  ,Autor        VARCHAR(240) NOT NULL
  ,Geburtsdatum DATE
  --yyyy-mm-dd           DATE DB2, DATE SQL Server
  --yyyy-mm-dd hh:mi:ss  DATE Oracle, DATETIME SQL Server
  ,PRIMARY KEY (Autornr)
  )
  ;
  GRANT SELECT ON Tautor TO PUBLIC
  ;
  ------------------------------------------------
  CREATE TABLE Tvautor
  (
   Buchnr    INTEGER NOT NULL
  ,Autornr   INTEGER NOT NULL
  ,Lfdnr     DECIMAL(1) NOT NULL
  ,Praemie   DECIMAL(9,3)
  ,PRIMARY KEY (Autornr,Buchnr)
  ,CONSTRAINT Altkey_Tvautor UNIQUE (Buchnr,lfdnr)
  ,FOREIGN KEY (Buchnr)
               REFERENCES Tbuch (Buchnr)
               ON DELETE CASCADE
  ,FOREIGN KEY (Autornr)
               REFERENCES Tautor (Autornr)
             --ON DELETE NO ACTION
  )
  ;
  GRANT SELECT ON Tvautor TO PUBLIC
  ;
-----------------------------------------------
ALTER TABLE Tbuch ADD CONSTRAINT Tbuchconpreis
CHECK ( 0.00 <  Preis)
;
ALTER TABLE Tbuch ADD CONSTRAINT Tbuchconerschj
CHECK ( 0. < Erschj)
;
ALTER TABLE Tisbn ADD CONSTRAINT Tisbnconlfdnr
CHECK ( 1. <= lfdnr AND lfdnr  <= 9. )
;
ALTER TABLE Tvautor ADD CONSTRAINT Tvautorconlfdnr
CHECK ( 0. <= lfdnr AND lfdnr  <= 9. )
;
ALTER TABLE Tvautor ADD CONSTRAINT Tvautorconpraemie
CHECK ( 0.00 <= praemie )
;
--ALTER TABLE Tbuch   DROP CONSTRAINT Tbuchconpreis;
--ALTER TABLE Tbuch   DROP CONSTRAINT Tbuchconerschj;
--ALTER TABLE Tisbn   DROP CONSTRAINT Tisbnconlfdnr;
--ALTER TABLE Tvautor DROP CONSTRAINT Tvautorconlfdnr;
--ALTER TABLE Tvautor DROP CONSTRAINT Tvautorconpraemie;
```

Die „Delete-Regeln für die Fremdschlüssel“ (--ON DELETE NO ACTION) sind jeweils auf Kommentar gesetzt. Bei Oracle zum Beispiel kann die Delete-Regel ON DELETE NO ACTION nicht explizit codiert werden.

ON DELETE NO ACTION ist der Default.

Die „Update-Regeln für die Fremdschlüssel“ sind nicht explizit codiert. Bei einigen Produkten am Markt kann eine Update-Regel nicht explizit codiert werden, diese Produkte unterstützen nur den Default ON UPDATE NO ACTION.

ON UPDATE NO ACTION ist der Default.

Beachten Sie bitte, dass in den Tables die vom SQL-Standard definierten Datentypen INTEGER, DECIMAL, CHAR, VARCHAR und DATE verwendet werden.

Der Datentyp DATE ist bei Oracle nicht dasselbe wie beim SQL Standard.

Oracle kennt kein INTEGER bzw. DECIMAL, Oracle kennt NUMBER.

Oracle’s VARCHAR2 ist nicht dasselbe wie VARCHAR.

4.12 Designüberlegungen zu den Delete- bzw. Update-Regeln für Fremdschlüssel

Die Delete- bzw. Update-Regeln für Fremdschlüssel sind Eigenschaften der Daten, nicht der Anwendungen. Die Delete- bzw. Update-Regeln für Fremdschlüssel müssen nicht in jeder Anwendung programmiert werden, sondern werden einmal definiert. Das System sichert die Konsistenz der Daten ab. Die Produktivität der Anwendungsentwicklung erhöht sich (Vermeidung von Doppelarbeit).

Die Delete- bzw. Update-Regeln für Fremdschlüssel sollte die Performance einer Anwendung nicht negativ beeinflussen. Selbstgeschriebene Routinen zur Kontrolle der Referentiellen Integrität bei zeilenweiser Verarbeitung haben in der Regel schlechtere Performance als die Systemroutinen. Masseninsert, Massenupdate bzw. Massendelete erfordern besondere Überlegungen beim Anwendungsdesign.

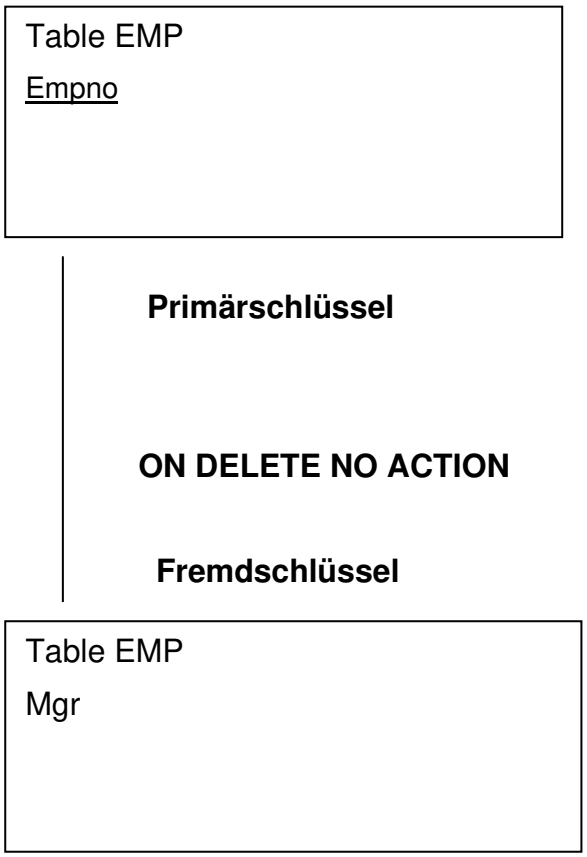
Flexibilität Die Delete- bzw. Update-Regeln können über die ALTER-Anweisung entfernt und neu definiert werden, sofern sich die reale Welt ändert.

Unabhängigkeit vom Zugriffspfad

Das Ergebnis einer DELETE- bzw. UPDATE-Operation ist bei NO ACTION unabhängig vom physischen Zugriffspfad. Um diese Unabhängigkeit zu gewährleisten sind bei einigen Produkten am Markt bei komplexen referentiellen Strukturen (z. B. bei einem referentiellen Cycle oder bei einer selbst-referenzierenden Table) gewisse Einschränkungen bei der Implementierung der Delete- bzw. Update-Regeln vorhanden.

4.13 Selbst-referenzierende Table

Business Rule: Ein Manager kann nicht gekündigt werden, sofern ihm noch ein Mitarbeiter zugeordnet ist.



Emp ist eine selbst-referenzierende Table

Emp.Empno ist Primärschlüssel

Emp.Mgr ist Fremdschlüssel und referenziert Emp.Empno

4.14 Unabhängigkeit vom Zugriffspfad, ON DELETE NO ACTION bzw. ON DELETE RESTRICT

Table Emp

Empno	Mgr
100	NULL
200	100
300	200
400	300

DELETE FROM Emp WHERE Empno > 200;

Angenommen ON DELETE RESTRICT wäre bei einer selbst-referenzierenden Table unter DB2 implementierbar:

- Bei Delete-Regel ON DELETE RESTRICT wird die Löschung aller Mitarbeiter mit Personalnummer > 200 erfolgreich ausgeführt, wenn die Zeilen von „unten nach oben“ gelöscht werden.
- Die Löschung ist nicht erfolgreich, wenn die Zeilen von „oben nach unten“ bearbeitet werden.
- Die Ausführung wäre also abhängig davon, welchen Zugriffspfad der Optimizer wählt.

Zwei Zitate zum Thema NO ACTION bzw. RESTRICT:

"The only difference between NO ACTION and RESTRICT is when the referential constraint is enforced. RESTRICT (IBM SQL rule) enforces the rule immediately, and NO ACTION (SQL standard rule) enforces the rule at the end of the statement. This difference matters only in the case of a searched DELETE involving a self-referencing constraint that deletes more than one row. NO ACTION might allow the DELETE to be successful where RESTRICT (if it where allowed) would prevent it." DB2 UDB for OS/390 Version 6 SQL Reference, Seite 604.

"The default value for the rule depends on the value of the CURRENT RULES special register when the CREATE TABLE statement is processed. If the value of the register is 'DB2', the delete rule defaults to RESTRICT, if the value is 'STD', the delete rule defaults to NO ACTION." DB2 UDB for OS/390 Version 6 SQL Reference Seite 547.

Bei ON DELETE NO ACTION (SQL:1992) wird der Check bezüglich der Referentiellen Integrität am Ende der Anweisung durchgeführt.

Bei ON DELETE RESTRICT (SQL:1999) wird der Check bezüglich der Referentiellen Integrität sofort beim Löschen einer Zeile durchgeführt.

4.15 Redundanz und funktionale Abhängigkeit

TverlagXYZ

Verlagnr	Verlag	Stadtnr	Stadt
10	rororo	4711	Köln
20	dtv	4711	Köln
30	fischer	4812	Düsseldorf
40	reclam	4812	Düsseldorf

Ein Blick auf die Table TverlagXYZ gibt uns sofort das Gefühl, dass etwas mit dem Design nicht in Ordnung ist: Redundanz!

Für die Namen der Städte, in denen sich unsere Verlage befinden, benötigen wir eine eigene Table.

Tstadt

Stadtnr	Stadt
4711	Köln
4812	Düsseldorf
0717	Neustadt
0919	Neustadt

Tverlag

Verlagnr	Verlag	Stadtnr
10	rororo	4711
20	dtv	4711
30	fischer	4812
40	reclam	4812

Wenn wir die Table Tverlag jetzt betrachten haben wir das Gefühl, dass das Design richtig ist.

Die Spalte Tverlag.Stadtnr ist Fremdschlüssel und referenziert den Primärschlüssel von Tstadt.

Beachten Sie bitte: TverlagXYZ ist das Ergebnis des natürlichen Joins zwischen Tverlag und Tstadt über Stadtnr.

Redundanz heißt das mehrfache Vorkommen der gleichen Aussage.

Die **Redundanzen** in der Table Tverlagxyz führen zu so genannten **Update-Anomalien**, d.h. Schwierigkeiten mit Operationen wie INSERT, DELETE und UPDATE. Wir können zum Beispiel Daten über eine Stadt nur dann erfassen, wenn in dieser Stadt auch ein Verlag ist. Wenn alle Verlage in einer Stadt aufgelöst werden, dann löschen wir auch die Informationen über die Stadt. Und wenn eine Stadt umbenannt wird müssen wir jede Zeile ändern, in der der alte Name auftaucht.

Für jeden Wert von Stadtnr gibt es in Tverlagxyz genau einen Wert von Stadt. Das gilt für den aktuellen Inhalt von Tverlagxyz, soll aber auch für jeden möglichen Inhalt von Tverlagxyz gelten.

Wir sagen, dass Stadt funktional abhängig ist von Stadtnr.

Diese funktionale Abhängigkeit (Functional Dependency) ist eine Integritätsbedingung und kann vom DBMS auf einfache Weise durchgesetzt werden: für die Namen der Städte, in welchen sich die Verlage befinden, benötigen wir eine eigene Table Tstadt.

4.16 Der Normalisierungsprozess, Integritätsbedingungen und Redundanz

Im Normalisierungsprozess werden gewisse Integritätsbedingungen (sie beschreiben Zusammenhänge der Daten innerhalb einer Table) benutzt, um vor allem **Redundanzen** zu beseitigen und diese **Integritätsbedingungen** auf einfache Weise durchzusetzen. Im voll normalisierten Bestand muss dazu nur die Eindeutigkeit der Primärschlüssel und Alternativschlüssel durchgesetzt werden!

Der Normalisierungsprozess ist eine Technik, mit deren Hilfe eine Table ohne Verlust von Information zerlegt wird in mehrere Tables.

Die grundlegende Idee ist wie folgt:

Gegeben sei eine Table und eine **Menge von Integritätsbedingungen** (FDs Functional Dependencies, MVDs Multi-Valued Dependencies, JD's Join Dependencies). Wir zerlegen die Table mit Hilfe von Projektionen in eine Menge kleinerer Tables. Bei jedem Zerlegungsschritt werden die Integritätsbedingungen benutzt um zu entscheiden, welche Projektion als nächstes gewählt wird.

Die Ziele des **Normalisierungsprozesses** sind

- das Erstellen eines Designs, das eine „gute" Präsentation der realen Welt ist – ein Design, das intuitiv leicht zu verstehen ist und eine gute Basis ist für zukünftige Erweiterungen
- das Beseitigen gewisser Redundanzen („eine Tatsache an einer Stelle") und damit das Vermeiden gewisser Update-Anomalien
- das Erleichtern der Durchsetzung der meisten dieser Integritätsbedingungen (wie z. B. funktionale Abhängigkeit zwischen Spalten)

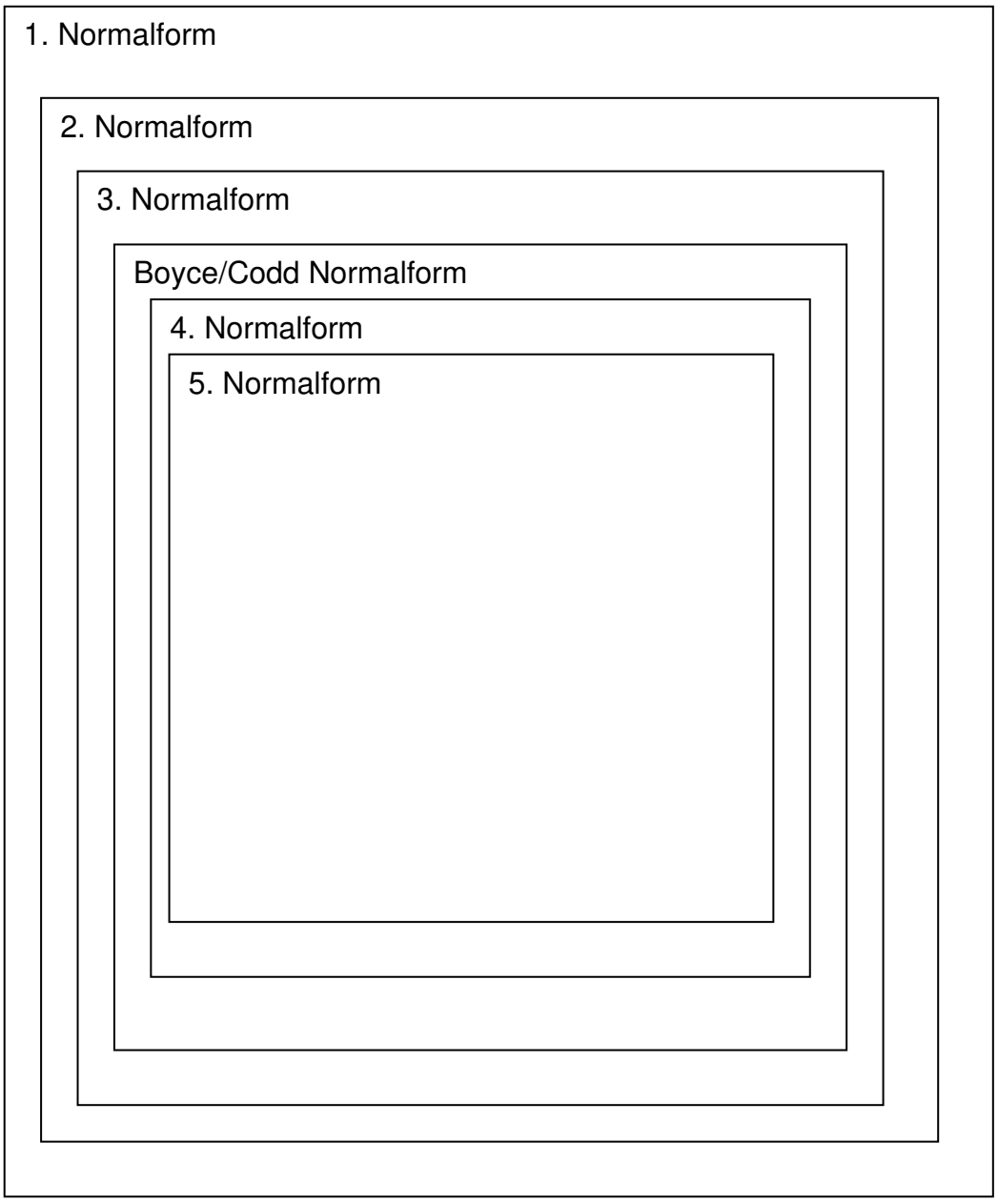
Beachten Sie bitte:

Der Normalisierungsprozess ist ein Hilfsmittel beim Datenbankdesign, er ist aber kein Allheilmittel.

- Die Zerlegung kann auf verschiedene Weise erfolgen, und es gibt wenig objektive Kriterien dafür, welche der alternativen Möglichkeiten ausgewählt werden soll.
- ***Nicht alle Redundanzen können beseitigt werden.***
- Nicht alle Integritätsbedingungen (FDs Functional Dependencies, MVDs Multi-Valued Dependencies, JD's Join Dependencies) können auf einfache Weise durchgesetzt werden.

4.17 Normalformen

Jede Table hat wenigstens einen Schlüsselkandidaten und damit einen Primärschlüssel!



1. Normalform

Eine Table befindet sich qua Definition in 1. Normalform und präsentiert in jeder Zeile für jede Spalte genau einen Wert des zugrunde liegenden Datentyps.

Eine Spalte oder Spaltenkombination B ist **funktional abhängig** von einer Spalte oder Spaltenkombination A derselben Table T, wenn für jeden aktuellen Inhalt der Table T gilt: zu einem bestimmten Wert von A ist höchstens ein Wert von B möglich ist.

2. Normalform (diese Definition setzt voraus, dass die Table nur einen Kandidatenschlüssel hat und dieser sei der Primärschlüssel)

Eine Table befindet sich in 2. Normalform, wenn sie in 1. Normalform ist und jede nicht zum Primärschlüssel gehörende Spalte nur vom ganzen Primärschlüssel funktional abhängig ist. Eine Table, deren Primärschlüssel aus genau einer Spalte besteht, ist immer in 2. Normalform.

3. Normalform (diese Definition setzt voraus, dass die Table nur einen Kandidatenschlüssel hat und dieser sei der Primärschlüssel)

Eine Table befindet sich in 3. Normalform, wenn sie in 2. Normalform ist und zwischen nicht zum Primärschlüssel gehörenden Spalten keine funktionale Abhängigkeiten bestehen.

Bei der Definition der 1., 2. und 3. Normalform war nur vom Primärschlüssel die Rede. Eine Table kann aber auch Alternativschlüssel haben. Schlüssel können sich sogar überlappen. Bei der Boyce/Codd Normalform werden Redundanzen und funktionale Abhängigkeiten der Spalten von den Schlüsselkandidaten (Primärschlüssel und Alternativschlüssel) untersucht.

Boyce/Codd Normalform

Eine Table befindet sich in Boyce/Codd Normalform, wenn jede Determinante der Table ein Schlüsselkandidat ist. Determinante heißt eine Spalte oder eine Spaltenkombination, von welcher irgendeine andere Spalte der Table voll abhängig ist. Eine Spalte ist voll abhängig von einer Spalte oder einer Spaltenkombination A, wenn sie von der Spaltenkombination A funktional abhängig ist, aber nicht bereits von Teilen der Spaltenkombination A.

4. Normalform und 5. Normalform

Bei diesen Normalformen geht es um Redundanzen, die auch in Tables in Boyce/Codd Normalform noch vorhanden sein können. Es werden mehrwertige Abhängigkeiten bzw. Verbundabhängigkeiten untersucht.

4.18 1. Normalform

Tvautorxyz

Buchnr	Autornr	Lfdnr	Praemie	Autor	Geburtsda-tum
5	1	0	NULL	Boell	NULL
6	**2**	0	NULL	**Grass**	NULL
7	3	0	NULL	Eco	NULL
8	**2**	0	20.00	**Grass**	NULL
9	10	1	10.00	Emil Hack	NULL
9	11	2	498.00	Frieda Holz	NULL
12	20	1	30.00	C. J. Date	NULL
12	21	2	NULL	Colin J. White	NULL
1	100	1	NULL	BUSCH	NULL
1	**200**	2	NULL	**BUSCH**	NULL
2	**200**	0	NULL	**BUSCH**	NULL

Die Table Tvautorxyz ist in 1. Normalform aber nicht in 2. Normalform. Autor und Geburtsdatum sind funktional abhängig von Autornr.

4.19 2. Normalform

Tbuchxyz

Buchnr	Erschj	Preis	Titel	Verlagnr	Verlag
8	1977	0.50	der Butt	**1111**	**Forkel**
9	1990	55.00	DB2 fuer Sie	**1111**	**Forkel**

Die Table Tbuchxyz ist in 2. Normalform aber nicht in 3. Normalform.

Verlag ist funktional abhängig von Verlagnr.

4.20 3. Normalform

Tverlag

Verlagnr	Verlag

Tbuch

Buchnr	Erschj	Preis	Titel	Verlagnr

Tautor

Autornr	Autor	Geburtsdatum

Tisbn

Buchnr	Isbn	Lfdnr

Tvautor

Buchnr	Autornr	Lfdnr	Praemie

Alternativschlüssel sind in zwei der Tables/Relationen vorhanden:

(Buchnr, Lfdnr) ist ein Alternativschlüssel von Tisbn.

(Buchnr, Lfdnr) ist ein Alternativschlüssel von Tvautor.

Die Tables/Relationen Tverlag, Tbuch, Tautor, Tisbn, Tvautor sind in **3. Normalform**, aber auch in **Boyce/Codd Normalform**, in **4. Normalform** und auch in **5. Normalform**.

Wir verzichten bewusst darauf, in diesen Seminarunterlagen Boyce/Codd Normalform, MVD Multi-Valued Dependency bzw. JD Join Dependency, 4. Normalform bzw. 5. Normalform exakt zu definieren und ausführlich zu diskutieren. Beispiele zu diesem Thema finden Sie am Ende dieses Kapitels.

Literaturempfehlung:

In dem Buch von C. J. Date – An Introduction to Database Systems – finden Sie neben der Diskussion zu BCNF auch eine ausführliche Diskussion zum Thema MVD Multi-Valued Dependency bzw. JD Join Dependency, 4. Normalform bzw. 5. Normalform.

Wir empfehlen auch das Buch von Fabian Pascal: Practical Issues in Database Management A Reference for the Thinking Practitioner.

4.21 Gutes Datenbankdesign

Die 5. Normalform setzt auf einfache Weise eine Menge von vorgegebenen funktionalen, mehrwertigen und Verbundabhängigkeiten durch, das System muss dazu nur die Eindeutigkeit der Kandidatenschlüssel durchsetzen.

Gutes Datenbankdesign heißt 5. Normalform!

Die 5. Normalform ist frei von Anomalien, die über Projektionen beseitigt werden können!

Eine Table, die sich in 3. Normalform befindet und die nur einfache Schlüssel besitzt (der Primärschlüssel und alle Alternativschlüssel bestehen jeweils nur aus einer Spalte) befindet sich automatisch auch in 5. Normalform.

Die 5. Normalform ist, was Projektionen und Joins betrifft, die „final normal form“.

4.22 „Das Mapping zwischen der konzeptionellen und der internen Ebene des Produkts ist unzureichend!“

In vielen Büchern über SQL und Datenbankmanagementsysteme finden Sie Bemerkungen wie die folgende:

„...möchten wir darauf hinweisen, dass bei den meisten Performance- und Tuning-Betrachtungen, die wir in der Praxis durchführen, die Ursachen für Probleme in unzureichendem Datenbankdesign zu suchen sind...“

Was ist eigentlich die Message dieser Bemerkung?

Ist das Ergebnis des Datenbankdesigns (alle Relationen/Tables in 5. Normalform, saubere Dokumentation aller Integritätsbedingungen) „unzureichend“ oder ist die Unterstützung des Datenbankmanagementsystems für die Implementierung „unzureichend“?

Die Message kann nur lauten: das Mapping zwischen der konzeptionellen und der internen Ebene des Produkts ist unzureichend!

Fast alle Anbieter relationaler Datenbankmanagementsysteme haben logische und physische Aspekte nicht sauber getrennt. Bei den meisten Produkten am Markt werden die Zeilen der konzeptionellen Ebene direkt zur Satzbildung der internen/physischen Ebene herangezogen (die Zeilen einer Base Table werden zu Sätzen einer Datei). Deswegen ist es für manche Anwendungen möglicherweise schwierig, gute Performance zu erreichen.

"....the relational model nowhere stipulates that base relvars must map one-for-one to stored files ... this area is one in which most DBMS vendors have seriously let us down; most SQL products do indeed map base relvars one-for-one to stored files ..." C. J. Date Database Programming & Design January 1998

"The direct mapping between logical and physical, a property of virtually all rdbms, means that physical organisation cannot be independent of logical normalisation. Because of this, database design always represents a compromise between functionality and performance" Database: An Evaluation and Comparison Published 1992 ButlerBloor Ltd.

4.23 Denormalisierung

Aus Performancegründen, weil die Unterstützung des Datenbankmanagementsystems für das Mapping zwischen der konzeptionellen und der internen Ebene sehr schlecht ist, zu restriktiv, „unzureichend“ ist, muss möglicherweise eine beabsichtigte, begründete und dokumentierte Denormalisierung auf konzeptioneller Ebene durchgeführt werden.

- Denormalisierung auf konzeptioneller Ebene kann gut sein für die Performance ganz bestimmter Anwendungen.
- Durch Denormalisierung wird die Anzahl der nötigen Join-Operationen für ganz bestimmte Anwendungen verringert.
- Denormalisierung auf konzeptioneller Ebene erhöht die Redundanz und führt zum Problem der Update-Anomalien.
- Manche Auswertungen sind bei einem denormalisierten Bestand schwieriger zu formulieren!

Denormalisierung auf konzeptioneller Ebene sollte nur im äußersten Notfall als letztes Hilfsmittel in Betracht gezogen werden.

4.24 Boyce/Codd Normalform und Schlüsselkandidaten

Das folgende Beispiel soll das Prinzip verdeutlichen. Es stammt aus dem Buch An Introduction to Database Systems von C. J. DATE („an example that we should warn you some people might consider pathological“).

SJT

S	J	T
Smith	Math	Prof. White
Smith	Physics	Prof. Green
Jones	Math	Prof. White
Jones	Physics	Prof. Brown

Integritätsbedingungen:

FD1 (S,J) -> T

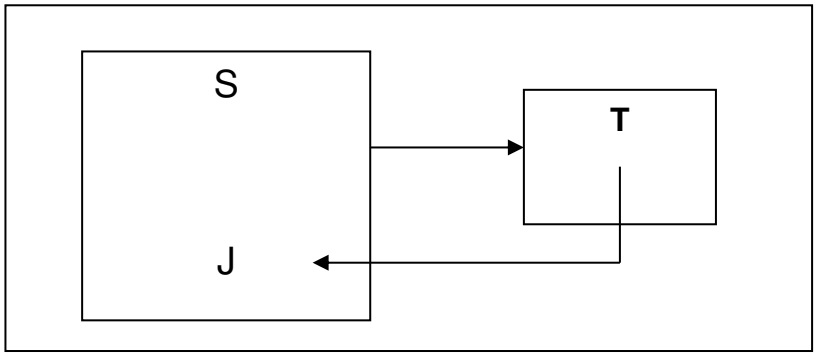
Für jedes Fach hat ein Student dieses Faches genau einen Lehrer.

FD2 (T) -> J

Jeder Lehrer unterrichtet genau ein Fach.

Schlüsselkandidaten sind (S,J) und (S,T), die Schlüsselkandidaten überlappen sich.

Die Table SJT ist in 3. Normalform, aber nicht in Boyce/Codd Normalform. T ist eine Determinante aber kein Schlüsselkandidat.



Update-Anomalie: Wenn wir die Information, dass Jones Physik studiert löschen, dann verlieren wir auch die Information, dass Prof. Brown Physik unterrichtet.

Wir zerlegen die Table SJT mit Hilfe von Projektionen in die Tables ST und TJ.

ST

S	**T**
Smith	Prof. White
Smith	Prof. Green
Jones	Prof. White
Jones	Prof. Brown

TJ

T	**J**
Prof. White	Math
Prof. Green	Physics
Prof. Brown	Physics

ST mit Primärschlüssel (S,T) und TJ mit Primärschlüssel (T) sind in Boyce/Codd Normalform (sogar in 4. Normalform und 5. Normalform).

Problem:

Die zwei Projektionen können nicht unabhängig voneinander verändert werden. Das Einfügen der Zeile

Smith	Prof. Brown

in die Table ST muss zurückgewiesen werden, da Prof. Brown Physik unterrichtet, aber Smith schon von Prof. Green unterrichtet wird.

Die funktionale Abhängigkeit FD1 (S,J) -> T kann aber nicht aus FD2 (T) -> J abgeleitet werden. Das heißt: die funktionale Abhängigkeit FD1 (S,J) -> T kann nach der Zerlegung nicht auf einfache Weise durchgesetzt werden.

Die zwei Ziele, a) Zerlegen einer Table in BCNF-Komponenten (Boyce/Codd Normalform) und b) Zerlegen in unabhängige Komponenten können manchmal nicht gleichzeitig erreicht werden.

Die Boyce/Codd Normalform (sogar die 5. Normalform) ist immer erreichbar; eine Table kann immer zerlegt werden in eine äquivalente Menge von Tables in Boyce/Codd Normalform (oder 5. Normalform).

Aber manchmal ist es vielleicht nicht wünschenswert (wie das Beispiel SJT zeigt) eine Table, die in 3. Normalform ist, noch weiter zu zerlegen. Es gibt keine formalen Kriterien um zu entscheiden, welches Design das „bessere“ ist.

4.25 4. Normalform und mehrwertige Abhängigkeit

Darstellung als Tabelle mit Wiederholungsgruppen

Seminarnr	Buchnr	Referentnr
3605	12 18	Ref01 Ref99
4142	12	Ref01

Darstellung als Relation/Table Trsb

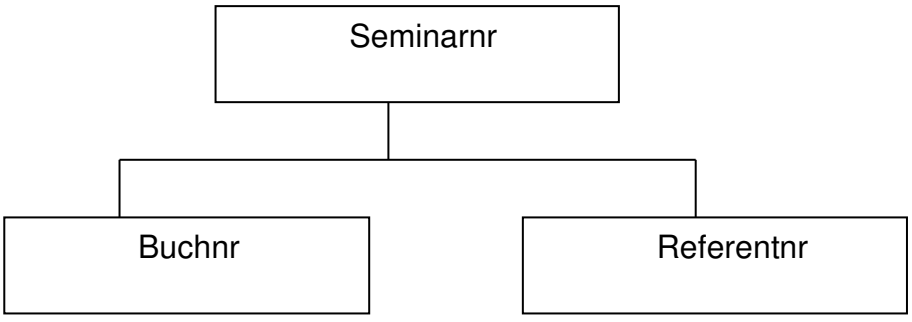
Seminarnr	**Buchnr**	**Referentnr**
3605	12	Ref01
3605	18	Ref01
3605	12	Ref99
3605	18	Ref99
4142	12	Ref01

Beachten Sie bitte, dass der Primärschlüssel von Trsb sich aus allen drei Spalten der Table zusammensetzt.

COBOL-ähnlicher Satz

```
01 seminarsatz.
  02 seminarnr ...
  02 buchnr occurs x times ...
  02 referentnr occurs x times ...
```

Hierarchie



Wir nehmen an, dass in einem Seminar gewisse Bücher benutzt werden und Referentnr und Buchnr unabhängig voneinander sind, d. h. in einem bestimmten Seminar werden von den Referenten des Seminars immer die gleichen Bücher benützt (für ein Seminar eines kommerziellen Schulungsanbieters durchaus realistisch).

Integritätsbedingung

„Wenn die Zeile (r1, s, b1) und (r2, s, b2) in der Table Trsb erscheinen, dann müssen auch die Zeilen (r1, s, b2) und (r2, s, b1) erscheinen."

Diese Integritätsbedingung wird auch mehrwertige Abhängigkeit (MVD Multi-Valued Dependency) genannt.

Referentnr und Buchnr sind mehrwertig abhängig von Seminarnr.

Referentnr ist mehrwertig abhängig von Seminarnr: für eine gegebene Seminarnr s und eine gegebene Buchnr b hängt die Menge der Referentnr r die zu (s,b) gehören nur vom Wert s ab.

Buchnr ist mehrwertig abhängig von Seminarnr: für eine gegebene Seminarnr s und eine gegebene Referentnr r hängt die Menge der Buchnr b die zu (s,r) gehören nur vom Wert s ab.

Die Table Trsb ist in Boyce/Codd Normalform aber nicht in 4. Normalform.

Wir zerlegen die Table Trsb mit Hilfe von Projektionen in die Tables Trs und Tsb.

Trs

Referentnr	**Seminarnr**
Ref01	3605
Ref99	3605
Ref01	4142

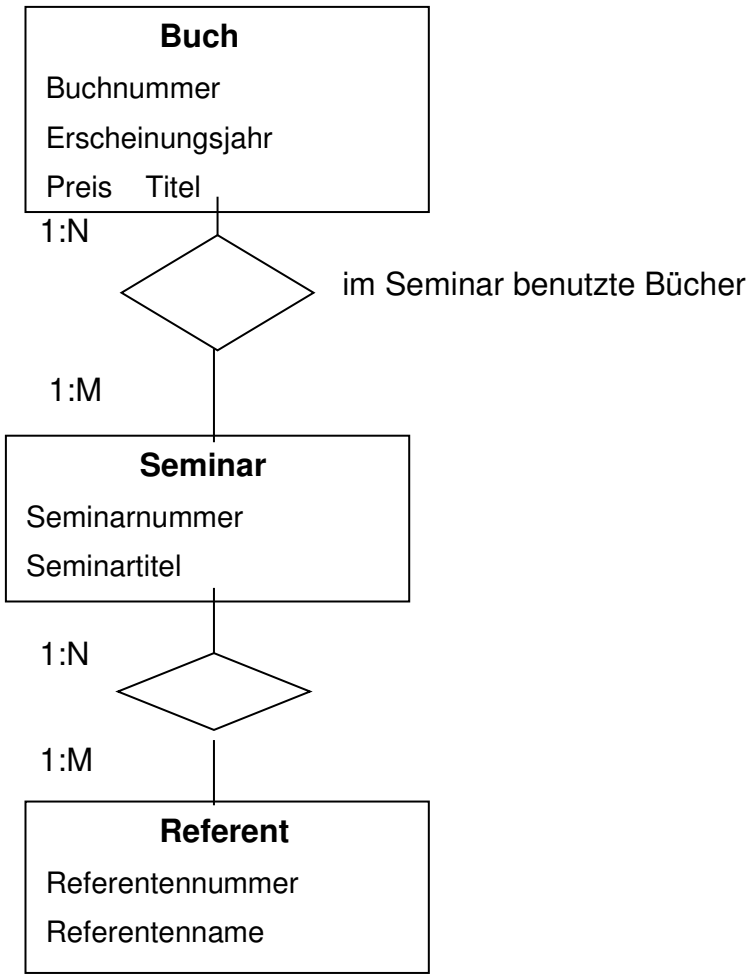
Tsb

Seminarnr	**Buchnr**
3605	12
3605	18
4142	12

Trsb ist gleich dem Join seiner 2 Projektionen Trs und Tsb.

Die Integritätsbedingung, d. h. **die mehrwertige Abhängigkeit** kann nach der Zerlegung in die beiden Projektionen Trs und Tsb vom System in einfacher Weise durchgesetzt werden!

Trs und Tsb sind in 4. Normalform (sogar in 5. Normalform).



Ein sorgfältiges top-down Vorgehen hätte auch zum voll normalisierten Design geführt!

Zitat Anfang
Neben den funktionalen Abhängigkeiten, die unbestritten die wichtigsten semantischen Integritätsbedingungen darstellen, gibt es noch eine Reihe weiterer spezieller semantischer Integritätsbedingungen, die man ebenfalls als Abhängigkeiten bezeichnet. Wir betrachten hier ... – allerdings recht kursorisch – die so genannten „mehrwertigen Abhängigkeiten“ und die „eingebetteten mehrwertigen Abhängigkeiten“.
MEHRWERTIGE ABHÄNGIKEITEN
Solche Abhängigkeiten entstehen in einer Relation aufgrund einer simplen Auflösung von Wiederholungsgruppen...
DIE VIERTEN NOMALFORMEN
Das Vorhandensein „echter mehrwertiger Abhängigkeiten entspricht also dem Vorhandensein von Wiederholungsgruppen, die auf simple Weise in 1NF-Relationen aufgelöst wurden – nämlich durch Vervielfältigung der entsprechenden Zeilen. Man kann sie aber ggf. auch eliminieren, indem man die Relation in geeigneter Weise in andere Relationen zerlegt; wir gelangen so zu den sog. „vierten“ Normalformen, in denen keine „echten“ mehrwertigen Abhängigkeiten mehr existieren...
Zitat Ende

G. Schlageter / W.Stucky Datenbanksysteme: Konzepte und Modelle 1983

4.26 Eingebettete mehrwertige Abhängigkeit

Trsbtage

Referentnr	Seminarnr	Buchnr	Tage
Ref01	3605	12	2
Ref01	3605	18	3
Ref99	3605	12	2,5
Ref99	3605	18	2,5
Ref01	4142	12	3

Wir nehmen an, dass in einem Seminar gewisse Bücher benutzt werden und Referentnr und Buchnr unabhängig voneinander sind, d. h. in einem bestimmten Seminar werden von den Referenten des Seminars immer die gleichen Bücher benützt (für ein Seminar eines kommerziellen Schulungsanbieters durchaus realistisch).

Wir haben eine weitere Spalte Tage. Diese Spalte präsentiert die Anzahl der Tage, die der jeweilige Referent mit dem jeweiligen Buch im Rahmen seiner Seminardurchführung arbeitet.

Beachten Sie bitte: **Trsbtage** enthält eine eingebettete mehrwertige Abhängigkeit und ist in 4. Normalform (sogar in 5. Normalform). Die Redundanzen können nicht mit Hilfe von Projektionen eliminiert werden, da Tage funktional abhängig ist vom gesamten Primärschlüssel (Referentnr, Seminarnr, Buchnr).

Die eingebettete mehrwertige Abhängigkeit kann nicht auf einfache Weise durchgesetzt werden.

4.27 5. Normalform und Verbundabhängigkeiten

Bisher haben wir eine gegebene Table im Normalisierungsprozess immer ohne Verlust von Informationen in 2 Tables zerlegt. Jetzt haben wir es mit der Tatsache zu tun, dass eine bestehende Table nicht in 2 Tables, aber ohne Verlust von Informationen in 3 oder mehr Tables zerlegt werden kann.

Trsbjoin

Referentnr	**Seminarnr**	**Buchnr**
Ref01	3605	12
Ref01	3605	18
Ref99	3605	12
Ref01	4142	12

Beachten Sie bitte, dass der Primärschlüssel von Trsbjoin sich aus allen drei Spalten der Table zusammensetzt.

Beachten Sie bitte, dass die folgenden Zeile sich nicht in Trsbjoin befindet.

Ref99	3605	18

Wir zerlegen die Table Trsbjoin mit Hilfe von 3 Projektionen in die Tables Trsjoin, Tsbjoin und Trbjoin.

Trsjoin (das Seminar wird vom Referenten gehalten)

Referentnr	**Seminarnr**
Ref01	3605
Ref99	3605
Ref01	4142

Tsbjoin (das Buch wird im Seminar benutzt)

Seminarnr	**Buchnr**
3605	12
3605	18
4142	12

Trbjoin (der Referent empfiehlt persönlich das Buch)

Referentnr	**Buchnr**
Ref01	12
Ref01	18
Ref99	12

Die Aussage

„Trsbjoin ist gleich dem Join seiner drei Projektionen Trbjoin, Tbsjoin und Tsrjoin aber nicht gleich dem Join einer Zerlegung in nur 2 Tables“

ist äquivalent zu folgender Integritätsbedingung

„wenn die Zeile (r1, s2, b1), (r2, s1, b1) und (r1, s1, b2) in der Table Trsbjoin vorhanden sind, dann muss auch (r1, s1, b1) in Trsbjoin vorhanden sein“.

und kann auch wie folgt formuliert werden:

```
wenn (r1,s1)     in Trsjoin
und  (s1,b1)     in Tsbjoin
und  (r1,b1)     in Trbjoin
dann auch (r1,s1,b1) in Trsbjoin
```

Trsbjoin repräsentiert einen Ausschnitt der realen Welt in der eine eigenartige, aber im konkreten Beispiel sinnvolle Businessregel gilt:

```
Wenn der Referent R1 das Seminar S1 hält
und im Seminar S1 das Buch B1 benutzt wird
und der Referent R1 das Buch B1 persönlich empfiehlt
dann empfiehlt der Referent R1 im Seminar S1 auch zwingend das Buch B1.
```

Diese Integritätsbedingung wird auch Verbundabhängigkeit (JD Join Dependency) genannt. Die Integritätsbedingung, d. h. **die Verbundabhängigkeit** kann nach der Zerlegung in die drei Projektionen Trsjoin, Tsbjoin und Trbjoin vom System in einfacher Weise durchgesetzt werden!

Die Table Trsbjoin ist in 4. Normalform aber nicht in 5. Normalform.

Trsjoin, Tsbjoin und Trbjoin sind in 5. Normalform.

Beachten Sie bitte:

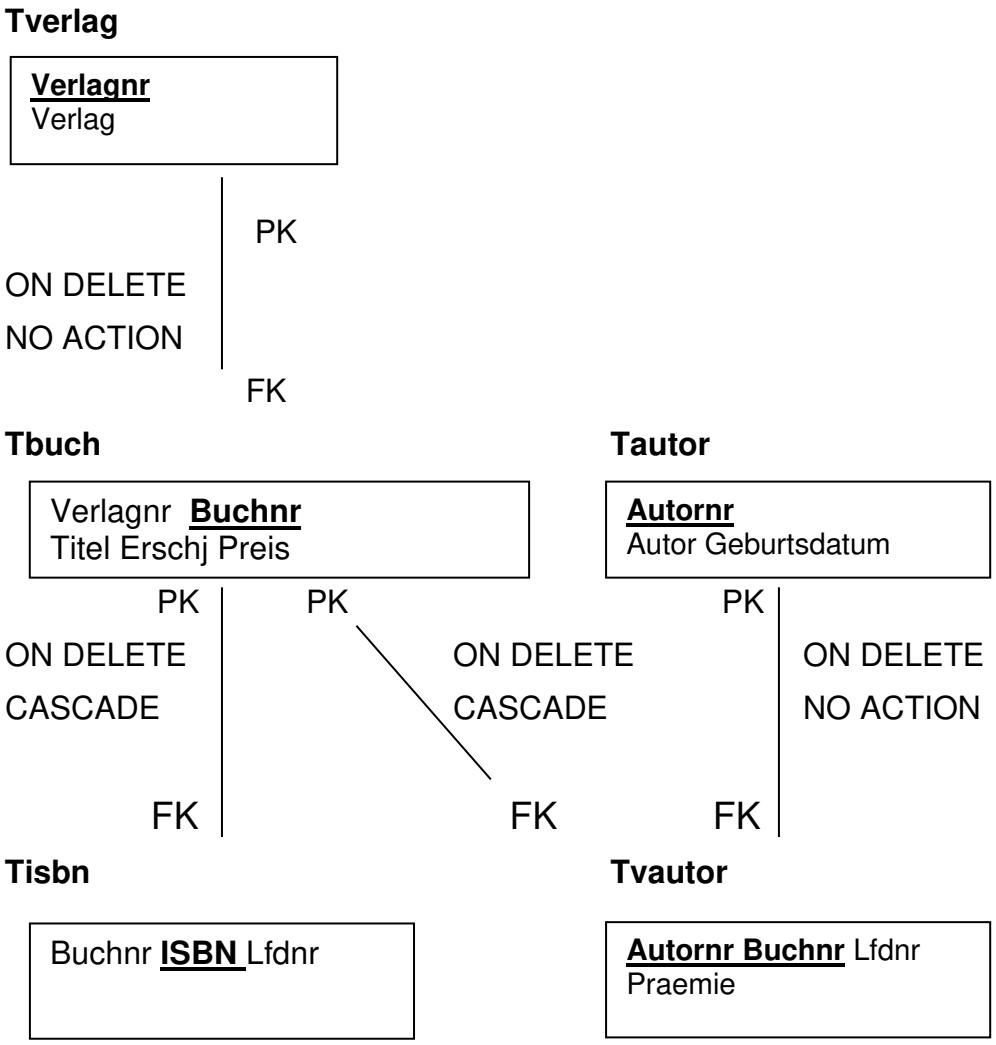
Tables die in 4. Normalform sind, aber nicht in 5. Normalform, sind sehr selten. Verbundabhängigkeiten zu erkennen ist ein nichttriviales Unterfangen weil die intuitive Bedeutung von Verbundabhängigkeiten möglicherweise nicht offensichtlich, augenfällig, einleuchtend, klar ist.

5

SELECT mit mehreren Tables

5 SELECT mit mehreren Tables

5.1 Bücher, Autoren und Verlage



5.2 Testdaten

Tverlag

Verlagnr	5.2.1.1.1.1 Verlag
1111	Forkel

Tbuch

Buchnr	Erschj	Preis	5.2.1.1.1.2 Titel	5.2.1.1.1.3 erlagnr
5	1988	3.50	Ansichten eines Clowns	NULL
27	NULL	99.99	die Jüdin von Toledo	NULL
6	1988	20.50	die Blechtrommel	NULL
7	1989	99.99	der Name der Rose	NULL
8	1977	0.50	der Butt	1111
9	1990	55.00	DB2 fuer Sie	1111
11	1990	NULL	Elvis in Heidelberg	NULL
12	1989	NULL	a guide to db2	NULL
18	1989	99.99	Database Systems	NULL
1	NULL	NULL	C	NULL
2	NULL	NULL	C	NULL

Tisbn

Buchnr	5.2.1.1.1.4 Isbn	Lfdnr
8	3472864303	1
12	0201501139	1
8	34728643yx	2

Tautor

Autornr	Autor	Geburtsdatum
1	Boell	NULL
2	Grass	NULL
3	Eco	NULL
6	Scheifele	NULL
10	Emil Hack	NULL
11	Frieda Holz	NULL
20	C. J. Date	NULL
21	Colin J. White	NULL
100	BUSCH	NULL
200	BUSCH	NULL

Tvautor

Buchnr	Autornr	Lfdnr	Praemie
5	1	0	NULL
6	2	0	NULL
7	3	0	NULL
8	2	0	20.00
9	10	1	10.00
9	11	2	498.00
12	20	1	30.00
12	21	2	NULL
1	100	1	NULL
1	200	2	NULL
2	200	0	NULL

5.3 <query specification> und <subquery>

```
<query specification>::=
        SELECT  [ ALL | DISTINCT ]  ...
        FROM   <table reference>
              ,<table reference>
              , ...
             WHERE  <search condition>
             GROUP BY ...
             HAVING  <search condition>


<search condition>::=
            (<predicate> OR <predicate>)
          AND (<predicate>)
          AND NOT (<predicate>)


<predicate>::=
             ... <comp op> ...
             ... BETWEEN ... AND ...
             ... IN (...)
             ... IN (<query expression>)
             ... LIKE ... ESCAPE ...
             ... IS NULL
             ... <comp op>  ANY (<query expression>)
             ... <comp op>  ALL (<query expression>)
             ... <comp op> SOME (<query expression>)
             EXISTS (<query expression>)
             UNIQUE (<query expression>)
             ... MATCH (<query expression>)


<comp op>::=  =   >   <   <=   >=  <>


<table reference>::=
           <table name>  [ [AS] <correlation name>
                    [(<derived column list>)] ]
         | <subquery>    [ [AS] <correlation name>
                    [(<derived column list>)] ]
         | <joined table>
```

5.4 <query expression>

```
<subquery>::= (<query expression>)


<query expression>::=
             <joined table>
             <non-join query expression>


<non-join query expression>::=
             <query expression> UNION ...
             <query expression> EXCEPT ...
             ... INTERSECT ...
             <query specification>
             <table value constructor>
             TABLE <table name>
             (<non-join query expression>)


<joined table>::=
     <table reference> CROSS JOIN < table reference >
   | <qualified join>
   | <(joined table)>
```

5.5 Kartesisches Produkt

```
SELECT *  FROM Tbuch, Tisbn;
```

Tbuch **Tisbn**

Buchnr	Erschj	Preis	5.5.1.1. itel	Verlagnr	Buchnr	5.5.1.1.1.2 sbn	Lfdnr
1	NULL	NULL	C	NULL	8	3472864303	1
1	NULL	NULL	C	NULL	12	0201501139	1
1	NULL	NULL	C	NULL	8	34728643yx	2
2	NULL	NULL	C	NULL	8	3472864303	1
2	NULL	NULL	C	NULL	12	0201501139	1
2	NULL	NULL	C	NULL	8	34728643yx	2
...							
...							
...							
8	1977	0.50	der B	1111	8	3472864303	1
8	1977	0.50	der B	1111	12	0201501139	1
8	1977	0.50	der B	1111	8	34728643yx	2
...							
...							
...							
27	NULL	99.99	Die J	NULL	8	3472864303	1
27	NULL	99.99	Die J	NULL	12	0201501139	1
27	NULL	99.99	Die J	NULL	8	34728643yx	2

- Die Ergebnistabelle zeigt alle Spalten aus Tbuch und Tisbn.
- Jede Zeile der einen TABLE wird mit allen Zeilen der anderen TABLE angelistet.
- Zeilenzahl Tbuch * Zeilenzahl Tisbn
- Die Daten aus der Spalte Tbuch.Titel können aus graphischen Gründen nur verkürzt angedruckt werden. **Achtung: Titel ist VARCHAR.**
- Semantisch äquivalent ist folgende Formulierung:

```
SELECT * FROM Tbuch CROSS JOIN Tisbn;
```

5.6 Der Inner Natural-Join / der innere natürliche Verbund

```
SELECT Tbuch.Buchnr, Isbn, Preis, Erschj
FROM   Tbuch, Tisbn
WHERE  Tbuch.Buchnr = Tisbn.Buchnr
;
```

```
Buchnr      Isbn       Preis     Erschj
----------- ---------- --------- ------
8           3472864303 .50       1977
8           34728643yx .50       1977
12          0201501139 NULL      1989
```

```
SELECT Tisbn.Buchnr    AS buchnr
     , Tisbn.Isbn      AS isbn
     , Tisbn.Lfdnr     AS lfdnri
     , Tvautor.autornr AS autornr
     , Tvautor.lfdnr   AS lfdnra
FROM   Tisbn, Tvautor
WHERE  Tisbn.Buchnr = Tvautor.Buchnr
;
```

```
buchnr      isbn       lfdnri           autornr     lfdnra
----------- ---------- ---------------- ----------- -----------
8           3472864303 1                2           0
8           34728643yx 2                2           0
12          0201501139 1                20          1
12          0201501139 1                21          2
```

- **Der Inner Natural-Join (auch kurz Join genannt) ist das Gegenstück zur Normalisierung.**
- Im Beispiel mit Tbuch und Tisbn wird der Join mit Hilfe von Primärschlüssel- und Fremdschlüsselspalte gebildet.
- Die Ergebnistable im Beispiel mit Tbuch und Tisbn entsteht konzeptionell aus dem kartesischen Produkt mit Hilfe einer Restriktion (Auswahl von Zeilen mit Hilfe der Joinbedingung) und einer Projektion!
- Die Spaltennamen Buchnr sind qualifiziert!
- Weglassen der Join-Bedingung führt zu einer "unerwartetet" großen Ergebnistable.
- Das Ergebnis präsentiert nur eine Spalte Buchnr!
- **Beim Natural-Join wird die Join-Spalte nur einmal angelistet!**
- Ein Join ist über beliebige Spalten möglich, sofern es fachlich Sinn macht und die Datentypen passen. Beachten Sie bitte: zwischen Tisbn und Tvautor besteht keine Fremdschlüssel- / Primärschlüsselbeziehung!
- Wie bewerten Sie folgende Anweisungen? Der Zugriff auf die Table-Tisbn ist überflüssig!

```
SELECT Tbuch.Buchnr, Tisbn.Isbn
FROM Tbuch, Tisbn
WHERE Tbuch.Buchnr = Tisbn.Buchnr
;
SELECT Tisbn.Buchnr, Tisbn.Isbn
FROM Tisbn
;
    Buchnr      Isbn
    ----------- ----------
    8           3472864303
    8           34728643yx
    12          0201501139
```

5.7 Der Inner Natural-Join, syntaktische Varianten

```
SELECT Tbuch.Buchnr, Isbn, Preis
FROM   Tbuch, Tisbn
WHERE  Tbuch.Buchnr = Tisbn.Buchnr
AND    Erschj = 1977
;
SELECT Tbuch.Buchnr, Isbn, Preis
FROM   Tisbn INNER JOIN Tbuch
       ON Tisbn.Buchnr = Tbuch.Buchnr
WHERE  Erschj = 1977
;
SELECT Tbuch.Buchnr, Isbn, Preis
FROM   Tisbn INNER JOIN Tbuch
       ON Tisbn.Buchnr = Tbuch.Buchnr
       AND Erschj = 1977
;
SELECT Tbuch.Buchnr, Isbn, Preis
FROM
(
       Tisbn INNER JOIN Tbuch
       ON Tisbn.Buchnr = Tbuch.Buchnr
)
WHERE  Erschj = 1977
;
Buchnr      Isbn       Preis
----------- ---------- ---------
8           3472864303 .50
8           34728643yx .50
```

- Die SELECT-Anweisungen sind was-äquivalent, d.h. sie liefern für jeden möglichen Testbestand dasselbe Ergebnis.
- Beachten Sie die Schreibweise bei der Variante mit INNER JOIN (**ON...WHERE bzw. ON…AND** statt **WHERE...AND**). Der Join-Typ INNER kann auch weggelassen werden, da er als Default genommen wird.
- **Alternative Formulierungen:**

```
SELECT
 Tab1.Buchnr AS Buchnr
,Tab2.Isbn   AS Isbn
,Tab1.Preis  AS Preis
FROM
( SELECT Tbuch.Buchnr  AS Buchnr
        ,Tbuch.Preis   AS Preis
  FROM   Tbuch
  WHERE  Tbuch.Erschj = 1977
) tab1
,
( SELECT Tisbn.Buchnr  AS Buchnr
        ,Tisbn.Isbn    AS Isbn
  FROM   Tisbn
) tab2
WHERE  Tab1.Buchnr = Tab2.Buchnr
;

SELECT
 Tab1.Buchnr AS Buchnr
,Tab2.Isbn   AS Isbn
,Tab1.Preis  AS Preis
FROM
( SELECT Tbuch.Buchnr  AS Buchnr
        ,Tbuch.Preis   AS Preis
  FROM   Tbuch
  WHERE  Tbuch.Erschj = 1977
) tab1
INNER JOIN
( SELECT Tisbn.Buchnr  AS Buchnr
        ,Tisbn.Isbn    AS Isbn
  FROM   Tisbn
) tab2
ON Tab1.Buchnr = Tab2.Buchnr
;
```

5.8 Inner Natural-Join und mehr als 2 Tables

```
SELECT
 Tbuch.Buchnr      AS Buchnr
,Tbuch.Preis       AS Preis
,Tbuch.Erschj      AS Erschj
,Tisbn.Lfdnr       AS Lfdnr
,Tisbn.ISBN        AS Isbn
,Tvautor.Autornr   AS Autornr
FROM   Tisbn, Tbuch, Tvautor
WHERE  Tisbn.Buchnr = Tbuch.Buchnr
AND    Tbuch.Buchnr = Tvautor.Buchnr
;

 SELECT
 Tbuch.Buchnr      AS Buchnr
,Tbuch.Preis       AS Preis
,Tbuch.Erschj      AS Erschj
,Tisbn.Lfdnr       AS Lfdnr
,Tisbn.ISBN        AS Isbn
,Tvautor.Autornr   AS Autornr
FROM   Tisbn INNER JOIN Tbuch
       ON Tisbn.Buchnr = Tbuch.Buchnr
       INNER JOIN Tvautor
       ON Tbuch.Buchnr = Tvautor.Buchnr
;
```

```
Buchnr      Preis     Erschj Lfdnr Isbn       Autornr
----------- --------- ------ ----- ---------- -----------
8           .50       1977   1     3472864303 2
8           .50       1977   2     34728643yx 2
12          NULL      1989   1     0201501139 20
12          NULL      1989   1     0201501139 21
```

```
SELECT
 SUBSTRING(Tbuch.Titel, 1, 10)      AS Titel
,tbuch.buchnr                       as buchnr
,tautor.autornr                     as autornr
,SUBSTRING(Tautor.Autor,1,10)       AS Autor
FROM   Tbuch, Tvautor, Tautor
WHERE  Tbuch.Buchnr    = Tvautor.Buchnr
AND    Tvautor.Autornr = Tautor.Autornr
ORDER BY Titel, buchnr, autornr, Autor
;
Titel      buchnr      autornr     Autor
---------- ----------- ----------- ----------
a guide to 12          20          C. J. Date
a guide to 12          21          Colin J. W
Ansichten  5           1           Boell
C          1           100         BUSCH
C          1           200         BUSCH
C          2           200         BUSCH
DB2 fuer S 9           10          Emil Hack
DB2 fuer S 9           11          Frieda Hol
der Butt   8           2           Grass
der Name d 7           3           Eco
die Blecht 6           2           Grass
```

Die folgende SELECT-Anweisung ist syntaktisch korrekt, sie implementiert aber keine relationale Projektion und liefert eine Liste mit Duplikaten. **Das Ergebnis ist keine Table/Relation!**

```
SELECT
 SUBSTRING(Tbuch.Titel, 1, 10)      AS Titel
----,tbuch.buchnr                       as buchnr
----,tautor.autornr                     as autornr
,SUBSTRING(Tautor.Autor,1,10)       AS Autor
FROM   Tbuch, Tvautor, Tautor
WHERE  Tbuch.Buchnr    = Tvautor.Buchnr
AND    Tvautor.Autornr = Tautor.Autornr
ORDER BY Titel, Autor
;
Titel      Autor
---------- ----------
a guide to C. J. Date
a guide to Colin J. W
Ansichten  Boell
C          BUSCH
C          BUSCH
C          BUSCH
DB2 fuer S Emil Hack
DB2 fuer S Frieda Hol
der Butt   Grass
der Name d Eco
die Blecht Grass
```

5.9 JOIN und GROUP BY

Tbuch

Buchnr	Titel
9	DB2 fuer Sie

Tvautor

Buchnr	Autornr	Praemie	Lfdnr
9	10	10.00	1
9	11	498.00	2

Aufgabe: Für jedes Buch hole Buchnr, Titel und Summe der Autorenprämie.

Buchnr	Titel	Sumprae
9	DB2 fuer Sie	508.00

Variante 1:

```
SELECT
 Tbuch.Buchnr         AS Buchnr
,Tbuch.Titel          AS Titel
,xx.sumprae           AS Sumprae
FROM
      Tbuch
    INNER JOIN
      ( SELECT Tvautor.Buchnr AS Buchnr
       ,SUM(Tvautor.Praemie)  AS Sumprae
       FROM Tvautor
       GROUP BY Tvautor.Buchnr
      ) XX
    ON Tbuch.Buchnr = XX.Buchnr
;
```

Variante 2:

```
SELECT
 Tbuch.Buchnr          AS Buchnr
,Tbuch.Titel           AS Titel
,SUM(Tvautor.Praemie) AS Sumprae
FROM
      Tbuch INNER JOIN Tvautor
      ON Tbuch.Buchnr = Tvautor.Buchnr
GROUP BY Tbuch.Buchnr, Tbuch.Titel
;
```

- In der from-clause kann geschachtelt werden (abgeleitete Table, derived table). Dabei ist ein Aliasname erforderlich!
- Beachten Sie bitte, dass bei Variante 2 und Variante 4 in der GROUP BY Klausel zwingend **`GROUP BY Tbuch.Buchnr, Tbuch.Titel`** codiert werden muss. Alle Spalten der SELECT-Klausel ohne Aggregatfunktion müssen angegeben werden. Ein Coding von `GROUP BY Tbuch.Buchnr` würde zu einem Syntaxfehler führen.

Variante 3 ohne "inner join":

```
SELECT
 Tbuch.Buchnr         AS Buchnr
,Tbuch.Titel          AS Titel
,xx.sumprae           AS Sumprae
FROM
      Tbuch
      ,
      ( SELECT Tvautor.Buchnr AS Buchnr
       ,SUM(Tvautor.Praemie)  AS Sumprae
       FROM Tvautor
       GROUP BY Tvautor.Buchnr
      ) XX
WHERE Tbuch.Buchnr = XX.Buchnr
;
```

Variante 4 ohne "inner join":

```
SELECT
 Tbuch.Buchnr          AS Buchnr
,Tbuch.Titel           AS Titel
,SUM(Tvautor.Praemie) AS Sumprae
FROM
      Tbuch, Tvautor
WHERE Tbuch.Buchnr = Tvautor.Buchnr
GROUP BY Tbuch.Buchnr, Tbuch.Titel
;
```

5.10 Join einer Table mit sich selbst, Theta-Join, Between-Join

Was ist das Ergebnis der folgenden SELECT-Anweisung?

```
SELECT A.Buchnr as abuchnr
      ,A.Erschj as aerschj
      ,B.Buchnr as bbuchnr
      ,B.Erschj as berschj
FROM (
      Tbuch A INNER JOIN Tbuch B
      ON A.Erschj <> B.Erschj
     )
WHERE A.Buchnr < B.Buchnr
ORDER BY abuchnr, bbuchnr
;
```

abuchnr	aerschj	bbuchnr	berschj
5	1988	7	1989
5	1988	8	1977
5	1988	9	1990
5	1988	11	1990
5	1988	12	1989
5	1988	18	1989
6	1988	7	1989
6	1988	8	1977
6	1988	9	1990
6	1988	11	1990
6	1988	12	1989
6	1988	18	1989
7	1989	8	1977
7	1989	9	1990
7	1989	11	1990
8	1977	9	1990
8	1977	11	1990
8	1977	12	1989
8	1977	18	1989
9	1990	12	1989
9	1990	18	1989
11	1990	12	1989
11	1990	18	1989

- Äquivalente Formulierung:

```
SELECT A.Buchnr as abuchnr
      ,A.Erschj as aerschj
      ,B.Buchnr as bbuchnr
      ,B.Erschj as berschj
FROM  Tbuch A, Tbuch B
WHERE A.Erschj <> B.Erschj
AND   A.Buchnr < B.Buchnr
ORDER BY abuchnr, bbuchnr
;
```

- **Die SELECT-Anweisung benützt Korrelationsnamen (oft auch Range Variable genannt).**
- Wir nehmen an, es gäbe eine Kopie der Table Tbuch, ansprechbar unter dem Namen A und eine weitere Kopie der Table Tbuch, ansprechbar unter dem Namen B.
- Das System muss die Kopien A bzw. B nicht physisch erstellen, die Betrachtung ist konzeptioneller Art!
- Bei diesem Beispiel ist die Verwendung von Korrelationsnamen erforderlich!
- Das Beispiel ist ein so genannter Inner Theta-Join (die Join-Spalte Erschj wird zweimal angelistet und der Vergleichsoperator ist <>).

Ein anderes Beispiel: Paare von Buchnummern buchnr1, buchnr2 mit buchnr1<buchnr2 die wenigstens einen Autor haben und wenigstens einen gemeinsamen Autor.

```
SELECT A.Buchnr as buchnr1
      ,B.Buchnr as buchnr2
FROM (
      Tvautor A INNER JOIN Tvautor B
      ON A.Autornr = B.Autornr
      )
WHERE A.Buchnr < B.Buchnr
;
buchnr1     buchnr2
----------- -----------
1           2
6           8
```

BETWEEN-Join

```
Tkategorie
kategorie  vonclosed                                bisclosed
---------- ---------------------------------------- ------------------
K-         -999999999999999.99999                   -0.00001
K0         0.00000                                  9.99999
K1         10.00000                                 19.99999
K2         20.00000                                 29.99999
K3         30.00000                                 39.99999
Kyy        40.00000                                 9999.99999
K+         10000.00000
999999999999999.99999
```

```
SELECT
   aaa.buchnr, aaa.preis, bbb.kategorie
FROM
  tbuch aaa, tkategorie bbb
WHERE
   aaa.Preis BETWEEN bbb.vonclosed AND bisclosed
;
```

```
buchnr     preis                                    kategorie
---------- ---------------------------------------- ----------
5          3.50                                     K0
8          0.50                                     K0
6          20.50                                    K2
7          99.99                                    Kyy
9          55.00                                    Kyy
18         99.99                                    Kyy
27         99.99                                    Kyy
```

```
drop table tkategorie
;
create table tkategorie
(
  vonclosed  decimal (20,5) NOT NULL
 ,bisclosed  decimal (20,5) NOT NULL
 ,kategorie  CHAR (10)      NOT NULL
 ,constraint con1 UNIQUE (kategorie)
 ,constraint con2 UNIQUE (vonclosed)
 ,constraint con3 UNIQUE (bisclosed)
 ,constraint con4 CHECK(vonclosed <=bisclosed)
 --overlaps ist möglich
 --meets bei gleichem wert ist möglich
 )
 ;
 INSERT INTO tkategorie (vonclosed, bisclosed, kategorie)
 values (  -999999999999999.99999 , -0.00001, 'K-')
 ;
 INSERT INTO tkategorie (vonclosed, bisclosed, kategorie)
 values (  0 , 9.99999, 'K0')
 ;
 INSERT INTO tkategorie (vonclosed, bisclosed, kategorie)
 values (  10 , 19.99999, 'K1')
 ;
 INSERT INTO tkategorie (vonclosed, bisclosed, kategorie)
 values (  20 , 29.99999, 'K2')
 ;
 INSERT INTO tkategorie (vonclosed, bisclosed, kategorie)
 values (  30 , 39.99999, 'K3')
 ;
 INSERT INTO tkategorie (vonclosed, bisclosed, kategorie)
 values (  40 , 9999.99999, 'Kyy')
 ;
 INSERT INTO tkategorie (vonclosed, bisclosed, kategorie)
 values ( +10000, +999999999999999.99999 , 'K+')
 ;
 select kategorie,vonclosed, bisclosed
 FROM tkategorie
 ;
```

5.11 Der Left Outer Natural-Join

```
SELECT Tbuch.Buchnr, Tbuch.Erschj, Tisbn.ISBN
FROM
     (
       Tbuch LEFT OUTER JOIN Tisbn
       ON  Tbuch.Buchnr = Tisbn.Buchnr
    )
;
```

Buchnr	Erschj	ISBN
1	NULL	NULL
2	NULL	NULL
5	1988	NULL
6	1988	NULL
7	1989	NULL
8	*1977*	*3472864303*
8	*1977*	*34728643yx*
12	*1989*	*0201501139*
9	1990	NULL
11	1990	NULL
18	1989	NULL
27	NULL	NULL

- Zusätzlich zu den Büchern mit ISBN (Zeilen des Inner Natural-Joins) werden alle Bücher aus **der linken Table Tbuch** angedruckt, die keine ISBN haben.
- Die folgenden SELECT-Anweisungen sind was-äquivalent:

```
SELECT Tbuch.Buchnr, Tbuch.Erschj, Tisbn.ISBN
FROM
     (
       Tbuch LEFT OUTER JOIN Tisbn
       ON  Tbuch.Buchnr = Tisbn.Buchnr
    )
;
SELECT Tbuch.Buchnr, Tbuch.Erschj, Tisbn.ISBN
FROM
       Tbuch LEFT OUTER JOIN Tisbn
       ON  Tbuch.Buchnr = Tisbn.Buchnr
;
SELECT Tbuch.Buchnr, Tbuch.Erschj, Tisbn.ISBN
FROM
    (
      Tisbn RIGHT OUTER JOIN Tbuch
      ON  Tbuch.Buchnr = Tisbn.Buchnr
    )
;
SELECT Tbuch.Buchnr, Tbuch.Erschj, Tisbn.ISBN
FROM
      Tisbn RIGHT OUTER JOIN Tbuch
      ON  Tbuch.Buchnr = Tisbn.Buchnr
;
```

5.11.1 Der Left Outer Natural-Join, Bedingung links

--Beispiel WHERE links

Alle Bücher von 1977 und 1988 und ihre ISBNs sofern vorhanden.

```
SELECT
 Tbuch.Buchnr                    AS Buchnr
,Tisbn.Lfdnr                     AS Lfdnri
,Tisbn.Isbn                      AS Isbn
,Tbuch.Erschj                    AS Erschj
FROM
     TBUCH LEFT OUTER JOIN TISBN
     ON TBUCH.Buchnr = TISBN.Buchnr
WHERE tbuch.erschj IN (1977, 1988)
;
```

```
Buchnr      Lfdnri                                 Isbn       Erschj
----------- -------------------------------------- ---------- --------
5           NULL                                   NULL       1988
6           NULL                                   NULL       1988
8           1                                      3472864303 1977
8           2                                      34728643yx 1977

(4 Zeile(n) betroffen)
```

--Beispiel AND links

Alle Bücher und die ISBNs von 1977 und 1988 sofern vorhanden.

```
SELECT
 Tbuch.Buchnr                    AS Buchnr
,Tisbn.Lfdnr                     AS Lfdnri
,Tisbn.Isbn                      AS Isbn
,Tbuch.Erschj                    AS Erschj
FROM
     TBUCH LEFT OUTER JOIN TISBN
     ON TBUCH.Buchnr = TISBN.Buchnr
AND tbuch.erschj IN (1977, 1988)
;
```

```
Buchnr      Lfdnri                                 Isbn       Erschj
----------- -------------------------------------- ---------- ----------
1           NULL                                   NULL       NULL
2           NULL                                   NULL       NULL
5           NULL                                   NULL       1988
6           NULL                                   NULL       1988
7           NULL                                   NULL       1989
8           1                                      3472864303 1977
8           2                                      34728643yx 1977
9           NULL                                   NULL       1990
11          NULL                                   NULL       1990
12          NULL                                   NULL       1989
18          NULL                                   NULL       1989
27          NULL                                   NULL       NULL

(12 Zeile(n) betroffen)
```

--Beispiel WHERE links

Mit Hilfe einer Restriktion (d. h. mit Hilfe der WHERE-Klausel) können aus dem Ergebnis des Left Outer Join gezielt Zeilen ausgewählt werden. Die Bedingung `tbuch.erschj IN (1977, 1988)` ist Bestandteil der WHERE-Klausel.

--Beispiel AND links

Die Bedingung `tbuch.erschj IN (1977, 1988)` ist Bestandteil der Join-Operation in der FROM-Klausel.

Nur für Bücher von 1977 und 1988 werden eventuell vorhandene ISBN's angelistet. Beachten Sie bitte die Zeile mit Buchnr = 12, diese Zeile präsentiert keinen Wert für die ISBN sondern die NULL.

Das Beispiel AND links ist eine **„elegante Formulierung"** für folgende logisch äquivalente Select-Anweisungen.

```
SELECT
 Tbuch.Buchnr                    AS Buchnr
,zwi.Lfdnr                       AS Lfdnri
,zwi.Isbn                        AS Isbn
,Tbuch.Erschj                    AS Erschj
FROM
   Tbuch LEFT OUTER JOIN
   (
        select
        aaa.buchnr as buchnr
       ,aaa.erschj as erschj
       ,tisbn.lfdnr  as lfdnr
       ,tisbn.isbn   as isbn
        from  tbuch aaa INNER JOIN Tisbn
        ON aaa.buchnr = tisbn.buchnr
           AND aaa.erschj IN (1977, 1988)
   ) zwi
  ON tbuch.buchnr = zwi.buchnr
;
SELECT
 Tbuch.Buchnr                    AS Buchnr
,Tisbn.Lfdnr                     AS Lfdnri
,Tisbn.Isbn                      AS Isbn
,Tbuch.Erschj                    AS Erschj
FROM
   Tbuch LEFT OUTER JOIN
   (
        TBUCH aaa INNER JOIN Tisbn
        ON aaa.buchnr = tisbn.buchnr
           AND aaa.erschj IN (1977, 1988)
   )
  ON tbuch.buchnr = aaa.buchnr
;
```

EMPFEHLUNG: Codieren Sie immer mit Klammern, das folgende Coding liefert dann einen Syntaxfehler!

```
FROM
  (
       TBUCH LEFT OUTER JOIN TISBN
       ON TBUCH.Buchnr = TISBN.Buchnr
  )
AND tbuch.erschj IN (1977, 1988)
```

5.11.2 Der Left Outer Natural-Join, Bedingung rechts

--Beispiel WHERE rechts

Alle Bücher die eine 1. Isbn haben, nur diese 1. ISBN mit anlisten

```
SELECT
 Tbuch.Buchnr                    AS Buchnr
,Tisbn.Lfdnr                     AS Lfdnri
,Tisbn.Isbn                      AS Isbn
,Tbuch.Erschj                    AS Erschj
FROM
     TBUCH LEFT OUTER JOIN TISBN
     ON TBUCH.Buchnr = TISBN.Buchnr
WHERE tisbn.lfdnr =1
;
```

```
Buchnr      Lfdnri                                 Isbn       Erschj
----------- -------------------------------------- ---------- -------
8           1                                      3472864303 1977
12          1                                      0201501139 1989

(2 Zeile(n) betroffen)
```

--Beispiel AND rechts

Alle Bücher und dazu die 1. Isbn sofern vorhanden

```
SELECT
 Tbuch.Buchnr                    AS Buchnr
,Tisbn.Lfdnr                     AS Lfdnri
,Tisbn.Isbn                      AS Isbn
,Tbuch.Erschj                    AS Erschj
FROM
     TBUCH LEFT OUTER JOIN TISBN
     ON TBUCH.Buchnr = TISBN.Buchnr
AND tisbn.lfdnr =1
;
```

```
Buchnr      Lfdnri                                 Isbn       Erschj
----------- -------------------------------------- ---------- --------
1           NULL                                   NULL       NULL
2           NULL                                   NULL       NULL
5           NULL                                   NULL       1988
6           NULL                                   NULL       1988
7           NULL                                   NULL       1989
8           1                                      3472864303 1977
9           NULL                                   NULL       1990
11          NULL                                   NULL       1990
12          1                                      0201501139 1989
18          NULL                                   NULL       1989
27          NULL                                   NULL       NULL

(11 Zeile(n) betroffen)
```

--Beispiel WHERE rechts

Mit Hilfe einer Restriktion (d. h. mit Hilfe der WHERE-Klausel) können aus dem Ergebnis des Left Outer Join gezielt Zeilen ausgewählt werden. Die Bedingung tisbn.lfdnr=1 ist Bestandteil der WHERE-Klausel.

Da diese Bedingung sich auf die rechte Table Tisbn bezieht wird aus dem Left Outer Join - logisch betrachtet - ein Inner Join.

```
SELECT
 Tbuch.Buchnr                    AS Buchnr
,Tisbn.Lfdnr                     AS Lfdnri
,Tisbn.Isbn                      AS Isbn
,Tbuch.Erschj                    AS Erschj
FROM
     TBUCH INNER JOIN TISBN
     ON TBUCH.Buchnr = TISBN.Buchnr
WHERE tisbn.lfdnr =1
;
```

--Beispiel AND rechts

Die Bedingung tisbn.lfdnr=1 ist Bestandteil der Join-Operation in der FROM-Klausel.

Für Bücher mit ISBNs wird nur die erste ISBN's angelistet. Beachten Sie bitte die Zeile mit Buchnr = 8, diese Zeile präsentiert die erste Isbn. Es gibt keine Zeile für die Buchnr=8 mit der zweiten Isbn.

Das Beispiel AND rechts ist eine **„elegante Formulierung"** für folgende logisch äquivalente Select-Anweisung.

```
SELECT
 Tbuch.Buchnr                    AS Buchnr
,zwi.Lfdnr                       AS Lfdnri
,zwi.Isbn                        AS Isbn
,Tbuch.Erschj                    AS Erschj
FROM
     Tbuch LEFT OUTER JOIN
     (
      select bbb.buchnr as buchnr
           , bbb.isbn   as isbn
           , bbb.lfdnr  as lfdnr
      from tisbn bbb
      where bbb.lfdnr =1
     ) zwi
     ON Tbuch.Buchnr = zwi.Buchnr
```

5.11.3 Der Left Outer Natural-Join und mehr als 2 Tables

Mit Hilfe einer **Restriktion** und einer **Projektion** können jetzt gezielt Fragen beantwortet werden.

- Gibt es wenigstens ein Buch im Bestand mit Titel ‚C' und Autor ‚BUSCH'?

Beachten Sie bitte, dass bei einer Projektion eventuell DISTINCT codiert werden muss!

```
SELECT  DISTINCT
-- Tbuch.Buchnr                      AS Buchnr
SUBSTRING(Tbuch.Titel, 1, 5)    AS Titel
--,Tbuch.Erschj                      AS Erschj
--,Tbuch.Preis                       AS Preis
--,Tbuch.Verlagnr                    AS Verlagnr
--,Tisbn.Lfdnr                       AS Lfdnri
--,Tisbn.Isbn                        AS Isbn
--,Tvautor.Lfdnr                     AS Lfdnra
--,Tvautor.Praemie                   AS Praemie
--,Tautor.Autornr                    AS Autornr
,SUBSTRING(Tautor.Autor,1,5)     AS Autor  -- SQL Server
--,SUBSTR(Tautor.Autor,1,5)         AS Autor  -- Oracle, DB2
--,Tautor.Geburtsdatum              AS Geburtsdatum
FROM
(
     TBUCH LEFT OUTER JOIN TISBN
     ON TBUCH.Buchnr = TISBN.Buchnr
           LEFT OUTER JOIN TVAUTOR
     ON TBUCH.buchnr = TVAUTOR.buchnr
           LEFT OUTER JOIN TAUTOR
     ON TVAUTOR.autornr = TAUTOR.autornr
)
WHERE
      Tbuch.Titel = 'C'
  AND UPPER(Tautor.Autor) = 'BUSCH'
----ORDER BY Buchnr, Lfdnri, Lfdnra, Autornr
;
```

Beachten Sie bitte, dass die Join-Bedingungen immer die Fremdschlüssel-Primärschlüssel-Beziehungen nutzen!

Wir codieren

```
ON TBUCH.buchnr = TVAUTOR.buchnr
```

und wir codieren nicht

```
ON TISBN.buchnr = TVAUTOR.buchnr
```

Beachten Sie bitte, dass in der SELECT-Klausel die Primärschlüsselspalten benutzt werden und nicht die Fremdschlüsselspalten!

Wir codieren

```
,Tautor.Autornr                 AS Autornr
```

und wir codieren nicht

```
,Tvautor.Autornr                AS Autornr
```

5.12 LEFT OUTER JOIN, RIGTH OUTER JOIN, FULL OUTER JOIN

```
Tab1                  Tab2
Tab1schl S1            S1    Tab2schl
1        5             5     aa
2        6             5     bb
3        6             6     cc
4        7             8     dd

SELECT *
FROM Tab1 INNER JOIN Tab2
     ON Tab1.S1=Tab2.S1
;
Tab1schl S1            S1    Tab2schl
1        5             5        aa
1        5             5        bb
2        6             6        cc
3        6             6        cc

SELECT *
FROM Tab1 LEFT OUTER JOIN Tab2
     ON Tab1.S1=Tab2.S1
;
Tab1schl S1            S1    Tab2schl
1        5             5        aa
1        5             5        bb
2        6             6        cc
3        6             6        cc
4        7             NULL     NULL

SELECT *
FROM Tab1 RIGHT OUTER JOIN Tab2
     ON Tab1.S1=Tab2.S1
;
Tab1schl S1            S1    Tab2schl
1        5             5        aa
1        5             5        bb
2        6             6        cc
3        6             6        cc
NULL     NULL          8        dd

SELECT *
FROM Tab1 FULL OUTER JOIN Tab2
     ON Tab1.S1=Tab2.S1
;
Tab1schl S1            S1       Tab2schl
1        5             5        aa
1        5             5        bb
2        6             6        cc
3        6             6        cc
4        7             NULL     NULL
NULL     NULL          8        dd
```

LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN haben alle mit der NULL zu tun! OUTER ist optional und kann auch weggelassen werden.

Das ist der Full Outer Natural-Join zwischen Tab1 und Tab2 (die Join-Spalte wird nur einmal angelistet):

```
SELECT
 Tab1schl
,COALESCE (Tab1.S1, Tab2.S1) AS Sxy
,Tab2schl
FROM Tab1 FULL OUTER JOIN Tab2
     ON Tab1.S1=Tab2.S1
;
```

Tab1schl	Sxy	Tab2schl
1	5	aa
1	5	bb
2	6	cc
3	6	cc
4	**7**	**NULL**
NULL	**8**	**dd**

COALESCE (Tab1.S1, Tab2.S1) liefert NULL genau dann, wenn beide Operanten NULL sind, ansonsten wird der Wert des ersten nicht-NULL Operanten geliefert.

```
drop table tab1
;
create table tab1
( tab1schl integer not null
 ,s1      integer not null
 ,primary key(tab1schl)
)
;
insert into tab1( tab1schl, s1) values( 1, 5);
insert into tab1( tab1schl, s1) values( 2, 6);
insert into tab1( tab1schl, s1) values( 3, 6);
insert into tab1( tab1schl, s1) values( 4, 7);
select tab1schl, s1 from tab1;
---------------
drop table tab2
;
create table tab2
( tab2schl varchar(10) not null
 ,s1      integer not null
 ,primary key(tab2schl)
)
;
insert into tab2( tab2schl, s1) values(  'aa', 5);
insert into tab2( tab2schl, s1) values(  'bb', 5);
insert into tab2( tab2schl, s1) values(  'cc', 6);
insert into tab2( tab2schl, s1) values(  'dd', 8);
select tab2schl, s1 from tab2;
```

```
--INNER JOIN
--links where
select
  tab1.tab1schl, tab1.s1, tab2.s1, tab2.tab2schl
FROM tab1 tab1 INNER JOIN tab2 tab2
     ON tab1.s1 = tab2.s1
WHERE tab1.s1 IN (5,7)
;
--links and
select
  tab1.tab1schl, tab1.s1, tab2.s1, tab2.tab2schl
FROM tab1 tab1 INNER JOIN tab2 tab2
     ON tab1.s1 = tab2.s1
     AND tab1.s1 IN (5,7)
;
---------------
--LEFT OUTER JOIN
--links where
select
  tab1.tab1schl, tab1.s1, tab2.s1, tab2.tab2schl
FROM tab1 tab1 LEFT OUTER JOIN tab2 tab2
     ON tab1.s1 = tab2.s1
WHERE tab1.s1 IN (5,7)
;
--links and
select
  tab1.tab1schl, tab1.s1, tab2.s1, tab2.tab2schl
FROM tab1 tab1 LEFT OUTER JOIN tab2 tab2
     ON tab1.s1 = tab2.s1
     AND tab1.s1 IN (5,7)
;
---------------
--RIGHT OUTER JOIN
--links where
select
  tab1.tab1schl, tab1.s1, tab2.s1, tab2.tab2schl
FROM tab1 tab1 RIGHT OUTER JOIN tab2 tab2
     ON tab1.s1 = tab2.s1
WHERE tab1.s1 IN (5,7)
;
--links and
select
  tab1.tab1schl, tab1.s1, tab2.s1, tab2.tab2schl
FROM tab1 tab1 RIGHT OUTER JOIN tab2 tab2
     ON tab1.s1 = tab2.s1
     AND tab1.s1 IN (5,7)
;
---------------
--FULL OUTER JOIN
--links where
select
  tab1.tab1schl, tab1.s1, tab2.s1, tab2.tab2schl
FROM tab1 tab1 FULL OUTER JOIN tab2 tab2
     ON tab1.s1 = tab2.s1
WHERE tab1.s1 IN (5,7)
;
--links and
select
  tab1.tab1schl, tab1.s1, tab2.s1, tab2.tab2schl
FROM tab1 tab1 FULL OUTER JOIN tab2 tab2
     ON tab1.s1 = tab2.s1
     AND tab1.s1 IN (5,7)
;
```

```
tab1schl    s1          s1          tab2schl
----------- ----------- ----------- ----------
1           5           5           aa
1           5           5           bb


tab1schl    s1          s1          tab2schl
----------- ----------- ----------- ----------
1           5           5           aa
1           5           5           bb

tab1schl    s1          s1          tab2schl
----------- ----------- ----------- ----------
1           5           5           aa
1           5           5           bb
4           7           NULL        NULL


tab1schl    s1          s1          tab2schl
----------- ----------- ----------- ----------
1           5           5           aa
1           5           5           bb
2           6           NULL        NULL
3           6           NULL        NULL
4           7           NULL        NULL


tab1schl    s1          s1          tab2schl
----------- ----------- ----------- ----------
1           5           5           aa
1           5           5           bb


tab1schl    s1          s1          tab2schl
----------- ----------- ----------- ----------
1           5           5           aa
1           5           5           bb
NULL        NULL        6           cc
NULL        NULL        8           dd


tab1schl    s1          s1          tab2schl
----------- ----------- ----------- ----------
1           5           5           aa
1           5           5           bb
4           7           NULL        NULL


tab1schl    s1          s1          tab2schl
----------- ----------- ----------- ----------
1           5           5           aa
1           5           5           bb
2           6           NULL        NULL
3           6           NULL        NULL
4           7           NULL        NULL
NULL        NULL        6           cc
NULL        NULL        8           dd
```

5.13 Der Full Outer Natural-Join und COALESCE

--(Tisbn full Tvautor) full Tbuch

```
SELECT
 COALESCE (zwi.buchnr, tbuch.buchnr) as buchnr
,zwi.lfdnri                          as lfdnri
,zwi.isbn                            as isbn
,zwi.autornr                         as autornr
,zwi.lfdnra                          as lfdnra
,zwi.praemie                         as praemie
,tbuch.titel                         as titel
,tbuch.erschj                        as erschj
,tbuch.preis                         as preis
,tbuch.verlagnr                      as verlagnr
FROM
(
       SELECT
        COALESCE
        (tisbn.buchnr, tvautor.buchnr)
                             AS buchnr
       ,tisbn.lfdnr          AS lfdnri
       ,tisbn.isbn           AS isbn
       ,tvautor.autornr      AS autornr
       ,tvautor.lfdnr        AS lfdnra
       ,tvautor.praemie      AS praemie
       FROM tisbn FULL OUTER JOIN tvautor
            ON tisbn.buchnr = tvautor.buchnr
)
       Zwi          FULL OUTER JOIN tbuch
       ON zwi.buchnr = tbuch.buchnr
;
```

Das ist der Full Outer Natural-Join zwischen (Tisbn full Tvautor) und Tbuch.

Beachten Sie bitte: zwischen Tisbn und Tvautor besteht keine Fremdschlüssel- / Primärschlüsselbeziehung!

--(Tisbn full Tbuch) full Tvautor

```
select
 tbuch.buchnr          AS buchnr
,tisbn.lfdnr           AS lfdnri
,tisbn.isbn            AS isbn
,tbuch.titel           AS titel
,tbuch.erschj          AS erschj
,tbuch.preis           AS preis
,tbuch.verlagnr        AS verlagnr
,tvautor.autornr       as autornr
,tvautor.lfdnr         as lfdnra
FROM
(
     tisbn FULL OUTER JOIN tbuch
     ON tisbn.buchnr = tbuch.buchnr
           FULL OUTER JOIN tvautor
     ON Tbuch.buchnr = tvautor.buchnr
)
;
```

Das ist eine syntaktische Variante des Full Outer Natural-Joins zwischen (Tisbn full Tbuch) und Tvautor.

Beachten Sie bitte, dass die Join-Bedingung jeweils über die Fremdschlüssel-/Primärschlüsselbeziehung formuliert ist!

5.14 Subquery mit EXISTS oder IN

Bücher mit wenigstens einer ISBN

```
SELECT
  Tbuch.Buchnr, Tbuch.Titel FROM Tbuch
WHERE Tbuch.Buchnr IN
     (SELECT Tisbn.Buchnr FROM Tisbn)
;
SELECT
  Tbuch.Buchnr, Tbuch.Titel FROM Tbuch
WHERE EXISTS
     (SELECT * FROM Tisbn
      WHERE Tisbn.Buchnr = Tbuch.Buchnr)
;
SELECT DISTINCT
  Tbuch.Buchnr, Tbuch.Titel
FROM Tbuch INNER JOIN Tisbn
     ON Tbuch.Buchnr = Tisbn.Buchnr
;
```

Die obigen Select-Anweisungen sind äquivalent, sie liefern für jeden möglichen Datenbestand dasselbe Ergebnis. **Beachten Sie DISTINCT beim Join.**

Die Wahrheitswerte für **"WHERE EXISTS (...)"** sind

- **wahr** die von der Subquery ermittelte Menge ist nicht leer **(enthält wenigstens eine Zeile)**
- **falsch** die von der Subquery ermittelte Menge ist leer **(enthält keine Zeile)**

Die Wahrheitswerte für **"WHERE ... IN (...)"** sind wahr, unbekannt bzw. falsch.

*Wenn die **Table Tisbn wenigstens eine Zeile** präsentiert, dann qualifizieren bei folgender SELECT-Anweisung **alle Zeilen aus Tbuch** (die Subquery ist nicht-korreliert).*

```
SELECT Buchnr, Titel FROM Tbuch
WHERE EXISTS (SELECT * FROM Tisbn)
;
```

- **IN-Subquery**: Jede Zeile aus Tbuch wird geprüft, ob ihre Buchnr enthalten ist in der Menge aller Buchnr aus Tisbn. Konzeptionell wird zunächst die Menge aller Buchnr von Tisbn gebildet, und dann jede Buchnr aus Tbuch gegen diese Menge geprüft. Der innere SELECT (die query expression) in der obigen IN-Subquery kann getrennt getestet werden! Eine solche Subquery wird oft auch nicht-korrelierte Subquery genannt!
- **EXISTS-Subquery**: Das System ermittelt für jede Zeile aus Tbuch einen Wahrheitswert. Angenommen, die aktuell betrachtete Zeile aus Tbuch hat Tbuch.Buchnr=5. Dann liefert „SELECT * FROM Tisbn WHERE Tisbn.Buchnr = 5“ die leere Menge, also ist EXISTS falsch. Angenommen, die aktuell betrachtete Zeile aus Tbuch hat Tbuch.Buchnr=8. Dann liefert „SELECT * FROM Tisbn WHERE Tisbn.Buchnr =8“ wenigstens eine Zeile, also ist EXISTS wahr und die Zeile wird angelistet. Der innere SELECT (die query expression) in der obigen EXISTS-Subquery kann nicht getrennt getestet werden. Eine solche Subquery wird oft auch korrelierte Subquery genannt!
- **Beachten Sie bitte, dass die konzeptionelle Beschreibung der IN-Subquery bzw. EXISTS-Subquery nicht notwendigerweise dem physischen Ablauf entspricht. Ein guter Optimizer wählt für diese logisch äquivalenten Varianten immer den gleichen effizienten physischen Zugriff.**
- Aus Performancegründen ist es bei vielen Produkten am Markt sinnvoll, an Stelle von "EXISTS (SELECT * " die Formulierung mit Literal "EXISTS (SELECT 123" zu benutzen. Die Formulierung mit Literal ermöglicht einen Index-Only-Zugriff.
- <subquery>::= (<query expression>) spezifiziert (abgeleitet von einer query expression) einen Skalarwert, eine Zeile oder eine TABLE. Eine Subquery ist - vereinfacht gesagt - eine Select-From-Where-Anfrage die in einer anderen solchen Select-From-Where-Anfrage eingebettet ist. In den Beispielen befindet sich die Subquery in der WHERE-Klausel.

5.15 Subquery mit NOT EXISTS oder NOT IN

Bücher ohne ISBN

```
SELECT
  Tbuch.Buchnr, Tbuch.Titel FROM Tbuch
WHERE Tbuch.Buchnr NOT IN
     (SELECT Tisbn.Buchnr FROM Tisbn
      WHERE Tisbn.Buchnr IS NOT NULL)
;
SELECT
  Tbuch.Buchnr, Tbuch.Titel FROM Tbuch
WHERE NOT EXISTS
     (SELECT * FROM Tisbn
      WHERE Tisbn.Buchnr = Tbuch.Buchnr)
;
SELECT
  Tbuch.Buchnr, Tbuch.Titel
FROM Tbuch LEFT OUTER JOIN Tisbn
     ON Tbuch.Buchnr = Tisbn.Buchnr
WHERE Tisbn.Isbn IS NULL
;
```

Die obigen Select-Anweisungen sind äquivalent, sie liefern für jeden möglichen Datenbestand dasselbe Ergebnis.

Die Wahrheitswerte für **"WHERE NOT EXISTS (...)"** sind

- **wahr** die von der Subquery ermittelte Menge ist leer **(enthält keine Zeile)**
- **falsch** die von der Subquery ermittelte Menge ist nicht leer **(enthält wenigstens eine Zeile)**

Die Wahrheitswerte für **"WHERE ... NOT IN (...)"** sind **wahr, unbekannt** bzw. **falsch**.

*Wenn die **Table Tisbn wenigstens eine Zeile** präsentiert, dann qualifiziert bei folgender SELECT-Anweisung **keine Zeile aus Tbuch** (die Subquery ist nicht-korreliert).*

```
SELECT Buchnr, Titel FROM Tbuch
WHERE NOT EXISTS (SELECT * FROM Tisbn)
;
```

- Tisbn.Isbn ist Primärschlüssel in Tisbn! Bei der Variante LEFT OUTER JOIN wird der Primärschlüssel der "rechten-right" Table Tisbn auf NULL geprüft!
- **Eventuell vorhandene NULLs müssen bei NOT IN besonders berücksichtigt werden!**

  --ein Testfall wird eingefügt

```
insert into tverlag(verlagnr, verlag)
        values     (2222    , 'RORORO')
;
```

  --NOT IN Varianten liefert den Testfall

```
select *
from tverlag
where tverlag.verlagnr NOT IN
        (SELECT tbuch.verlagnr
         FROM tbuch
         WHERE tbuch.verlagnr IS NOT NULL
        )
;
```

  --NOT EXISTS liefert den Testfall

```
select *
from tverlag
where NOT EXISTS
         (SELECT Tbuch.verlagnr
          FROM tbuch
          WHERE tbuch.verlagnr = tverlag.verlagnr)
;
```

  *--Die folgende **nicht-korrelierte NOT IN-Subquery ohne Berücksichtigung der NULL** ist nicht äquivalent und liefert keine Zeile*

```
select *
from tverlag
where tverlag.verlagnr NOT IN
        (SELECT tbuch.verlagnr
         FROM tbuch
        )
;
```

5.16 Die Subquery mit IN etwas genauer betrachtet

5.16.1 Es gibt nicht <>10, alle sind =10

Beispiele 1

--es gibt Praemie=10

```
SELECT
  Tbuch.Buchnr, Tbuch.Titel
FROM Tbuch
WHERE
 Tbuch.Buchnr IN
      (SELECT DISTINCT
         tvautor.buchnr
       FROM  tvautor
       WHERE tvautor.praemie=10.00
      )
;
```

--es gibt nicht Praemie=10

```
SELECT
  Tbuch.Buchnr, Tbuch.Titel
FROM Tbuch
WHERE
 Tbuch.Buchnr NOT IN
      (SELECT DISTINCT
         tvautor.buchnr
       FROM  tvautor
       WHERE tvautor.praemie=10.00
      )
;
```

--es gibt Praemie<>10

```
SELECT
  Tbuch.Buchnr, Tbuch.Titel
FROM Tbuch
WHERE
 Tbuch.Buchnr IN
      (SELECT DISTINCT
         tvautor.buchnr
       FROM  tvautor
       WHERE NOT tvautor.praemie=10.00
      )
;
```

--es gibt nicht Praemie<>10

```
SELECT
  Tbuch.Buchnr, Tbuch.Titel
FROM Tbuch
WHERE
 Tbuch.Buchnr NOT IN
      (SELECT DISTINCT
         tvautor.buchnr
       FROM  tvautor
       WHERE NOT tvautor.praemie=10.00
      )
;
```

5.16.2 Subquery geschachtelt, IN bzw. NOT IN

```
--es gibt ein Buch von 1989
SELECT
      Tautor.Autornr AS Autornr
    , Tautor.Autor   AS Autor
FROM  Tautor
WHERE Tautor.Autornr IN
      (SELECT Tvautor.Autornr FROM Tvautor
       WHERE  Tvautor.Buchnr IN
              (SELECT Tbuch.Buchnr FROM Tbuch
               WHERE  Tbuch.Erschj = 1989
              )
     )
;
--es gibt kein Buch von 1989
SELECT
      Tautor.Autornr AS Autornr
    , Tautor.Autor   AS Autor
FROM  Tautor
WHERE Tautor.Autornr NOT IN
      (SELECT Tvautor.Autornr FROM Tvautor
       WHERE  Tvautor.Buchnr IN
              (SELECT Tbuch.Buchnr FROM Tbuch
               WHERE  Tbuch.Erschj = 1989
              )
     )
;
```

```
--es gibt ein Buch nicht von 1989
SELECT
      Tautor.Autornr AS Autornr
    , Tautor.Autor   AS Autor
FROM  Tautor
WHERE Tautor.Autornr IN
      (SELECT Tvautor.Autornr FROM Tvautor
       WHERE  Tvautor.Buchnr NOT IN
              (SELECT Tbuch.Buchnr FROM Tbuch
               WHERE  Tbuch.Erschj = 1989
              )
     )
;
--es gibt kein Buch nicht von 1989, jedes Buch ist von 1989
SELECT
      Tautor.Autornr AS Autornr
    , Tautor.Autor   AS Autor
FROM  Tautor
WHERE Tautor.Autornr NOT IN
      (SELECT Tvautor.Autornr FROM Tvautor
       WHERE  Tvautor.Buchnr NOT IN
              (SELECT Tbuch.Buchnr FROM Tbuch
               WHERE  Tbuch.Erschj = 1989
              )
     )
;
```

5.16.3 LEFT OUTER JOIN und dann eine Subquery mit IN bzw. NOT IN

```
--alle Daten von Büchern,
--mit wenigstens einer wohldefinierten Praemie
select
 tbuch.buchnr
,tbuch.Titel
,tvautor.Autornr
,tvautor.lfdnr
,tvautor.praemie
,tverlag.Verlagnr
,tverlag.verlag
FROM
     tbuch LEFT OUTER JOIN tvautor
     ON tbuch.buchnr = tvautor.buchnr
           LEFT OUTER JOIN Tverlag
     ON tbuch.Verlagnr = tverlag.verlagnr
WHERE
  tbuch.buchnr IN
     (
      SELECT tvautor.buchnr
      FROM tvautor
      WHERE tvautor.praemie IS NOT NULL
     )
;
```

```
--alle Daten von Büchern,
--ohne wohldefinierte Praemie
select
 tbuch.buchnr
,tbuch.Titel
,tvautor.Autornr
,tvautor.lfdnr
,tvautor.praemie
,tverlag.Verlagnr
,tverlag.verlag
FROM
     tbuch LEFT OUTER JOIN tvautor
     ON tbuch.buchnr = tvautor.buchnr
           LEFT OUTER JOIN Tverlag
     ON tbuch.Verlagnr = tverlag.verlagnr
WHERE
  tbuch.buchnr NOT IN
     (
      SELECT tvautor.buchnr
      FROM tvautor
      WHERE tvautor.praemie IS NOT NULL
     )
 ;
```

5.17 IN, EXISTS, JOIN, Varianten des Codings

5.17.1 IN-Subquery nicht-korreliert

Buchnr, Titel von Büchern mit wenigstens einem Autor der eine Praemie von <50.00 hat

```
Buchnr      Titel
----------- --------------------
8           der Butt
9           DB2 fuer Sie
12          a guide to db2
```

--Variante nicht-korrelierte IN-Subquery:

```
SELECT
Tbuch.Buchnr, Tbuch.Titel
FROM Tbuch
WHERE
     Tbuch.Buchnr IN
          (SELECT DISTINCT Tvautor.Buchnr
           FROM Tvautor
           WHERE 1=1
           AND Tvautor.Praemie <50.00
          )
;
```

--Variante EXISTS-Subquery:

```
SELECT
Tbuch.Buchnr, Tbuch.Titel
FROM Tbuch
WHERE
     EXISTS
          (SELECT DISTINCT Tvautor.Buchnr
           FROM Tvautor
           WHERE 1=1
           AND Tvautor.Praemie <50.00
           AND Tvautor.Buchnr = Tbuch.Buchnr
          )
;
```

--Variante Join und Projektion:

--va1 mit Nested Table Expression

--va1-1

```
SELECT DISTINCT
Tbuch.Buchnr, Tbuch.Titel
FROM
     Tbuch INNER JOIN
     ( SELECT DISTINCT Tvautor.Buchnr
       FROM Tvautor
       WHERE 1=1
       AND Tvautor.Praemie <50.00
     ) zwi1
     ON Tbuch.Buchnr = zwi1.Buchnr
;
```

--va1-2 AND

```
SELECT DISTINCT
Tbuch.Buchnr, Tbuch.Titel
FROM
     Tbuch INNER JOIN
     ( SELECT DISTINCT
              Tvautor.Buchnr as buchnr
             ,tvautor.praemie as praemie
       FROM Tvautor
       WHERE 1=1
     ) zwi1
     ON Tbuch.Buchnr = zwi1.Buchnr
     AND zwi1.Praemie <50.00
;
```

--va1-3 WHERE

```
SELECT DISTINCT
Tbuch.Buchnr, Tbuch.Titel
FROM
     Tbuch INNER JOIN
     ( SELECT DISTINCT
              Tvautor.Buchnr as buchnr
             ,tvautor.praemie as praemie
       FROM Tvautor
       WHERE 1=1
     ) zwi1
     ON Tbuch.Buchnr = zwi1.Buchnr
WHERE zwi1.praemie <50.00
;
```

--va2 ohne Nested Table Expression AND

```
SELECT DISTINCT
Tbuch.Buchnr, Tbuch.Titel
FROM
     Tbuch INNER JOIN Tvautor
     ON Tbuch.Buchnr = Tvautor.Buchnr
     AND Tvautor.Praemie <50.00
;
```

--va3 ohne Nested Table Expression WHERE

```
SELECT DISTINCT
Tbuch.Buchnr, Tbuch.Titel
FROM
     Tbuch INNER JOIN Tvautor
     ON Tbuch.Buchnr = Tvautor.Buchnr
WHERE Tvautor.Praemie <50.00
;
```

5.17.2 IN-Subquery korreliert

Buchnr, Titel von Büchern mit wenigstens einem Autor mit Praemie < Preis

```
Buchnr      Titel
----------- --------------
9           DB2 fuer Sie
```

--Variante korrelierte IN-Subquery:

```
SELECT
Tbuch.Buchnr, Tbuch.Titel
FROM Tbuch
WHERE
     Tbuch.Buchnr IN
          (SELECT DISTINCT Tvautor.Buchnr
           FROM Tvautor
           WHERE 1=1
           AND Tvautor.Praemie < Tbuch.Preis
          )
;
```

--Variante EXISTS-Subquery:

```
SELECT
Tbuch.Buchnr, Tbuch.Titel
FROM Tbuch
WHERE
     EXISTS
          (SELECT DISTINCT Tvautor.Buchnr
           FROM Tvautor
           WHERE 1=1
           AND Tvautor.Praemie < Tbuch.Preis
           AND Tvautor.Buchnr  = Tbuch.Buchnr
          )
;
```

--Variante Join und Projektion:
--va1 mit Nested Table Expression

```
SELECT DISTINCT
Tbuch.Buchnr, Tbuch.Titel
FROM
     Tbuch INNER JOIN
     ( SELECT DISTINCT
        Tvautor.Buchnr  AS Buchnr
       ,Tvautor.Praemie AS Praemie
       FROM Tvautor
       WHERE 1=1
       AND tvautor.Praemie < Tbuch.Preis
     ) zwi1
     ON Tbuch.Buchnr = zwi1.Buchnr
;
SQL Server Meldung 4104, Ebene 16, Status 1, Zeile 10
Der mehrteilige Bezeichner 'Tbuch.Preis' konnte nicht gebunden
werden.
DB2 SQL0204N "TBUCH.PREIS" ist ein nicht definierter Name.
SQLSTATE=42704
Oracle ORA-00904: "TBUCH"."PREIS": ungültiger Bezeichner
```

```
--va1-2 AND
SELECT DISTINCT
Tbuch.Buchnr, Tbuch.Titel
FROM
     Tbuch INNER JOIN
     ( SELECT DISTINCT
        Tvautor.Buchnr  AS Buchnr
       ,Tvautor.Praemie AS Praemie
       FROM Tvautor
     ) zwi1
     ON Tbuch.Buchnr = zwi1.Buchnr
     AND Zwi1.Praemie < Tbuch.Preis
;
--va1-3 WHERE
SELECT DISTINCT
Tbuch.Buchnr, Tbuch.Titel
FROM
     Tbuch INNER JOIN
     ( SELECT DISTINCT
        Tvautor.Buchnr  AS Buchnr
       ,Tvautor.Praemie AS Praemie
       FROM Tvautor
     ) zwi1
     ON Tbuch.Buchnr = zwi1.Buchnr
WHERE Zwi1.Praemie < Tbuch.Preis
;
--va2 ohne Nested Table Expression AND
SELECT DISTINCT
Tbuch.Buchnr, Tbuch.Titel
FROM
     Tbuch INNER JOIN Tvautor
     ON Tbuch.Buchnr = Tvautor.Buchnr
     AND Tvautor.Praemie < Tbuch.Preis
;
--va3 ohne Nested Table Expression WHERE
SELECT DISTINCT
Tbuch.Buchnr, Tbuch.Titel
FROM
     Tbuch INNER JOIN Tvautor
     ON Tbuch.Buchnr = Tvautor.Buchnr
WHERE Tvautor.Praemie < Tbuch.Preis
;
```

5.17.3 Join einer Table mit sich selbst, IN (=10) AND IN (=498)

Buchnr, Titel von Büchern mit wenigstens einer Praemie =10.00 und wenigstens einer Praemie = 498.00

```
Buchnr      Titel
----------- ------------------
9           DB2 fuer Sie
```

--Variante nicht-korrelierte IN-Subquery:

```
SELECT
Tbuch.Buchnr, Tbuch.Titel
FROM Tbuch
WHERE
     Tbuch.Buchnr IN
          (SELECT DISTINCT tab1.Buchnr
           FROM Tvautor tab1
           WHERE 1=1
           AND tab1.Praemie =10.00
          )
AND
     Tbuch.Buchnr IN
          (SELECT DISTINCT tab2.Buchnr
           FROM Tvautor tab2
           WHERE 1=1
           AND tab2.Praemie =498.00
          )
;
```

--Variante EXISTS-Subquery:

```
SELECT
Tbuch.Buchnr, Tbuch.Titel
FROM Tbuch
WHERE
     EXISTS
          (SELECT DISTINCT tab1.Buchnr
           FROM Tvautor tab1
           WHERE 1=1
           AND tab1.Praemie =10.00
           AND tab1.Buchnr = Tbuch.Buchnr
          )
AND
     EXISTS
          (SELECT DISTINCT tab2.Buchnr
           FROM Tvautor tab2
           WHERE 1=1
           AND tab2.Praemie =498.00
           AND tab2.Buchnr = Tbuch.Buchnr
          )

;
```

--Variante Join und Projektion:

--va1 mit Nested Table Expression

--va1-1

```
SELECT DISTINCT
Tbuch.Buchnr, Tbuch.Titel
FROM
     Tbuch INNER JOIN
     ( SELECT DISTINCT Tvautor.Buchnr
       FROM Tvautor
       WHERE 1=1
       AND Tvautor.Praemie =10.00
     ) zwi1
     ON Tbuch.Buchnr = zwi1.Buchnr
           INNER JOIN
     ( SELECT DISTINCT Tvautor.Buchnr
       FROM Tvautor
       WHERE 1=1
       AND Tvautor.Praemie =498.00
     ) zwi2
     ON Tbuch.Buchnr = zwi2.Buchnr
;
```

--va3 ohne Nested Table Expression WHERE

```
SELECT DISTINCT
Tbuch.Buchnr, Tbuch.Titel
FROM
     Tbuch INNER JOIN Tvautor tab1
     ON Tbuch.Buchnr = tab1.Buchnr
           INNER JOIN Tvautor tab2
     ON Tbuch.Buchnr = tab2.Buchnr
WHERE
       tab1.Praemie =10.00
AND
       tab2.Praemie =498.00
;
```

Und wenn der Titel nicht mit angelistet werden muss können die SELECT-Anweisungen vereinfacht werden (ein Zugriff auf Tbuch ist nicht mehr nötig).

Buchnr von Büchern mit wenigstens einer Praemie =10.00 und wenigstens einer Praemie = 498.00

```
Buchnr
-----------
9
```

--Variante nicht-korrelierte IN-Subquery:

```
            SELECT DISTINCT tab1.Buchnr
            FROM Tvautor tab1
            WHERE 1=1
            AND tab1.Praemie =10.00
AND
     tab1.Buchnr IN
          (SELECT DISTINCT tab2.Buchnr
           FROM Tvautor tab2
           WHERE 1=1
           AND tab2.Praemie =498.00
          )
;
```

--Variante EXISTS-Subquery:

```
            SELECT DISTINCT tab1.Buchnr
            FROM Tvautor tab1
            WHERE 1=1
            AND tab1.Praemie =10.00
AND
     EXISTS
          (SELECT DISTINCT tab2.Buchnr
           FROM Tvautor tab2
           WHERE 1=1
           AND tab2.Praemie =498.00
           AND tab2.Buchnr = tab1.Buchnr
          )

;
```

--Variante Join und Projektion:

--va1 mit Nested Table Expression

--va1-1

```
SELECT DISTINCT
zwi1.Buchnr
FROM
     ( SELECT DISTINCT Tvautor.Buchnr
       FROM Tvautor
       WHERE 1=1
       AND Tvautor.Praemie =10.00
     ) zwi1
           INNER JOIN
     ( SELECT DISTINCT Tvautor.Buchnr
       FROM Tvautor
       WHERE 1=1
       AND Tvautor.Praemie =498.00
     ) zwi2
     ON zwi1.buchnr = zwi2.buchnr
;
```

--va3 ohne Nested Table Expression WHERE

```
SELECT DISTINCT
tab1.Buchnr
FROM
     Tvautor tab1
          INNER JOIN
     Tvautor tab2
     ON tab1.Buchnr = tab2.Buchnr
WHERE
       tab1.Praemie =10.00
AND
       tab2.Praemie =498.00
;
```

5.17.4 Subquery geschachtelt

Autoren mit einem Buch im Jahr 1989 (mindestens ein Buch des Autors ist 1989 erschienen).

```
--IN-Subquery nicht-korreliert
SELECT
      Tautor.Autornr AS Autornr
    , Tautor.Autor   AS Autor
FROM  Tautor
WHERE Tautor.Autornr IN
      (SELECT Tvautor.Autornr FROM Tvautor
       WHERE  Tvautor.Buchnr IN
              (SELECT Tbuch.Buchnr FROM Tbuch
               WHERE  Tbuch.Erschj = 1989
              )
     )
;
--EXISTS-Subquery korreliert
SELECT
      Tautor.Autornr AS Autornr
    , Tautor.Autor   AS Autor
FROM   Tautor
WHERE  EXISTS
       ( SELECT * FROM Tvautor
         WHERE Tvautor.Autornr = Tautor.Autornr
         AND   EXISTS
               ( SELECT * FROM Tbuch
                 WHERE Tbuch.Buchnr = Tvautor.Buchnr
                 AND   Tbuch.Erschj = 1989
               )
          )
;
```

Äquivalente SELECT-Anweisungen mit JOIN und Projektion.

```
SELECT DISTINCT
      Tautor.Autornr AS Autornr
    , Tautor.Autor   AS Autor
FROM   Tautor INNER JOIN Tvautor
       ON Tautor.Autornr = Tvautor.Autornr
              INNER JOIN
                         (select tbuch.buchnr
                          FROM tbuch
                          WHERE tbuch.erschj = 1989
                          )
                         zwi
       ON Tvautor.Buchnr = zwi.Buchnr
;
SELECT DISTINCT
      Tautor.Autornr AS Autornr
    , Tautor.Autor   AS Autor
FROM   Tautor INNER JOIN Tvautor
       ON Tautor.Autornr = Tvautor.Autornr
              INNER JOIN Tbuch
       ON Tvautor.Buchnr = Tbuch.Buchnr
       AND Tbuch.Erschj = 1989
;
SELECT DISTINCT
      Tautor.Autornr AS Autornr
    , Tautor.Autor   AS Autor
FROM   Tautor INNER JOIN Tvautor
       ON Tautor.Autornr = Tvautor.Autornr
              INNER JOIN Tbuch
       ON Tvautor.Buchnr = Tbuch.Buchnr
WHERE Tbuch.Erschj = 1989
;
SELECT DISTINCT
      Tautor.Autornr AS Autornr
    , Tautor.Autor   AS Autor
FROM   Tautor, Tvautor, Tbuch
WHERE  Tautor.Autornr = Tvautor.Autornr
AND    Tvautor.Buchnr = Tbuch.Buchnr
AND    Tbuch.Erschj = 1989
;
```

5.17.5 Subquery geschachtelt, Join einer Table mit sich selbst

Autornr von Autoren die wenigstens ein Buch geschrieben haben und zusammen mit einem Autor von 'a guide to db2' bzw. ‚A GUIDE TO DB2‘ bzw. ‚A Guide To DB2‘ ein Buch geschrieben haben.

Tvautor

Buchnr	Autornr	Lfdnr
12	20	1
12	21	2
77	**21**	**1**
77	**31**	**1**
88	**20**	**1**
88	**41**	**1**

‚a guide to db2‘ >> Buchnr 12 >> Autornr 20 21 >> Buchnr 12 77 88 >> Autornr?

Analysieren und testen Sie die Select-Anweisungen, erstellen Sie dazu geeignete Testdaten.

```
SELECT DISTINCT
  aa.Autornr
FROM Tvautor aa
WHERE aa.Buchnr IN
       (SELECT bb.Buchnr FROM Tvautor bb
        WHERE bb.Autornr IN
              (SELECT cc.Autornr FROM Tvautor cc
               WHERE cc.Buchnr IN
                     (SELECT dd.Buchnr FROM Tbuch dd
                      WHERE LOWER(dd.Titel)
                        = 'a guide to db2'
       )      )      )
;
```

Schreiben Sie eine äquivalente SELECT-Anweisung mit Hilfe der Join-Operation und der Projektion.

Die Lösung finden Sie auf der folgenden Seite.

```
SELECT DISTINCT
  aa.Autornr
FROM
(
  Tvautor aa JOIN Tvautor bb
  ON aa.Buchnr = bb.Buchnr
             JOIN Tvautor cc
  ON bb.Autornr = cc.Autornr
             JOIN Tbuch dd
  ON cc.Buchnr = dd.Buchnr
)
WHERE UPPER(dd.Titel) = 'A GUIDE TO DB2'
;
SELECT DISTINCT
  aa.Autornr
FROM
(
  Tvautor aa JOIN Tvautor bb
  ON aa.Buchnr = bb.Buchnr
             JOIN Tvautor cc
  ON bb.Autornr = cc.Autornr
             JOIN Tbuch dd
  ON cc.Buchnr = dd.Buchnr
  AND UPPER(dd.Titel) = 'A GUIDE TO DB2'
)
;
SELECT DISTINCT
  aa.Autornr
FROM
(
  Tvautor aa JOIN Tvautor bb
  ON aa.Buchnr = bb.Buchnr
             JOIN Tvautor cc
  ON bb.Autornr = cc.Autornr
             JOIN
             (select tbuch.buchnr
              from tbuch
              WHERE LOWER(tbuch.Titel)
              = 'a guide to db2'
             )
             dd
  ON cc.Buchnr = dd.Buchnr
)
;
```

5.18 Ein guter Optimizer liefert …

Beispiel1 Subquery noncorrelated:

```
SELECT
     abc.Buchnr
   , abc.Titel
FROM    Tbuch abc
WHERE   abc.Erschj = 1988
  AND   abc.Preis >=
        (
         SELECT  AVG(xyz.Preis)
         FROM    Tbuch xyz
         WHERE   xyz.Erschj = 1988
        )
;
```

Buchnr, Titel von Büchern des Jahres 1988 mit gleichem oder grösserem Preis als der Durchschnittpreis aller Bücher von 1988.

Bei einem Vergleichsoperator wie <= kann der Vergleich nur dann erfolgreich ausgeführt werden, wenn die Subquery genau einen Wert ermittelt.

Beispiel2a Subquery correlated:

```
SELECT
     abc.Buchnr
   , abc.Titel
FROM    Tbuch abc
WHERE   -------------
        abc.Preis >=
        (
         SELECT  AVG(XYZ.Preis)
         FROM    Tbuch XYZ
         WHERE   XYZ.Erschj = abc.Erschj
        )
;
```

Buchnr, Titel von Büchern mit gleichem oder grösserem Preis als der Durchschnittpreis aller Bücher ihres Erscheinungsjahres.

Angenommen, das Erscheinungsjahr ist nicht definiert, dann werden die Zeilen nicht angelistet (vgl. Buchnr = 27).

Angenommen, der Durchschnitt für ein Erscheinungsjahr ist nicht definiert (AVG(Preis) ergibt NULL), dann werden die Zeilen dieses Jahres nicht angelistet.

Beispiel2b äquivalent mit Beispiel2a:

```
SELECT
     ABC.Buchnr, ABC.Titel
FROM Tbuch ABC INNER JOIN
     (SELECT Erschj     AS Erschj
           , AVG(Preis) AS Avgpreis
      FROM   Tbuch
      GROUP  BY Erschj
     ) xxx
     ON ABC.Erschj = xxx.Erschj
WHERE
     ABC.Preis >= xxx.Avgpreis
;
```

Ein guter Optimizer liefert die Daten bei äquivalenten Anweisungen gleich schnell!

5.19 SQL und der gesunde Menschenverstand, >ALL bzw. >MAX

Bücher, die teurer sind als alle Bücher des Jahres 2000.

- Teurer als alle Bücher von 2000.
- Teurer als das teuerste Buch von 2000.

Die folgenden Select-Anweisungen liefern 11 Zeilen bzw. 0 Zeilen bzw. bzw. 11 Zeilen!

Preis > ALL ist wahr für jede Zeile von Tbuch (auch die Bücher mit Preis NULL!) sofern der innere SELECT die leere Menge ergibt!

```
SELECT * FROM Tbuch
WHERE  Tbuch.Preis > ALL
     (SELECT xyz.Preis
      FROM Tbuch xyz
      WHERE  xyz.Erschj = 2000
     )
;
```

Bei der Formulierung mit MAX qualifiziert aber, sofern kein Buch mit Erschj=2000 vorhanden ist, keine Zeile!

```
SELECT * FROM Tbuch
WHERE  Tbuch.Preis >
     (SELECT MAX(xyz.Preis)
      FROM Tbuch xyz
      WHERE  xyz.Erschj = 2000
     )
;
```

Die NOT EXISTS-Variante liefert alle Zeilen aus Tbuch, sofern kein Buch mit Erschj=2000 vorhanden ist!

```
SELECT * FROM Tbuch
WHERE  NOT EXISTS
      (SELECT xyz.preis
       FROM Tbuch xyz
       WHERE xyz.Erschj = 2000
       AND NOT (Tbuch.Preis > xyz.preis  )
      )
;
```

Bücher, die teurer sind als alle Bücher des Jahres 2000.

- Es gibt kein Buch von 2000 das nicht billiger ist.

- **Preis > ALL ist wahr, sofern der innere SELECT die leere Menge ergibt!**

Die COALESCE-Variante liefert, sofern kein Buch mit Erschj=2000 vorhanden ist, nur die Bücher mit wohldefiniertem Preis!

```
SELECT * FROM Tbuch
WHERE  CAST (Tbuch.Preis AS DECIMAL(8,2)) >
      (SELECT
       COALESCE( MAX(xyz.Preis), -99999.99 -0.01 )
       FROM Tbuch xyz
       WHERE xyz.Erschj = 2000
      )
;
```

Anmerkung zur COALESCE-Variante: Sofern der Preis eines Buches vom Datentyp DECIMAL(7,2) den Wert -99999.99 annehmen kann muss natürlich auch dieser kleinste mögliche Wert des Datentyps berücksichtigt werden. Bücher mit NULL-Preis werden aber nicht angelistet.

5.20 UNION, INTERSECT, EXCEPT

5.20.1 UNION und LEFT OUTER JOIN

```
  SELECT Tbuch.Buchnr    AS Buchnr
        ,Tbuch.Erschj    AS Erschj
        ,Tisbn.ISBN      AS Isbn
  FROM   Tbuch, Tisbn
  WHERE  Tbuch.Buchnr = Tisbn.Buchnr
UNION
  SELECT Tbuch.Buchnr    AS Buchnr
        ,Tbuch.Erschj    AS Erschj
        ,'nicht da'      AS Isbn
  FROM   Tbuch
  WHERE  NOT EXISTS
    (
     SELECT *
     FROM Tisbn
     WHERE Buchnr = Tbuch.Buchnr
    )
ORDER BY Buchnr, isbn
;
```

Buchnr	Erschj	Isbn
1	NULL	nicht da
2	NULL	nicht da
5	1988	nicht da
6	1988	nicht da
7	1989	nicht da
8	*1977*	*3472864303*
8	*1977*	*34728643yx*
9	1990	nicht da
11	1990	nicht da
12	*1989*	*0201501139*
18	1989	nicht da
27	NULL	nicht da

- Vereinigung der beiden Ergebnismengen. Die Ergebnismengen müssen gleich viele Spalten aufweisen, die Spalten müssen zusammenpassen (Datentyp), bei numerischen Spalten findet wenn nötig eine Konversion statt, bei Charakterspalten ist die längere entscheidend (die kürzere Spalte wird mit Blanks verlängert).
- Anlisten von Konstanten ist möglich.
- Bei UNION ALL werden Duplikate nicht entfernt. Duplikate werden entfernt, sofern UNION ohne die Option ALL verwendet wird (implizites DISTINCT).
- Wenn Sie wissen, dass keine Duplikate auftreten können (wie in obigem Beispiel), dann können Sie UNION ALL kodieren, Sie vermeiden dadurch einen unnötigen Sort.
- Vergleichen Sie auch die äquivalente Anweisung mit LEFT OUTER JOIN und COALESCE bzw. CASE.

```
SELECT Tbuch.Buchnr                       AS Buchnr
     , Tbuch.Erschj                       AS Erschj
     , COALESCE(Tisbn.ISBN, 'nicht da') AS Isbn
FROM   Tbuch LEFT OUTER JOIN Tisbn
       ON  Tbuch.Buchnr = Tisbn.Buchnr
ORDER BY Buchnr, isbn
;
SELECT Tbuch.Buchnr                       AS Buchnr
     , Tbuch.Erschj                       AS Erschj
     , CASE
            WHEN tisbn.isbn IS NOT NULL
                 THEN tisbn.isbn
            ELSE
                 'nicht da'
       END                                AS Isbn
FROM   Tbuch LEFT OUTER JOIN Tisbn
       ON  Tbuch.Buchnr = Tisbn.Buchnr
ORDER BY Buchnr, isbn
;
```

5.20.2 UNION OR, INTERSECT AND, EXCEPT NOT

OR / Vereinigung / UNION / FULL OUTER JOIN

```
  select tvautor.buchnr as buchnr
  from tvautor
UNION
  select tisbn.buchnr   as buchnr
  from tisbn
;
SELECT DISTINCT
  COALESCE (tvautor.buchnr, tisbn.buchnr)  as buchnr
FROM Tvautor FULL OUTER JOIN tisbn
     ON tvautor.buchnr = Tisbn.Buchnr
;
```

AND / Durchschnitt / INTERSECT / INNER JOIN

```
  select tvautor.buchnr as buchnr
  from tvautor
INTERSECT
  select tisbn.buchnr   as buchnr
  from tisbn
;
SELECT DISTINCT
  tvautor.buchnr        as buchnr
FROM Tvautor INNER JOIN tisbn
     ON tvautor.buchnr = Tisbn.Buchnr
;
```

NOT / Differenz / EXCEPT / LEFT OUTER JOIN …WHERE

```
  select tvautor.buchnr as buchnr
  from tvautor
EXCEPT
  select tisbn.buchnr   as buchnr
  from tisbn
;
SELECT DISTINCT
    tvautor.buchnr      as buchnr
FROM Tvautor LEFT OUTER JOIN tisbn
     ON tvautor.buchnr = Tisbn.Buchnr
WHERE tisbn.isbn IS NULL
;
```

SQL Server:

INTERSECT und EXCEPT werden unterstützt.

DB2 for z/OS Version 9:

INTERSECT und EXCEPT werden unterstützt.

Oracle:

Statt EXCEPT muss MINUS codiert werden.

Folgende Schreibweise ist natürlich auch möglich:

```
SELECT vereinigung.buchnr as buchnr
FROM
(
  select tvautor.buchnr  as buchnr
  from tvautor
UNION
  select tisbn.buchnr    as buchnr
  from tisbn
) AS vereinigung

SELECT schnitt.buchnr as buchnr
FROM
(
  select tvautor.buchnr  as buchnr
  from tvautor
INTERSECT
  select tisbn.buchnr    as buchnr
  from tisbn
) AS schnitt

SELECT diff.buchnr as buchnr
FROM
(
  select tvautor.buchnr  as buchnr
  from tvautor
EXCEPT
  select tisbn.buchnr    as buchnr
  from tisbn
) AS diff
```

5.20.3 UNION INTERSECT EXCEPT und die NULL und Alternativen

```
tab1
pk          s1          s2
----------- ----------- -----------
1           11          11
2           22          NULL
3           33          33
4           44          NULL
tab2
pk          s1          s2
----------- ----------- -----------
3           33          33
4           44          NULL
5           55          55
6           66          NULL
  select pk, s1, s2
  from tab1
UNION
  select pk, s1, s2
  from tab2
;
pk          s1          s2
----------- ----------- -----------
1           11          11
2           22          NULL
3           33          33
4           44          NULL
5           55          55
6           66          NULL
  select pk, s1, s2
  from tab1
INTERSECT
  select pk, s1, s2
  from tab2
;
pk          s1          s2
----------- ----------- -----------
3           33          33
4           44          NULL

  select pk, s1, s2
  from tab1
EXCEPT
  select pk, s1, s2
  from tab2
;
pk          s1          s2
----------- ----------- -----------
1           11          11
2           22          NULL
```

UNION Alternative mit FULL OUTER JOIN

```
select --DISTINCT
  COALESCE  (tab1.pk, tab2.pk)  AS pk
 ,COALESCE  (tab1.s1, tab2.s1)  AS s1
 ,COALESCE  (tab1.s2, tab2.s2)  AS s2
FROM
   tab1 FULL OUTER JOIN tab2
   ON tab1.pk=tab2.pk
```

Die Alternativen zu INTERSECT und EXCEPT finden Sie auf den folgenden Seiten.

Beachten Sie bitte die Definition von tab1 bzw. tab2.

*Die **Spalte s1** kann **keine NULL** präsentieren.*

*Die **Spalte s2** kann **NULL** präsentieren.*

```
drop table tab1;
drop table tab2;

create table tab1
(pk integer not null
,s1 integer not null
,s2 integer
,primary key (pk));
create table tab2
(pk integer not null
,s1 integer not null
,s2 integer
,primary key (pk));

insert into tab1 (pk, s1, s2 ) values ( 1 , 11, 11);
insert into tab1 (pk, s1, s2 ) values ( 2 , 22, NULL);
insert into tab1 (pk, s1, s2 ) values ( 3 , 33, 33 );
insert into tab1 (pk, s1, s2 ) values ( 4 , 44, NULL);

insert into tab2 (pk, s1, s2 ) values ( 3 , 33, 33 );
insert into tab2 (pk, s1, s2 ) values ( 4 , 44, NULL);
insert into tab2 (pk, s1, s2 ) values ( 5 , 55, 55 );
insert into tab2 (pk, s1, s2 ) values ( 6 , 66, NULL);
```

INTERSECT Alternative 1 mit EXISTS-Subquery

```
SELECT DISTINCT
       tab1.pk AS pk
     , tab1.s1 AS s1
     , tab1.s2 AS s2
from tab1
WHERE
EXISTS
   (select * from tab2
    WHERE
          tab2.pk = tab1.pk
    AND   tab2.s1 = tab1.s1
    AND   tab2.s2 = tab1.s2
   )
OR
   (
    tab1.s2 IS NULL
    AND EXISTS
     (select * from tab2
      WHERE
            tab2.pk = tab1.pk
      AND   tab2.s1 = tab1.s1
      AND   tab2.s2 IS NULL
    )
   )
```

INTERSECT Alternative 2 mit INNER JOIN

```
SELECT DISTINCT
       tab1.pk AS pk
     , tab1.s1 AS s1
     , tab1.s2 AS s2
from tab1 INNER JOIN tab2
ON
  (      tab1.pk = tab2.pk
   AND   tab1.s1 = tab2.s1
   AND   tab1.s2 = tab2.s2
  )
  OR
  (      tab1.pk = tab2.pk
   AND   tab1.s1 = tab2.s1
   AND   tab1.s2 IS NULL
   AND   tab2.s2 IS NULL
  )
```

EXCEPT Alternative 1 mit NOT EXISTS-Subquery

```
SELECT DISTINCT
       tab1.pk AS pk
     , tab1.s1 AS s1
     , tab1.s2 AS s2
from tab1
WHERE NOT
(
EXISTS
   (select * from tab2
    WHERE
          tab2.pk = tab1.pk
    AND   tab2.s1 = tab1.s1
    AND   tab2.s2 = tab1.s2
   )
OR
   (
    tab1.s2 IS NULL
    AND EXISTS
     (select * from tab2
      WHERE
            tab2.pk = tab1.pk
      AND   tab2.s1 = tab1.s1
      AND   tab2.s2 IS NULL
    )
   )
)
```

EXCEPT Alternative 2 mit LEFT OUTER JOIN

```
SELECT DISTINCT
       tab1.pk AS pk
     , tab1.s1 AS s1
     , tab1.s2 AS s2
from tab1 LEFT OUTER JOIN tab2
ON
  (      tab1.pk = tab2.pk
   AND   tab1.s1 = tab2.s1
   AND   tab1.s2 = tab2.s2
  )
  OR
  (      tab1.pk = tab2.pk
   AND   tab1.s1 = tab2.s1
   AND   tab1.s2 IS NULL
   AND   tab2.s2 IS NULL
  )
WHERE tab2.pk IS NULL
```

5.21 Common Table Expression

5.21.1 Common Table Expression und ein Inner Join

```
WITH
T1 AS
 (
   select
         tbuch.buchnr AS buchnr
        ,tbuch.preis  AS preis
   from tbuch
   where tbuch.preis is not null
 )
,
T2 AS
  (
   select
          tisbn.buchnr as buchnr
         ,Tisbn.Isbn   as isbn
   from Tisbn
   where tisbn.lfdnr = 1
  )
----
SELECT
      T1.buchnr as buchnr
     ,T1.preis  as preis
     ,T2.isbn   as isbn
FROM T1 INNER JOIN T2
     ON T1.buchnr = t2.buchnr
;
```

5.21.2 Common Table Expression und ein Left Outer Join

```
WITH
Txyz AS
 (
   select
    Tbuch.Buchnr                    AS Buchnr
   ,Tisbn.Lfdnr                     AS Lfdnri
   ,Tisbn.Isbn                      AS Isbn
   from tbuch INNER JOIN Tisbn
        ON Tbuch.Buchnr = Tisbn.buchnr
   where tbuch.erschj IN (1977, 1988)
 )
SELECT
 Tbuch.Buchnr                     AS Buchnr
,Txyz.Lfdnri                      AS Lfdnri
,Txyz.Isbn                       AS Isbn
,Tbuch.Erschj                     AS Erschj
FROM Tbuch LEFT OUTER JOIN Txyz
     ON Tbuch.Buchnr = Txyz.buchnr
;
--------logisch äquivalent
SELECT
 Tbuch.Buchnr                     AS Buchnr
,Txyz.Lfdnri                      AS Lfdnri
,Txyz.Isbn                       AS Isbn
,Tbuch.Erschj                     AS Erschj
FROM Tbuch LEFT OUTER JOIN
 (
   select
    Tbuch.Buchnr                    AS Buchnr
   ,Tisbn.Lfdnr                     AS Lfdnri
   ,Tisbn.Isbn                      AS Isbn
   from tbuch INNER JOIN Tisbn
        ON Tbuch.Buchnr = Tisbn.buchnr
   where tbuch.erschj IN (1977, 1988)
 )
Txyz
     ON Tbuch.Buchnr = Txyz.buchnr
;
--------logisch äquivalent
SELECT
 Tbuch.Buchnr                     AS Buchnr
,Tisbn.Lfdnr                      AS Lfdnri
,Tisbn.Isbn                       AS Isbn
,Tbuch.Erschj                     AS Erschj
FROM
     TBUCH LEFT OUTER JOIN TISBN
     ON TBUCH.Buchnr = TISBN.Buchnr
     AND tbuch.erschj IN (1977, 1988)
;
```

„Der Left Outer Natural-Join, AND Bedingung links“

Erinnern Sie sich noch an das Unterkapitel?

5.21.3 Common Table Expression und Redundanz im Coding

```
Tvautor
Buchnr      Autornr     Lfdnr Praemie
----------- ----------- ----- -----------
1           100         1     NULL
1           200         2     NULL
2           200         0     NULL
5           1           0     NULL
6           2           0     NULL
7           3           0     NULL
8           2           0     20.000
9           10          1     10.000
9           11          2     498.000
12          20          1     30.000
12          21          2     NULL
```

Hole Buchnr, Autornr, Praemie aller Bücher mit wohldefinierter Praemie, aber das Buch soll nur eine wohldefinierte Praemie haben.

```
BUCHNR      AUTORNR     PRAEMIE
----------- ----------- -----------
8           2           20,000
12          20          30,000
```

```
SELECT
 T1.buchnr, t1.autornr, t1.praemie
FROM tvautor t1
WHERE 1=1
AND t1. praemie IS NOT NULL
--AND b2 AND b3 AND b4
AND
t1.buchnr IN
    (SELECT t2.buchnr
     FROM tvautor t2
     WHERE 1=1
     AND t2. praemie IS NOT NULL
     --AND b2 AND b3 AND b4
     GROUP BY t2.buchnr
     HAVING COUNT(t2.praemie) = 1
    )
;
```

Problem: Die Bedingung

```
t1. praemie IS NOT NULL
--AND b2 AND b3 AND b4
```

muss für die gegebene fachliche Anforderung auch in der Subquery codiert werden!

```
WITH t12
AS
(SELECT
      Tvautor.buchnr  as buchnr
     ,tvautor.autornr as autornr
     ,tvautor.praemie as praemie
 FROM tvautor
 WHERE 1=1
 AND tvautor.praemie IS NOT NULL
 --AND b2 AND b3 AND b4
)
SELECT
 Tx.buchnr, tx.autornr, tx.praemie
FROM t12 tx
WHERE tx.buchnr
IN
    (
    SELECT ty.buchnr
    FROM t12 ty
    GROUP BY ty.buchnr
    HAVING COUNT(ty.praemie) = 1
)
;
BUCHNR      AUTORNR     PRAEMIE
----------- ----------- -----------
8           2           20,000
12          20          30,000
```

Vermeidung des Problems:

Mit Hilfe einer common-table-expression kann diese Redundanz vermieden werden, die SELECT-Anweisung ist jetzt wartungsfreundlicher und kann leichter erweitert werden!

5.22 Division

```
TABLE tmitnrfnr
mitnr           fnr
--------------- ---------------
Emil            Auto
Emil            Fahrrad
Franz           Fahrrad
Franz           Unimog
Fritz           Auto
Fritz           Motorrad
Max             Unimog
Otto            Auto
Otto            Fahrrad
Otto            Motorrad

TABLE tfahrzeug
fnr             farbe
--------------- ---------------
Auto            rot
Fahrrad         rot
Motorrad        lila
Unimog          grün
```

Emil kann wenigstens ein Fahrzeug und **alle roten Fahrzeuge** des Fuhrparks fahren.

Es gibt kein rotes Fahrzeug im Fuhrpark, das Emil **nicht** fahren kann!

Franz kann wenigstens ein Fahrzeug und **nicht alle roten Fahrzeuge** des Fuhrparks fahren.

Es gibt ein rotes Fahrzeug im Fuhrpark, das Franz **nicht** fahren kann!

Anforderung1 Division „wenigstens eins und alle roten“

Wer kann wenigstens ein Fahrzeug fahren und alle roten Fahrzeuge des Fuhrparks und auch vielleicht noch andere?

Sofern es keine roten Fahrzeuge gibt werden alle Fahrer, die wenigstens ein Fahrzeug fahren, angelistet.

Pseudocode:

„tmitnrfnr{mitnr,fnr} DIVIDED BY Tfahrzeug{fnr WHERE farbe=rot}“

```
mitnr
---------------
Emil
Otto
```

```
--Anforderung1 Division „wenigstens eins und alle roten“
--SQL-Implementierung mit INNER JOIN und GROUP BY und UNION
--performant, aber die Logik ist „nicht schön“
SELECT
tabab.mitnr as mitnr
FROM tmitnrfnr tabab INNER JOIN Tfahrzeug tabb
     ON tabab.fnr = tabb.fnr
     AND tabb.farbe = 'rot'
GROUP BY  tabab.mitnr
HAVING COUNT(DISTINCT tabb.fnr)
     = (
         SELECT COUNT(DISTINCT fnr)
         FROM Tfahrzeug
          WHERE  farbe = 'rot'
       )
UNION
select DISTINCT
tabab.mitnr
from tmitnrfnr tabab
where not exists
      (
       select *
       from tfahrzeug tabb
       where tabb.farbe = 'rot'
      )
;

--Anforderung1 Division „wenigstens eins und alle roten“
--SQL-Implementierung mit NOT EXISTS
--nicht performant, aber die Logik ist “schön”
SELECT DISTINCT
tabab.mitnr
FROM tmitnrfnr tabab
WHERE NOT EXISTS
      (
      SELECT * FROM Tfahrzeug tabb
       WHERE tabb.farbe = 'rot'
       AND NOT EXISTS
             (
            SELECT * FROM tmitnrfnr tabxy
             WHERE tabxy.mitnr   = tabab.mitnr
             AND   tabxy.fnr     = tabb.fnr)
             )
;
```

--Anforderung1 Division „wenigstens eins und alle roten“

--SQL-Implementierung mit Common Table Expression

--und INNER JOIN und GROUP BY und UNION

```
WITH
dividend AS
(
 select DISTINCT
   tmitnrfnr.mitnr as mitnr
 from tmitnrfnr
)
,
divisor AS
(
select DISTINCT
   tfahrzeug.fnr as fnr
 from tfahrzeug
where farbe = 'rot'
)
,
mediator AS
(
select DISTINCT
   tmitnrfnr.mitnr as mitnr
 , tmitnrfnr.fnr   as fnr
 From tmitnrfnr
)
SELECT
  mediator.mitnr as mitnr
FROM mediator INNER JOIN divisor
     ON mediator.Fnr = divisor.Fnr
GROUP BY mediator.mitnr
HAVING COUNT(divisor.fnr)
=
(
 SELECT COUNT(divisor.fnr)
 FROM divisor
)
UNION
select
dividend.mitnr as mitnr
from dividend
where not exists
      (select *
       from divisor
      )
;
```

--Anforderung1 Division „wenigstens eins und alle roten“

--SQL-Implementierung mit Common Table Expression

--und NOT EXISTS

```
WITH
dividend AS
(
 select DISTINCT
   tmitnrfnr.mitnr as mitnr
 from tmitnrfnr
)
,
divisor AS
(
select DISTINCT
   tfahrzeug.fnr as fnr
 from tfahrzeug
where farbe = 'rot'
)
,
mediator AS
(
select DISTINCT
   tmitnrfnr.mitnr as mitnr
 , tmitnrfnr.fnr as fnr
 From tmitnrfnr
)
SELECT
  dividend.mitnr AS mitnr
FROM dividend
WHERE NOT EXISTS
       (SELECT *
        FROM divisor
        WHERE NOT EXISTS
             (SELECT *
              FROM mediator
              WHERE
                    mediator.fnr = divisor.fnr
              AND
                    mediator.mitnr = dividend.mitnr
              )
     )
;
```

TABLE tmitarbeiter

```
mitnr
---------------
Emil
Franz
Fritz
Max
Otto
Xaver
```

TABLE tfahrzeug

```
fnr             farbe
--------------- ---------------
Auto            rot
Fahrrad         rot
Motorrad        lila
Unimog          grün
```

TABLE tmitnrfnr

```
mitnr           fnr
--------------- ---------------
Emil            Auto
Emil            Fahrrad
Franz           Fahrrad
Franz           Unimog
Fritz           Auto
Fritz           Motorrad
Max             Unimog
Otto            Auto
Otto            Fahrrad
Otto            Motorrad
```

Anforderung2 Division „alle roten“

Wer kann alle roten Fahrzeuge des Fuhrparks und auch vielleicht noch andere?

Sofern es keine roten Fahrzeuge gibt werden alle Mitarbeiter angelistet.

Pseudocode:

„tmitarbeiter{mitnr} DIVIDED BY Tfahrzeug{fnr WHERE farbe=rot}

PER tmitnrfnr{mitnr,fnr}“

--Anforderung2 Division „alle roten“

--SQL-Implementierung mit Common Table Expression

--und INNER JOIN und GROUP BY und UNION

--wir müssen nur den Dividenden ändern

```
WITH
dividend AS
(
 select DISTINCT
   tmitarbeiter.mitnr as mitnr
 from tmitarbeiter
)
,
divisor AS
(
select DISTINCT
   tfahrzeug.fnr as fnr
 from tfahrzeug
where farbe = 'rot'
)
,
mediator AS
(
select DISTINCT
   tmitnrfnr.mitnr as mitnr
 , tmitnrfnr.fnr as fnr
 From tmitnrfnr
)
SELECT
  mediator.mitnr as mitnr
FROM
     mediator INNER JOIN divisor
     ON mediator.Fnr = divisor.Fnr
GROUP BY     mediator.mitnr
HAVING COUNT(DISTINCT divisor.fnr)
=
(
 SELECT COUNT(divisor.fnr)
 FROM divisor
)
UNION
select
dividend.mitnr as mitnr
from dividend
where not exists
      (select *
       from divisor
      )
;
```

```
--Anforderung2 Division „alle roten“
--SQL-Implementierung mit Common Table Expression
--und NOT EXISTS
--wir müssen nur den Dividenden ändern
WITH
dividend AS
(
 select DISTINCT
   tmitarbeiter.mitnr as mitnr
 from tmitarbeiter
)
,
divisor AS
(
select DISTINCT
   tfahrzeug.fnr as fnr
 from tfahrzeug
where farbe = 'rot'
)
,
mediator AS
(
select DISTINCT
   tmitnrfnr.mitnr as mitnr
 , tmitnrfnr.fnr as fnr
 From tmitnrfnr
)
SELECT
  dividend.mitnr AS mitnr
FROM dividend
WHERE NOT EXISTS
      (SELECT *
       FROM divisor
       WHERE NOT EXISTS
            (SELECT *
             FROM mediator
             WHERE
                   mediator.fnr = divisor.fnr
             AND
                   mediator.mitnr = dividend.mitnr
             )
     )
;
```

--Datendefinition Tfahrzeug, Tmitarbeiter, Tmitnrfnr,

```
drop table tmitnrfnr;
drop table tmitarbeiter;
drop table tfahrzeug;

create table Tmitarbeiter
( mitnr      char (15) not null
 ,primary key (mitnr)
)
;
create table Tfahrzeug
( fnr char (15) not null
 ,farbe char (15) not null
 ,primary key (fnr)
)
;
create table tmitnrfnr
(mitnr     char(15)  not null
,fnr      char (15) not null
,primary key (mitnr, fnr     )
)
;
ALTER TABLE tmitnrfnr ADD FOREIGN KEy(fnr)
REFERENCES tfahrzeug(fnr);
ALTER TABLE tmitnrfnr ADD FOREIGN KEy(mitnr)
REFERENCES tmitarbeiter(mitnr);

insert into tmitarbeiter (mitnr) values ('Otto');
insert into tmitarbeiter (mitnr) values ('Fritz');
insert into tmitarbeiter (mitnr) values ('Emil');
insert into tmitarbeiter (mitnr) values ('Franz');
insert into tmitarbeiter (mitnr) values ('Max');
insert into tmitarbeiter (mitnr) values ('Xaver');

select mitnr from tmitarbeiter order by mitnr;

insert into tfahrzeug
(fnr     , farbe) values ('Auto', 'rot');
insert into tfahrzeug
(fnr     , farbe) values ('Fahrrad', 'rot');
insert into tfahrzeug
(fnr     , farbe) values ('Motorrad', 'lila');
insert into tfahrzeug
(fnr     , farbe) values ('Unimog', 'grün');

select fnr, farbe from tfahrzeug order by fnr;

insert into tmitnrfnr ( mitnr, fnr     )
values ('Otto','Auto');
insert into tmitnrfnr ( mitnr, fnr     )
values ('Otto','Motorrad');
insert into tmitnrfnr ( mitnr, fnr     )
values ('Otto','Fahrrad');
insert into tmitnrfnr ( mitnr, fnr     )
values('Fritz','Auto');
insert into tmitnrfnr ( mitnr, fnr     )
values('Fritz','Motorrad');
insert into tmitnrfnr ( mitnr, fnr     )
values('Emil','Auto');
insert into tmitnrfnr ( mitnr, fnr     )
values('Emil','Fahrrad');
insert into tmitnrfnr ( mitnr, fnr     )
values('Franz','Fahrrad');
insert into tmitnrfnr ( mitnr, fnr     )
values('Franz','Unimog');
insert into tmitnrfnr ( mitnr, fnr     )
values('Max','Unimog');

select mitnr, fnr
from tmitnrfnr
order by mitnr, fnr
;
```

5.23 Quota Query „die zwei teuersten Bücher“

```
select preis, buchnr
from tbuch
where preis is not null
order by preis desc, buchnr
;
preis     buchnr
--------- -----------
99.99     7
99.99     18
99.99     27
55.00     9
20.50     6
3.50      5
0.50      8
```

```
select preis, buchnr
from tbuch
where 1=1
and preis is not null
and
(select count(*) from tbuch aaa
 where aaa.preis > tbuch.preis) < 2;

preis     buchnr
--------- -----------
99.99     7
99.99     18
99.99     27
```

Mit Hilfe einer Reformulierung in normaler Umgangssprache ist die Select-Anweisung leichter zu verstehen:

„Ein Buch qualifiziert dann, wenn folgendes gilt: die Anzahl der teureren Bücher ist echt kleiner als 2“

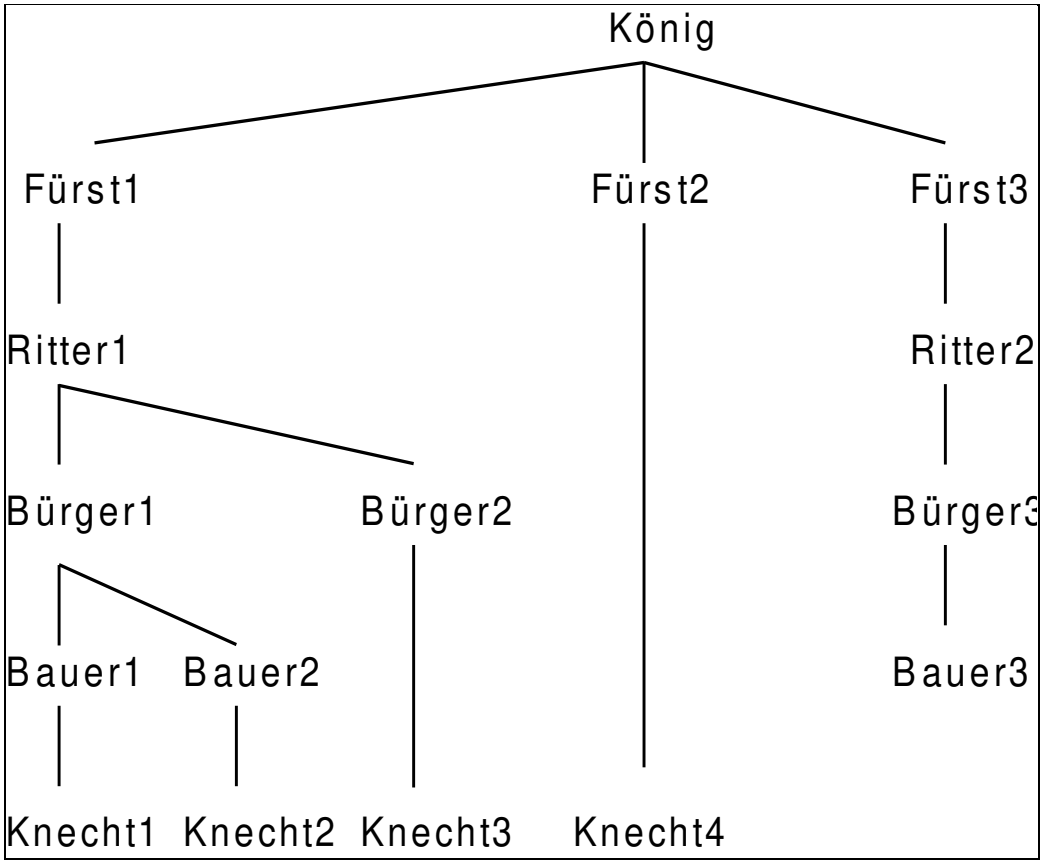
Liste der Bücher mit den 2 größten Preisen

```
select preis, buchnr
from tbuch
where preis is not null
order by preis desc, buchnr
;
preis     buchnr
--------- -----------
99.99     7
99.99     18
99.99     27
55.00     9
20.50     6
3.50      5
0.50      8
```

```
select preis, buchnr
from tbuch
where 1=1
and preis is not null
AND
(select count(distinct (preis))
 from tbuch aaa
 where aaa.Preis >= tbuch.preis) <= 2
ORDER BY preis, buchnr
;
preis           buchnr
--------------- -----------
55.00           9
99.99           7
99.99           18
99.99           27
```

5.24 Hierarchical Query, Recursive Query

5.24.1 Feudalgesellschaft



Eine Person kommt höchstens einmal im Baum vor. Im Folgenden wird nur dieser einfache Fall diskutiert.

```
DROP TABLE Tfeudal;
CREATE TABLE Tfeudal
(
   parent     CHAR (10)
 , dependent  CHAR (10) NOT NULL
 , steuer     DECIMAL (10)
 , PRIMARY KEY ( dependent )
 , FOREIGN KEY ( parent )
   REFERENCES Tfeudal(dependent)
----   ON DELETE NO ACTION
)
;
DELETE FROM Tfeudal;
INSERT INTO tfeudal( parent , dependent, steuer)
VALUES (NULL, 'könig',   NULL );
INSERT INTO tfeudal( parent , dependent, steuer)
VALUES ('könig', 'fürst1',   1000 );
INSERT INTO tfeudal( parent , dependent, steuer)
VALUES ('könig', 'fürst2', 2000 );
INSERT INTO tfeudal( parent , dependent, steuer)
VALUES ('könig', 'fürst3', 3000 );
INSERT INTO tfeudal( parent , dependent, steuer)
VALUES ('fürst1', 'ritter1', 1500 );
INSERT INTO tfeudal( parent , dependent, steuer)
VALUES ('fürst2', 'knecht4', 2500 );
INSERT INTO tfeudal( parent , dependent, steuer)
VALUES ('fürst3', 'ritter2', 3500 );
INSERT INTO tfeudal( parent , dependent, steuer)
VALUES ('ritter1', 'bürger1', 2001 );
INSERT INTO tfeudal( parent , dependent, steuer)
VALUES ('ritter1', 'bürger2', 2002 );
INSERT INTO tfeudal( parent , dependent, steuer)
VALUES ('bürger1', 'bauer1', 1501 );
INSERT INTO tfeudal( parent , dependent, steuer)
VALUES ('bürger1', 'bauer2', 1502 );
INSERT INTO tfeudal( parent , dependent, steuer)
VALUES ('bauer1', 'knecht1', 1601 );
INSERT INTO tfeudal( parent , dependent, steuer)
VALUES ('bauer2', 'knecht2', 1602 );
INSERT INTO tfeudal( parent , dependent, steuer)
VALUES ('bürger2', 'knecht3', 2503 );
INSERT INTO tfeudal( parent , dependent, steuer)
VALUES ('ritter2', 'bürger3', 3502 );
INSERT INTO tfeudal( parent , dependent, steuer)
VALUES ('bürger3', 'bauer3' , 3503 );

SELECT dependent, parent, steuer
FROM Tfeudal
order by dependent
;
dependent  parent     steuer
---------- ---------- ---------------------------------------
bauer1     bürger1    1501
bauer2     bürger1    1502
bauer3     bürger3    3503
bürger1    ritter1    2001
bürger2    ritter1    2002
bürger3    ritter2    3502
fürst1     könig      1000
fürst2     könig      2000
fürst3     könig      3000
knecht1    bauer1     1601
knecht2    bauer2     1602
knecht3    bürger2    2503
knecht4    fürst2     2500
könig      NULL       NULL
ritter1    fürst1     1500
ritter2    fürst3     3500

(16 row(s) affected)
```

----"alle, die direkt oder indirekt an Fürst1 Hirse abliefern?"

```
WITH tab123 (lev, par, dep, ste) as
(
  (
    SELECT
    1 as ebene
    ,Tfeudal.parent, Tfeudal.dependent, Tfeudal.steuer
    from Tfeudal
    where Tfeudal.parent ='fürst1'
   )
  UNION ALL
  (
    SELECT
    tab123.lev + 1 as ebene
   ,Tfeudal.parent, Tfeudal.dependent, Tfeudal.steuer
    from tab123 , Tfeudal
    where tab123.dep = Tfeudal.parent )
)
select lev, par, dep, ste
from tab123
order by lev, dep
;
```

```
lev         par         dep         ste
----------- ----------- ----------- --------
1           fürst1      ritter1     1500
2           ritter1     bürger1     2001
2           ritter1     bürger2     2002
3           bürger1     bauer1      1501
3           bürger1     bauer2      1502
3           bürger2     knecht3     2503
4           bauer1      knecht1     1601
4           bauer2      knecht2     1602

(8 row(s) affected)
```

Beispiel Oracle proprietäre Implementierung

```
SELECT LEVEL as lev
, parent as par, dependent as dep, steuer as ste
FROM tfeudal
CONNECT BY PRIOR  dependent = parent
START WITH parent = 'fürst1'
ORDER BY lev, dep
;
```

5.24.2 Erstellen von Testdaten mit Hilfe einer Rekursiven Query

Hole alle Einheitsintervalle von 1 bis 100 mit Hilfe einer Rekursiven Query (getestet mit SQL Server bzw. DB2 bzw. Oracle 11g).

```
VON         BIS
----------- -----------
          1           1
          2           2
          3           3
        ...         ...
       1000        1000
```

```
WITH tabxx (von, bis) AS
    (
          SELECT 1,1   ----FROM sysibm.sysdummy1
                       FROM DUAL
       UNION ALL
          SELECT  von+1 , bis+1
          FROM tabxx WHERE von <  1000
     )
SELECT
  tabxx.von as von
 ,tabxx.bis as bis
FROM tabxx
;
```

----oder mit WITH, UNION und kartesischem Produkt, CROSS JOIN, Beispiel SQL Server

```
with
zwi as
(
      select 0 as zahl union select 1 as zahl
union select 2 as zahl union select 3 as zahl
union select 4 as zahl union select 5 as zahl
union select 6 as zahl union select 7 as zahl
union select 8 as zahl union select 9 as zahl
)
,
zwa as
(
select
d4.zahl*1000 + d3.zahl*100 + d2.zahl*10 + d1.zahl + 1 as n
from
zwi  d1 CROSS JOIN zwi  d2 CROSS JOIN zwi  d3 CROSS JOIN zwi
d4
)
select n as von, n as bis from zwa
WHERE n <=1000
ORDER BY n
;
```

----oder mit Oracle proprietär

```
select
LEVEL as von , LEVEL as bis
FROM DUAL
CONNECT BY LEVEL <=1000
;
```

```
--- SQL Server
WITH tabxx (von, bis) AS
    (
          SELECT 0, 0  ----FROM sysibm.sysdummy1
                       ----FROM DUAL
       UNION ALL
          SELECT  von+1 , bis+1
          FROM tabxx WHERE von <  30
     )
SELECT
 DATEADD( DAY, von , CAST('2020-01-01' AS DATE)) as von
,DATEADD( DAY, bis , CAST('2020-01-01' AS DATE)) as von
FROM tabxx
;

--- Db2
WITH tabxx (von, bis) AS
    (
          SELECT 0,0   FROM sysibm.sysdummy1
                       ----FROM DUAL
       UNION ALL
          SELECT  von+1 , bis+1
          FROM tabxx WHERE von <  30
     )
SELECT
 CAST('2020-01-01' AS DATE) + von DAYS as von
,CAST('2020-01-01' AS DATE) + bis DAYS as bisclosed
FROM tabxx
;

--- Oracle
alter session set nls_date_format =  'yyyy-mm-dd hh24:mi:ss';

WITH tabxx (von, bis) AS
    (
       SELECT 0,0  FROM DUAL
       UNION ALL
          SELECT  von+1 , bis+1
          FROM tabxx WHERE von <  30
     )
SELECT
TO_DATE('2020-01-01','yyyy-mm-dd')
  + NUMTODSINTERVAL(+von ,'DAY')            as von
,TO_DATE('2020-01-01', 'yyyy-mm-dd') + bis  as bisclosed
FROM tabxx
;
```

```
--- SQL Server
WITH tabxx (von, bis) AS
    (
          SELECT 0, 0  ----FROM sysibm.sysdummy1
                       ----FROM DUAL
       UNION ALL
          SELECT  von+1 , bis+1
          FROM tabxx WHERE von <  30
     )
SELECT
 DATEADD( MILLISECOND, von , CAST('2020-01-01' AS DATETIME2(3))) as von
,DATEADD( MILLISECOND, bis , CAST('2020-01-01' AS DATETIME2(3))) as von
FROM tabxx
;
von                         von
--------------------------- ---------------------------
2020-01-01 00:00:00.000     2020-01-01 00:00:00.000
2020-01-01 00:00:00.001     2020-01-01 00:00:00.001
2020-01-01 00:00:00.002     2020-01-01 00:00:00.002
2020-01-01 00:00:00.003     2020-01-01 00:00:00.003
2020-01-01 00:00:00.004     2020-01-01 00:00:00.004
2020-01-01 00:00:00.005     2020-01-01 00:00:00.005
2020-01-01 00:00:00.006     2020-01-01 00:00:00.006
2020-01-01 00:00:00.007     2020-01-01 00:00:00.007
2020-01-01 00:00:00.008     2020-01-01 00:00:00.008
2020-01-01 00:00:00.009     2020-01-01 00:00:00.009
2020-01-01 00:00:00.010     2020-01-01 00:00:00.010
2020-01-01 00:00:00.011     2020-01-01 00:00:00.011
2020-01-01 00:00:00.012     2020-01-01 00:00:00.012
2020-01-01 00:00:00.013     2020-01-01 00:00:00.013
2020-01-01 00:00:00.014     2020-01-01 00:00:00.014
2020-01-01 00:00:00.015     2020-01-01 00:00:00.015
2020-01-01 00:00:00.016     2020-01-01 00:00:00.016
2020-01-01 00:00:00.017     2020-01-01 00:00:00.017
2020-01-01 00:00:00.018     2020-01-01 00:00:00.018
2020-01-01 00:00:00.019     2020-01-01 00:00:00.019
2020-01-01 00:00:00.020     2020-01-01 00:00:00.020
2020-01-01 00:00:00.021     2020-01-01 00:00:00.021
2020-01-01 00:00:00.022     2020-01-01 00:00:00.022
2020-01-01 00:00:00.023     2020-01-01 00:00:00.023
2020-01-01 00:00:00.024     2020-01-01 00:00:00.024
2020-01-01 00:00:00.025     2020-01-01 00:00:00.025
2020-01-01 00:00:00.026     2020-01-01 00:00:00.026
2020-01-01 00:00:00.027     2020-01-01 00:00:00.027
2020-01-01 00:00:00.028     2020-01-01 00:00:00.028
2020-01-01 00:00:00.029     2020-01-01 00:00:00.029
2020-01-01 00:00:00.030     2020-01-01 00:00:00.030

(31 row(s) affected)

--- Db2 ?




--- Oracle ?
```

5.24.3 Stücklisten - Bill of Materials Implementierung

```
TABLE tabstueckliste
teilnroben  teilnrunten menge
----------- ----------- -----------
7788        123         4
7788        1           2
123         45          3
123         14          9
123         1           1
45          5           3
45          4           7
14          4           3
14          1           8

(9 row(s) affected)

DROP TABLE tabstueckliste;
DROP TABLE tabteil;

CREATE TABLE tabteil
(
teilnr integer not null
,PRIMARY KEY(teilnr)
,teilname VARCHAR(200) not null
)
;
CREATE TABLE tabstueckliste
(
 teilnroben  INTEGER NOT NULL
,teilnrunten INTEGER NOT NULL
,menge       integer not null
,PRIMARY KEY(teilnroben,teilnrunten)
);
ALTER TABLE tabstueckliste
ADD CONSTRAINT conimp
FOREIGN KEY (teilnroben)
REFERENCES  tabteil(teilnr)
;
ALTER TABLE tabstueckliste
ADD CONSTRAINT conexp
FOREIGN KEY (teilnrunten)
REFERENCES  tabteil(teilnr)
;
go
insert into tabteil (teilnr, teilname)
values (7788, 'teil7788');
insert into tabteil (teilnr, teilname)
values (1, 'teil1');
insert into tabteil (teilnr, teilname)
values (123, 'teil123');
insert into tabteil (teilnr, teilname)
values (45, 'teil45');
insert into tabteil (teilnr, teilname)
values (14, 'teil14');
insert into tabteil (teilnr, teilname)
values (4, 'teil4');
insert into tabteil (teilnr, teilname)
values (5, 'teil5');


insert into tabstueckliste
```

```
( menge,teilnroben, teilnrunten)
values ( 2, 7788, 1)
;
insert into tabstueckliste
( menge, teilnroben, teilnrunten)
values ( 4, 7788, 123)
;
insert into tabstueckliste
(menge, teilnroben, teilnrunten)
values ( 1, 123,1)
;
insert into tabstueckliste
( menge, teilnroben, teilnrunten)
values ( 3, 123,45)
;
insert into tabstueckliste
(menge, teilnroben, teilnrunten)
values ( 9, 123,14)
;
insert into tabstueckliste
(menge, teilnroben, teilnrunten)
values ( 7, 45,4)
;
insert into tabstueckliste
(menge, teilnroben, teilnrunten)
values ( 3, 45, 5)
;
insert into tabstueckliste
(menge, teilnroben, teilnrunten)
values ( 8, 14,1)
;
insert into tabstueckliste
(menge, teilnroben, teilnrunten)
values ( 3, 14, 4)
;
select teilnroben, teilnrunten, menge
from tabstueckliste
order by teilnroben desc, teilnrunten desc
;
```

```
TABLE tabstueckliste
teilnroben  teilnrunten menge
----------- ----------- -----------
7788        123         4
7788        1           2
123         45          3
123         14          9
123         1           1
45          5           3
45          4           7
14          4           3
14          1           8

(9 row(s) affected)
```

```
--gewünschte Liste part explosion
zeilennr             ebene       teilnroben  teilnrunten menge
-------------------- ----------- ----------- ----------- -----------
1                    1           7788        1           2
2                    1           7788        123         4
3                    2           123         1           1
4                    2           123         14          9
5                    2           123         45          3
6                    3           14          1           8
7                    3           14          4           3
8                    3           45          4           7
9                    3           45          5           3

(9 row(s) affected)
```

```
WITH
zwi (teilnroben, teilnrunten,  menge, ebene)
AS
(
  ( SELECT abc.teilnroben   as teilnroben
         , abc.teilnrunten  as teilnrunten
         , abc.menge        as menge
           , 1                as ebene
    from tabstueckliste  abc
    where abc.teilnroben = 7788
  )
  UNION ALL
  ( select abc.teilnroben    as teilnroben
          ,abc.teilnrunten   as teilnrunten
          ,abc.menge         as menge
          ,zwi.ebene + 1 as ebene
    from zwi inner join tabstueckliste abc
         on zwi.teilnrunten = abc.teilnroben
  )
)
select
 ROW_NUMBER() OVER( order by ebene, teilnroben, teilnrunten )
 as zeilennr
,zwi.ebene as ebene
,zwi.teilnroben  as teilnroben
,zwi.teilnrunten as teilnrunten
,zwi.menge       as menge
from zwi
order by  ebene, teilnroben, teilnrunten
;
```

Beispiel Oracle proprietäre Implementierung

```
SELECT
 ROWNUM        as zeilennr
,LEVEL         as ebene
, teilnroben   as teilnroben
, teilnrunten  as teilnrunten
, menge        as menge
FROM tabstueckliste
CONNECT BY PRIOR  teilnrunten = teilnroben
START WITH teilnroben = 7788
ORDER BY ebene, teilnroben, teilnrunten
;

--welche teilnr ist oben in der Hierarchie
teilnroben
-----------
7788

select DISTINCT
abc.teilnroben as teilnroben
from tabstueckliste abc
where not exists
    (select * from tabstueckliste xyz
     where xyz.teilnrunten=abc.teilnroben)
;

--welche teilnr sind unten in der Hierarchie
einzelteil
-----------
1
4
5
select  DISTINCT
 abc.teilnrunten  as einzelteil
from tabstueckliste abc
where not exists
    (select * from tabstueckliste xyz
     where xyz.teilnroben=abc.teilnrunten)
order by teilnrunten
;
--aus welchen Einzelteilen setzt sich teil 7788 zusammen?
einzelteil  anzahl
----------- -----------
1           11
4           10
5           3

select
 abc.teilnrunten  as einzelteil
,SUM(abc.menge)   as anzahl
from tabstueckliste abc
where not exists
    (select * from tabstueckliste xyz
     where xyz.teilnroben=abc.teilnrunten)
GROUP BY teilnrunten
order by teilnrunten
;
```

5.25 Transformationen GROUP BY HAVING und Subqueries

Transformation 1 GROUP BY

Für jedes Buch mit Autoren, hole Buchnr, maximal laufende und minimal laufende Nummer, Anzahl der Autoren.

```
SELECT
       Tvautor.Buchnr           AS Buchnr
     , MAX(Tvautor.Lfdnr)       AS Maxlfdnr
     , COUNT(tvautor.autornr)   AS countautornr
FROM Tvautor
GROUP BY Tvautor.Buchnr
;
SELECT DISTINCT
       Tvautor.Buchnr
                                  AS Buchnr
     , (SELECT MAX(Abc.Lfdnr)
        FROM Tvautor Abc
        WHERE Abc.Buchnr = Tvautor.Buchnr)
                                  AS Maxlfdnr
     , (SELECT COUNT(abc.autornr)
        FROM Tvautor Abc
        WHERE Abc.Buchnr = Tvautor.Buchnr)
                                  AS countautornr
FROM Tvautor
;
```

Transformation 2 HAVING

Für jedes Buch mit mehr als einem Autor, hole die Buchnummer.

```
SELECT
       Tvautor.Buchnr           AS Buchnr
FROM Tvautor
GROUP BY Tvautor.Buchnr
HAVING COUNT(tvautor.autornr) > 1
;
SELECT DISTINCT
       Tvautor.Buchnr           AS Buchnr
FROM Tvautor
WHERE (SELECT COUNT(abc.autornr)
       FROM Tvautor Abc
       WHERE Abc.Buchnr = Tvautor.Buchnr) >1
;
```

Korrelierte skalare Subquery in der SELECT-Klausel bzw. in der WHERE-Klausel. Die Subquery benützt einen Korrelationsnamen (bzw. eine Range Variable):

```
(SELECT COUNT(abc.autornr)
 FROM Tvautor Abc
 WHERE Abc.Buchnr = Tvautor.Buchnr)
```

Wir nehmen an, es gäbe eine Kopie der Table Tvautor, ansprechbar unter dem Namen Abc. Das System muss die Kopie Abc nicht physisch erstellen, die Betrachtung ist konzeptioneller Art.

Das System untersucht jede Zeile der äußeren Tvautor. Angenommen, die aktuell untersuchte Zeile hat Buchnr=12, Tvautor.Buchnr hat also aktuell den Wert 12. Das System führt konzeptionell die innere Query

```
(SELECT COUNT(abc.autornr)
 FROM Tvautor Abc
 WHERE Abc.Buchnr = 2)
```

aus um den Skalarwert zu erhalten.

Type 1 Transformation (es sei keine NULL im Spiel):

sei R{A,B,...} ein Table, agg eine aggregate function, jedes Statement

```
SELECT
       R.A       AS spalte1
      ,agg(R.B)  AS spalte2
FROM R
GROUP BY R.A;
```

kann in das äquivalente Statement (mit scalar subquery) umgewandelt werden:

```
SELECT DISTINCT
       R.A                  AS spalte1
      ,(SELECT agg(Rx.B)
        FROM R Rx
        WHERE Rx.A = R.A)   AS spalte2
FROM R;
```

Type 2 Transformation (es sei keine NULL im Spiel):

sei R{A,B,...} ein Table, agg eine aggregate function, jedes Statement

```
SELECT
       R.A   AS spalte1
FROM R
GROUP BY R.A
HAVING agg(R.B) comp scalar;
```

kann in das äquivalente Statement (mit scalar subquery) umgewandelt werden:

```
SELECT DISTINCT
       R.A    AS spalte1
FROM R
WHERE  (SELECT agg(Rx.B)
        FROM R Rx
        WHERE Rx.A = R.A) comp scalar;
```

5.26 Die Summarize-Operation

Die Summarize-Operation ist die Verallgemeinerung der „Verdichtungsoperation mit GROUP BY“.

```
select buchnr from tbuch order by buchnr
;
select buchnr, autornr, praemie from tvautor order by buchnr, autornr
;
TABLE Tbuch
buchnr
-----------
1
2
5
6
7
8
9
11
12
18
27

(11 row(s) affected)

TABLE Tvautor
buchnr      autornr     praemie
----------- ----------- ---------------------------------------
1           100         NULL
1           200         NULL
2           200         NULL
5           1           NULL
6           2           NULL
7           3           NULL
8           2           20.000
9           10          10.000
9           11          498.000
12          20          30.000
12          21          NULL
```

gewünschte Liste
pro Buch die Anzahl der Autoren und die Summe der Praemien

```
buchnr      countautornr sumpraemie
----------- ------------ ---------------------------------------
1           2             0.000
2           1             0.000
5           1             0.000
6           1             0.000
7           1             0.000
8           1             20.000
9           2             508.000
11          0             0.000
12          2             30.000
18          0             0.000
27          0             0.000

(11 row(s) affected)
```

Manche Bücher haben wenigstens einen Autor aber keine wohldefinierte Praemie, die Summe der Praemien ist dann 0.

- Interessant sind die Zeilen mit den Buchnummern 11, 18 bzw. 27; die Bücher mit diesen Buchnummern haben keine Autoren und damit auch keine Prämien! Die Summe der Praemien ist dann auch 0.
- Ein Buch kann wohldefinierte Praemien haben, aber vielleicht sind alle 0, die Summe der Praemien ist dann 0.

Beachten Sie bitte, dass zwischen tvautor.buchnr und tbuch.buchnr eine Fremdschlüssel-Primärschlüsselbeziehung besteht!

Beachten Sie bitte, dass in der Liste aus Tbuch nur die Spalte Buchnr angelistet ist.

--Variante-1a mit LEFT OUTER JOIN und guter Performance

```
WITH
zwi AS
(
 select
 tva.buchnr          AS buchnr
,COUNT(tva.autornr)  AS countautornr
,SUM(tva.praemie)    AS sumpraemie
from tvautor tva
GROUP BY tva.buchnr
)
select
 tbu.buchnr                    as buchnr
,COALESCE(zwi.countautornr,0)  as countautornr
,COALESCE(zwi.sumpraemie,0)    as sumpraemie
from tbuch tbu LEFT OUTER JOIN zwi
on tbu.buchnr = zwi.buchnr
order by buchnr
;
```

--Variante-1b mit LEFT OUTER JOIN

```
select
 tbu.buchnr                    as buchnr
,COALESCE(zwi.countautornr,0)  as countautornr
,COALESCE(zwi.sumpraemie,0)    as sumpraemie
from tbuch tbu LEFT OUTER JOIN
(
select
 tva.buchnr          AS buchnr
,COUNT(tva.autornr)  AS countautornr
,SUM(tva.praemie)    AS sumpraemie
from tvautor tva
GROUP BY tva.buchnr
) zwi
on tbu.buchnr = zwi.buchnr
order by buchnr
;
```

--Variante-2 mit UNION und NOT EXISTS

```
select
       tva.buchnr                          AS buchnr
      ,COUNT(tva.autornr)                  AS countautornr
      ,COALESCE(sum(tva.praemie), 0.000) AS sumpraemie
from tvautor tva
GROUP BY tva.buchnr
UNION
select
      tbu.buchnr             AS buchnr
     ,0                      AS countautornr
     ,0.000                  AS sumpraemie
FROM tbuch tbu
where not exists
      ( select * from tvautor tva
        where tva.buchnr = tbu.buchnr)
Order by buchnr
;
```

--Variante-3 mit korrelierten Subqueries in der Select-Klausel

```
 SELECT
    buchnr                                    AS buchnr
   ,( select COUNT(tva.autornr)
      from tvautor tva
      where tva.buchnr = tbu.buchnr)          AS countautornr
   ,( select COALESCE(SUM(tva.praemie) , 0.000)
      from tvautor tva
      where tva.buchnr = tbu.buchnr)          AS sumpraemie
FROM tbuch tbu
order by buchnr
;
```

--Variante-4 mit OUTER APPLY SQL Server, Oracle

```
select
 tbu.buchnr                    as buchnr
,COALESCE(zwi.countautornr,0)  as countautornr
,COALESCE(zwi.sumpraemie,0)    as sumpraemie
from tbuch tbu OUTER APPLY
(
select
 COUNT(tva.autornr)  AS countautornr
,SUM(tva.praemie)    AS sumpraemie
from tvautor tva
WHERE tva.buchnr = tbu.buchnr
) zwi
order by buchnr
;
```

--Variante-5 mit LEFT OUTER JOIN TABLE Db2

```
select
 tbu.buchnr                    as buchnr
,COALESCE(zwi.countautornr,0)  as countautornr
,COALESCE(zwi.sumpraemie,0)    as sumpraemie
from tbuch tbu LEFT OUTER JOIN TABLE
(
select
 COUNT(tva.autornr)  AS countautornr
,SUM(tva.praemie)    AS sumpraemie
from tvautor tva
WHERE tva.buchnr = tbu.buchnr
) zwi
ON 1=1
order by buchnr
;
```

5.27 UNIQUE wird von wenigen Produkten am Markt unterstützt

Tautor

Autornr Autor

44

Tvautor

Autornr Lfdnr Buchnr

44 5

44 5

Autoren, die wenigstens zwei verschiedene Bücher geschrieben haben in gleicher Funktion (als einziger bzw. als 1. bzw. als 2. bzw. als 3. Autor ...).

```
SELECT Autornr, Autor
FROM Tautor
WHERE NOT UNIQUE
           (SELECT Lfdnr
            FROM Tvautor
            WHERE Tvautor.Autornr = Tautor.Autornr)
Formulierung ohne UNIQUE:
SELECT Autornr, Autor
FROM Tautor
WHERE EXISTS
   (SELECT * FROM
    Tvautor xxx
    WHERE EXISTS
       (SELECT * FROM Tvautor yyy
        WHERE xxx.Autornr  = Tautor.Autornr
        AND   yyy.Autornr  = Tautor.Autornr
        AND   xxx.Lfdnr    = yyy.Lfdnr
        AND   xxx.Buchnr  <> yyy.Buchnr
  ))
```

UNIQUE ist wahr, sofern die innere SELECT-Anweisung eine Tabelle ergibt, in der alle Zeilen unterschiedlich sind.

UNIQUE ist wahr, sofern die innere SELECT-Anweisung die leere Menge liefert.

UNIQUE ist wahr, sofern die innere SELECT-Anweisung mit SELECT DISTINCT codiert ist.

Unterstützt Ihr Server UNIQUE?

Zitat Anfang

UNIQUE-Bedingungen werden verwendet, um zu testen, ob jede Zeile in einer Tabelle eindeutig ist (d. h. es gibt keine Duplikate). Die Syntax :

```
UNIQUE( tabellenausdruck )
```

Die Bedingung wird zu *true* ausgewertet, wenn der Tabellenausdruck zu einer Tabelle ausgewertet wird in der alle Zeilen unterschiedlich (distinkt) sind, ansonsten zu *false*. Man beachte deswegen insbesondere, daß die Bedingung immer dann zu *true* ausgewertet wird, wenn das Argument nur eine Zeile enthält oder leer ist (d. h., überhaupt keine Zeilen enthält). Sie wir auch notwendigerweise zu **true** ausgewertet, wenn das Argument die Form SELECT DISTINCT ... hat (oder jede der verschiedenen anderen Formate von Tabellenausdrücken, die garantieren, daß es keine Duplikate im Ergebnis gibt - beispielsweise eine Vereinigungsmenge ohne die Option ALL).

Hier ist ein Beispiel (« Ermittle Namen der Lieferanten, die mindestens zwei unterschiedliche Produkte in derselben Stückzahl liefern. ») :

```
SELECT DISTINCT S.SNAME
FROM S
WHERE NOT UNIQUE ( SELECT SP.QTY
                   FROM SP
                   WHERE SP.SNO = S.SNO )
```

Hinweis : Der Leser mag erkennen, daß wir aus der Not eine Tugend machen. Wir haben mehrere Male in diesem Buch erwähnt, daß Duplikate in Tabellen nach unserer Meinung nie erlaubt sein sollten. Würde diese Disziplin befolgt, dann wäre UNIQUE natürlich sinnlos (weil es immer zu true ausgewertet würde). Wir haben das « realistische » Beispiel oben gezeigt, um eine « realistische » Anwendung einer UNIQUE-Bedingung zu zeigen, aber der Leser sollte aus diesem Beispiel nicht den Schluß ziehen, daß das System Duplikaten erlauben muß, um Anfragen wie die oben diskutierte zu unterstützen.

Zitat Ende

SQL - Der Standard SQL/92 mit den Erweiterungen CLI und PSM Deutsche Ausgabe des amerikanischen Klassikers Ausblick auf SQL3 Chris J. Date Hugh Darwen Addison Wesley Longman GmbH, 1998 Seite 207

Zitat Anfang

"If there are no two rows ... such that the value of each column in one row is non-null and is equal to the value of the corresponding column in the other row ..., then the result of the <unique predicate> is true; otherwise, the result of the <unique predicate> is false."

Zitat Ende

DIN 66315 Seite 183

6

INSERT, UPDATE, DELETE, MERGE, COMMIT und ROLLBACK, Transaktionsverarbeitung

6 INSERT, UPDATE, DELETE, MERGE, COMMIT und ROLLBACK, Transaktionsverarbeitung

6.1 Daten sind korrekt, Daten sind konsistent

Daten sind konsistent, wenn sie alle im System definierten Integritätsbedingungen erfüllen.

Daten sind korrekt, wenn sie die Realität richtig beschreiben.

Anfang der Transaktion

Girokonto +100

Sparkonto -100

Ende der Transaktion

Eine Transaktion ist eine logische Verarbeitungseinheit von Daten, alle Veränderungen oder keine muss durchgeführt werden.

6.2 INSERT

```
-- INSERT eine Zeile
INSERT INTO Tbuch (Buchnr, Erschj, Titel)
VALUES (0815 ,  1991.  , 'Hintertupfingen');


INSERT INTO Tbuch (Titel, Preis, Buchnr)
VALUES ('Köln' , 123.78  , 4711);
```

```
CREATE TABLE Tbuchnrtitel
 (
  Buchnrabc INTEGER NOT NULL
 ,Titelabc  VARCHAR (127) NOT NULL
 ,PRIMARY KEY (Buchnrabc)
 );


-- INSERT mehrere Zeilen
INSERT INTO Tbuchnrtitel (Titelabc, Buchnrabc)
       SELECT Titel,Buchnr FROM Tbuch;
```

Mit der INSERT-Anweisung können in eine TABLE eine oder mehrere Zeilen eingefügt werden.

- Jede mit Hilfe einer SELECT-Anweisung ermittelte Menge kann auf diese Weise in eine passende TABLE eingefügt werden.
- Spalten die mit NOT NULL definiert sind, müssen beim INSERT angegeben werden (vgl. die Spalten Tbuch.Titel und Tbuch.Buchnr).
- Spalten, die beim INSERT nicht angegeben werden, erhalten den "Wert NULL", sofern das auf Grund der Definition möglich ist (vgl. die Spalten Tbuch.Preis und Tbuch.Erschj).
- Wenn die Namen der Spalten nicht angegeben werden, dann müssen die Werte in der Reihenfolge der Spalten wie im CREATE TABLE angegeben werden. Es muss dann auch für jede Spalte ein Wert angegeben werden.

6.3 UPDATE

```
UPDATE Tbuch
   SET    Preis = 25
   WHERE  Buchnr = 11;

UPDATE Tbuch
   SET    Preis = 19.99
   WHERE  ERSCHJ > 2000 AND Preis < 9.99;

UPDATE Tbuch
   SET    Erschj  = NULL
   WHERE  Erschj = 1990;

UPDATE Tbuch
   SET    Preis = Preis + 10;

UPDATE Tbuch
   SET    Preis = (SELECT MAX(Preis) FROM Tbuch);

UPDATE Tbuch
   SET    Preis = -77.77 , Erschj = -2000
   WHERE  Buchnr = 11;
```

Mit der UPDATE-Anweisung können in einer TABLE eine oder mehrere Zeilen verändert werden.

- Wenn eine einzelne Zeile verändert werden soll, dann muss der Primärschlüssel zum Zugriff benutzt werden.
- Mit einer einzigen Anweisung können alle Zeilen einer TABLE verändert werden.
- Syntaktisch illegal ist (was SQL:1992 und einige Produkte am Markt betrifft) "SET Tbuch.Preis", d. h. die Referenz auf die Spalte darf nicht mit dem Namen der Table qualifiziert werden.

Achtung:

```
update tbuch set titel = '';
```

DB2 und SQL Server kennen einen VARCHAR-String der Länge 0.

Oracle interpretiert einen VARCHAR2-String ohne Daten als NULL mit der Länge NULL.

6.4 DELETE

```
DELETE FROM Tbuch
   WHERE  Buchnr = 11;


DELETE FROM Tbuch
   WHERE  Erschj = 1955;


DELETE FROM Tbuch;
--DELETE FROM Tisbn;
--DELETE FROM Tvautor;


DELETE FROM Tautor;
```

Mit der DELETE-Anweisung können in einer TABLE eine oder mehrere Zeilen gelöscht werden.

- Wenn eine einzelne Zeile gelöscht werden soll, dann muss der Primärschlüssel zum Zugriff benutzt werden.
- Mit einer einzigen Anweisung können alle Zeilen einer TABLE gelöscht werden.
- Nach dem Löschen aller Zeilen bleibt die TABLE erhalten.
- **Ein DELETE in einer Primärschlüsseltabelle wird möglicherweise abgewiesen bzw. hat möglicherweise Auswirkungen auf andere Tabellen. Vergleichen Sie bitte dazu die Diskussion über Referentielle Integrität!**

Del001: Löschen Sie alle Bücher, die keinen Autor haben!

```
DELETE FROM Tbuch
WHERE          Tbuch.Buchnr NOT IN
      (SELECT Tvautor.Buchnr
       FROM Tvautor);
```

Del002: Löschen Sie alle Autoren, die keine Bücher haben!

```
DELETE FROM Tautor
WHERE         Tautor.Autornr NOT IN
      (SELECT Tvautor.Autornr
       FROM Tvautor);
```

Del003: Löschen Sie alle Verlage, die keine Bücher haben!

```
DELETE FROM Tverlag
WHERE      Tverlag.Verlagnr NOT IN
      (SELECT Tbuch.Verlagnr
       FROM Tbuch
       WHERE Tbuch.Verlagnr IS NOT NULL
      );
```

6.5 MERGE

```
select * from tbuch
;
MERGE INTO tbuch
USING tbuchpreisneu abc
ON ( tbuch.buchnr = abc.buchnr )
WHEN MATCHED THEN
      UPDATE SET tbuch.preis = abc.preis
                , tbuch.titel = abc.titel
WHEN NOT MATCHED THEN
      INSERT (buchnr     , preis   , titel)
      VALUES (abc.buchnr, abc.preis, abc.titel)
;
select * from tbuch;
Rollback;
select * from tbuch;
UPDATE tbuch SET tbuch.preis =
                 (SELECT abc.preis
                    FROM tbuchpreisneu abc
                    WHERE abc.buchnr = tbuch.buchnr)
                ,tbuch.titel =
                  (SELECT abc.titel
                    FROM tbuchpreisneu abc
                    WHERE abc.buchnr = tbuch.buchnr)
WHERE EXISTS
              (SELECT * FROM
               tbuchpreisneu abc
               WHERE abc.buchnr = tbuch.buchnr
              )
;
select * from tbuch;
INSERT INTO tbuch (buchnr, preis, titel)
            SELECT abc.buchnr, abc.preis, abc.titel
            FROM  tbuchpreisneu abc
            WHERE NOT EXISTS
                  (SELECT * FROM tbuch
                   WHERE tbuch.buchnr = abc.buchnr)
;
select * from tbuch;
rollback;
select * from tbuch;
```

Die MERGE-Anweisung ist funktional äquivalent zu den beiden INSERT- und UPDATE-Anweisungen.

```
drop table tbuchpreisneu
;
create table tbuchpreisneu
(
  buchnr integer      not null
 ,preis DECIMAL (7,2) not null
 ,titel varchar (127) NOT NULL
 ,primary key (buchnr)
 );
 insert into tbuchpreisneu (buchnr , preis, titel   )
 values                    ( 1     , 11.11,  'AAAA' )
 ;
 insert into tbuchpreisneu (buchnr , preis, titel   )
 values                    ( 3     , 33.33 , 'CCCC' )
 ;
 insert into tbuchpreisneu (buchnr , preis, titel   )
 values                    ( 27     , 27.27 , 'xyz' )
 ;
 select * from tbuchpreisneu
 ;
 commit;
```

6.5.1 MERGE und Syntax und Logik

Wir verändern die Aufgabenstellung um einige Besonderheiten der verschiedenen Produkte zu demonstrieren.

Die table txyz hat neben der Spalte Preis noch eine Spalte Preisart, der Primärschlüssel von txyz ist (buchnr, preisart). Nach Tbuch sollen die Preise und der Titel einer gewissen Preisart übernommen werden.

```
txyz
preisart buchnr      preis                     titel
-------- ----------- ------------------------- ------------
1        1           11.11                     ABC1#1
1        3           33.33                     ABC1#3
1        27          27.27                     ABC1#27
2        1           2211.11                   ABC2#1
2        3           2233.33                   ABC2#3
2        27          2227.27                   ABC2#27
```

Welche der folgenden Varianten ist logisch korrekt und wird von Ihrem Server unterstützt?

--Variante 1

```
MERGE INTO tbuch
USING txyz abc
ON ( tbuch.buchnr = abc.buchnr
   )
WHEN MATCHED and preisart= '1' THEN
      UPDATE SET tbuch.preis = abc.preis
                , tbuch.titel = abc.titel
WHEN NOT MATCHED and preisart= '1' THEN
      INSERT (buchnr     , preis   , titel)
      VALUES (abc.buchnr, abc.preis, abc.titel)
;
------------------------------
```

--Variante 2

```
WITH neuedaten as
(
  select aa.buchnr
        ,aa.preis
        ,aa.titel
  from txyz aa
  where aa.preisart =  '1'
)
MERGE INTO tbuch
USING neuedaten abc
ON ( tbuch.buchnr = abc.buchnr
   )
WHEN MATCHED THEN
      UPDATE SET tbuch.preis = abc.preis
                , tbuch.titel = abc.titel
WHEN NOT MATCHED THEN
      INSERT (buchnr     , preis   , titel)
      VALUES (abc.buchnr, abc.preis, abc.titel)
;
------------------------------
```

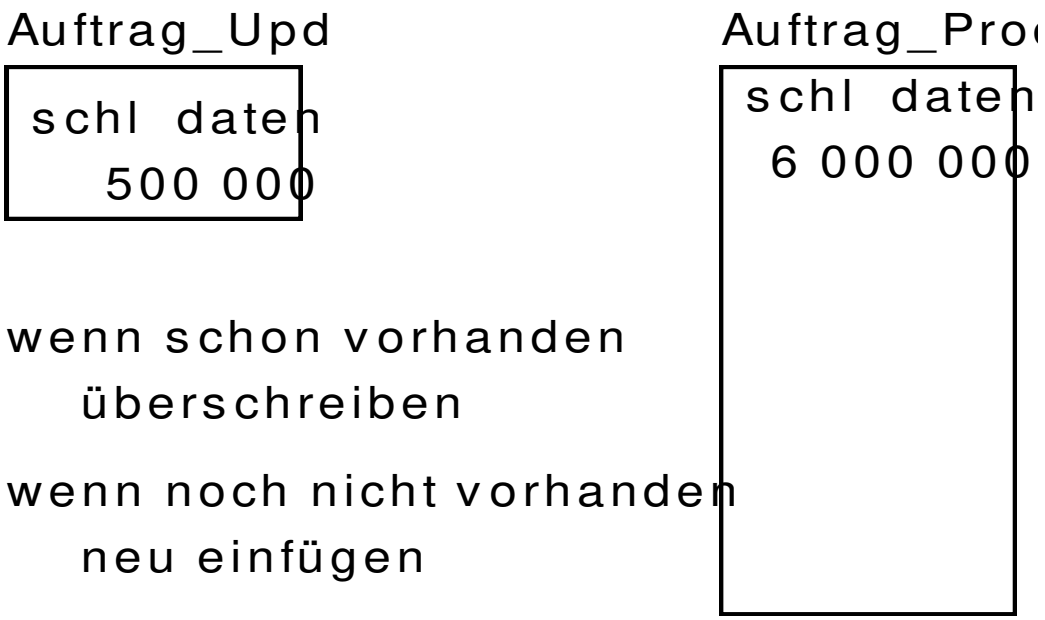
--Variante 3 logisch nicht korrekt!

```
MERGE INTO tbuch
USING txyz abc
ON ( tbuch.buchnr = abc.buchnr
     and preisart= '1'
   )
WHEN MATCHED THEN
      UPDATE SET tbuch.preis = abc.preis
                , tbuch.titel = abc.titel
WHEN NOT MATCHED THEN
      INSERT (buchnr     , preis   , titel)
      VALUES (abc.buchnr, abc.preis, abc.titel)
;

drop table txyz
;
create table txyz
(
  buchnr integer        not null
 ,preisart char(1)      not null
 ,preis DECIMAL (7,2)   not null
 ,titel varchar (127)NOT NULL
 ,primary key (buchnr, preisart)
 )
;
insert into txyz (buchnr , preis, preisart , titel )
 values          ( 1     , 11.11, '1',       'ABC1#1');
insert into txyz (buchnr , preis, preisart , titel )
 values          ( 3     , 33.33, '1',       'ABC1#3');
insert into txyz (buchnr , preis, preisart , titel )
 values          ( 27    , 27.27, '1',       'ABC1#27');
insert into txyz (buchnr , preis, preisart , titel )
 values          ( 1     , 2211.11, '2',     'ABC2#1');
insert into txyz (buchnr , preis, preisart , titel )
 values          ( 3     , 2233.33, '2',     'ABC2#3');
insert into txyz (buchnr , preis, preisart , titel )
 values          ( 27    , 2227.27, '2',     'ABC2#27');
select preisart, buchnr, preis, titel
from txyz
order by preisart, buchnr;
```

6.5.2 MERGE, Massenupdate, FULL OUTER JOIN

Mit Hilfe ein TABLE Auftrag-Upd soll in regelmäßigen Abständen eine TABLE Auftrag_Prod bearbeitet werden. Eine Zeile in Auftrag_Upd ersetzt die Zeile in Auftrag_Prod, sofern sie dort vorhanden ist. Wenn sie nicht vorhanden ist, soll sie eingefügt werden.

Diskutieren Sie das Programmdesign!



Der Optimizer macht die Arbeit (sortiertes Entladen). Die qualifizierenden Zeilen werden vom Entladeutility auf eine sequentielle Datei geschrieben. Diese Datei wird dann mit einem Ladeutility zurückgeladen.

```
SELECT
 COALESCE(Auftrag_Upd.Schl , Auftrag_Prod.Schl)
 AS Schl
,COALESCE(Auftrag_Upd.Daten, Auftrag_Prod.Daten)
 AS Daten
FROM Auftrag_Upd FULL OUTER JOIN Auftrag_Prod
     ON Auftrag_Upd.Schl = Auftrag_Prod.Schl
ORDER BY Schl
;
```

```
Schl         Daten
------------ ----------
1            ALT
2            NEU
3            NEU
```

Der Optimizer macht die Arbeit (sortiertes Entladen). Auslesen der neuen Daten als Zeichenkette. Diese Daten werden dann mit einem Ladeutility zurückgeladen.

```
SELECT
CAST(COALESCE(Auftrag_Upd.Schl , Auftrag_Prod.Schl)
as CHAR(15))
 || ';' ||
CAST(COALESCE(Auftrag_Upd.Daten, Auftrag_Prod.Daten)
as CHAR(15))
AS zeichenkette
FROM Auftrag_Upd FULL OUTER JOIN Auftrag_Prod
     ON Auftrag_Upd.Schl = Auftrag_Prod.Schl
ORDER BY zeichenkette
;
zeichenkette
------------------------------
1              ;ALT
2              ;NEU
3              ;NEU

drop table auftrag_prod;
drop table auftrag_upd;
create table auftrag_prod
(
 schl   integer  not null
,daten  char(10) not null
,primary key (schl)
)
;
create table auftrag_upd
(
 schl   integer  not null
,daten  char(10) not null
,primary key (schl)
)
;
insert into auftrag_prod (schl,daten)
            values        (1  , 'ALT');
insert into auftrag_prod (schl,daten)
            values        (2  , 'ALT');
insert into auftrag_upd (schl,daten)
            values        (2  , 'NEU');
insert into auftrag_upd (schl,daten)
            values        (3  , 'NEU');
select * from auftrag_prod
;
schl        daten
----------- ----------
1           ALT
2           ALT
select * from auftrag_upd
;
schl        daten
----------- ----------
2           NEU
3           NEU
```

6.6 Transaktion, COMMIT und ROLLBACK, SAVEPOINT

Unter der Voraussetzung „autocommit-modus aus“:

Oracle

```
SET AUTOCOMMIT OFF;
```

SQL Server

```
SET IMPLICIT_TRANSACTIONS ON;
```

DB2 for Linux UNIX and Windows

```
UPDATE COMMAND OPTIONS USING C OFF;
```

Was bewirken die folgenden Anweisungen?

```
SELECT * FROM Tbuch;
UPDATE Tbuch SET Preis = 0 WHERE Erschj = 1990;
SELECT * FROM Tbuch;
--sind an dieser Stelle innerhalb der Transaktion
--noch die alten oder schon die neuen Daten sichtbar?
ROLLBACK;

SELECT * FROM Tbuch;
COMMIT;
```

Oracle

SET AUTOCOMMIT ON bedeutet: jede INSERT-, UPDATE-, DELETE-Anweisung oder PL/SQL Block löst einen COMMIT aus und beendet die Transaktion.

Unter der Voraussetzung **SET AUTOCOMMIT OFF** muss eine Transaktion mit COMMIT oder ROLLBACK beendet!

SQL Server

`SET IMPLICIT_TRANSACTIONS ON`

Empfehlung: Vermeiden Sie bei interaktivem Arbeiten und bei der Skriptverarbeitung von Batches die Anweisung `BEGIN TRANSACTION`! Benutzen Sie `BEGIN TRANSACTION`, sofern nötig, nur innerhalb von Stored Procedures.

DB2 for Windows, UNIX, Linux

```
UPDATE COMMAND OPTIONS USING C OFF;
```

DB2 for z/OS

SPUFI kennt kein „autocommit modus ein“. Ein SPUFI-Skript wird immer mit „autocommit modus aus“ abgearbeitet.

„AUTOCOMMIT ===> YES (Y/N Commit after successful run?)” auf dem Spufi-Panel bedeutet, dass nach einer erfolgreichen Verarbeitung des Skripts (d. h. aller SQL-Anweisungen) die Transaktion mit einem COMMIT beendet wird.

Innerhalb einer Transaktion können während der Ausführung vom Anwender Savepoints gesetzt werden. Wenn nötig kann auf einen solchen **SAVEPOINT** zurückgesetzt werden.

```
select * from tbuch;
SAVEPOINT save01;                                --Oracle
--SAVE TRANSACTION save01;                       --SQL Server
--SAVEPOINT save01 ON ROLLBACK RETAIN CURSORS;--DB2
update tbuch set preis = preis + 10.00;
select * from tbuch;
ROLLBACK TO SAVEPOINT save01;                     --Oracle, DB2
--ROLLBACK TRANSACTION save01;                    --SQL Server
select * from tbuch;
commit;
```

6.7 Testen der Referentiellen Integrität

```
--auf001
-- diese Zeile ist Kommentar
SELECT * FROM Tautor;
UPDATE Tautor
    SET Autor = 'ECCO'
    WHERE Tautor.Autornr = 3;
SELECT * FROM Tautor;
ROLLBACK;
SELECT * FROM Tautor;
-- COMMIT;

--auf002
update tautor set  autornr = 2 where autornr = 1;
--sqlstate= 23505

--auf003
update tbuch set buchnr = 55 where buchnr = 5;
--sqlstate= 23504

--auf004
select * from tbuch;
select * from tvautor;
delete from tbuch where buchnr = 5;
select * from tbuch;
select * from tvautor;
rollback;
select * from tbuch;
select * from tvautor;

--auf005
delete from tautor where autornr = 20;
--sqlstate= 23504

--auf006
select * from tautor;
delete from tautor where autornr = 6;
select * from tautor;
rollback;
select * from tautor;
```

```
--auf007
update tvautor set buchnr = 11
where buchnr = 5 and autornr = 1;
rollback;

--auf008
update tvautor set buchnr = 6
where buchnr = 5 and autornr = 1;
--SQLSTATE=23505 (Buchnr,Lfdnr) ist UNIQUE

--auf009
update tvautor set autornr = 11
where buchnr = 9 and autornr = 10;
--sqlstate= 23505

--auf010
update tvautor set autornr = 55
where buchnr = 9 and autornr = 10;
--sqlstate= 23503

--auf011
insert into tvautor
(buchnr, autornr , lfdnr)
values ( 11, 1, 0);
rollback;

--auf012
insert into tvautor
(buchnr, autornr , lfdnr)
values ( 11, 55, 0);
--sqlstate= 23503

--auf013
insert into tvautor
(buchnr, autornr , lfdnr) values
( 11, NULL, 0);
--sqlstate= 23502
```

6.8 INSERT, UPDATE, MERGE und Subqueries

Eine Subquery ist (syntaktisch betrachtet) vereinfacht gesagt eine Select-From-Where-Anfrage in Klammern.

Eine Subquery, die einen Skalarwert zurückgibt, ist auch eine Expression!

Mit gewissen Einschränkungen kann eine Expression überall auftreten, wo auch ein Literal des entsprechenden Typs möglich ist: in der Select-Klausel, in der Where-Klausel bzw. Having-Klausel,

- **in der Set-Klausel beim UPDATE,**
- **in der Set-Klausel beim MERGE,**
- beim INSERT.

6.8.1 UPDATE und skalare Subquery

```
update tbuch
set preis =  (select sum(praemie) from tvautor);

update tbuch
set preis =
    (select sum(xyz.praemie) from tvautor xyz
     where xyz.buchnr = tbuch.buchnr)
where tbuch.buchnr in
    (select efg.buchnr from tvautor efg);
```

Oracle und DB2 und SQL Server, es funktioniert!

aber

SQL Server

```
update tbuch abc
set preis =
    (select sum(xyz.praemie) from tvautor xyz
     where xyz.buchnr = abc.buchnr);

Server: Nachr.-Nr. 170, Schweregrad 15, Status 1,
Zeile 1
Zeile 1: Falsche Syntax in der Nähe von 'abc'.
```

Für Tbuch darf bei SQL Server kein Korrelationsname verwendet werden!

6.8.2 UPDATE und row Subquery

```
update tbuch
set (erschj, preis) = (2002 , 66.13);
```

DB2, es funktioniert, aber nicht bei SQL Server und Oracle.

```
update tbuch
set (erschj , preis ) =
     ( select max(erschj), max(preis)
       from tbuch  );
```

Oracle und DB2, es funktioniert, aber nicht bei SQL Server:

6.8.3 INSERT und skalare Subquery

```
insert into tbuch (buchnr, titel, preis)
values (7777, 'abc',
         (select avg(preis) from tbuch) );
```

aber bei SQL Server 2005 funktionierte es noch nicht.

6.9 Rückgabe von Daten bei INSERT, UPDATE, DELETE

6.9.1 SQL Server: INSERT/UPDATE/DELETE …OUTPUT INSERTED, DELETED INTO…

```
update tbuch set preis=preis + 10.00
OUTPUT
       INSERTED.buchnr as buchbr
      ,INSERTED.preis  as preisneu
      ,DELETED.preis   as  preisalt
where buchnr in (5, 6)
;
delete from tbuch
OUTPUT
       DELETED.*
where buchnr in (5,6 )
;
select * from Tbuch
;
insert into Tbuch (buchnr, titel)
OUTPUT
       INSERTED.buchnr
      ,inserted.titel
      ,inserted.preis
       values ( 5, 'xxAnsichten eines Clowns')
             ,( 6, 'xxdie Blechtrommel')
;
select * from Tbuch
;
```

```
use sample
go
drop table dbo.t123
;
create table dbo.t123
(
 schl integer   not null
,daten char(10) not null
,primary key (schl)
)
;
insert into
  dbo.t123 (schl, daten)
values (1, 'AAAA')
;
insert into
  dbo.t123 (schl, daten)
values (2, 'BBBB')
;
select * from dbo.t123
;
-------------------
declare @t123var TABLE
(
  idt123var integer  not null IDENTITY(1,1)
 ,schl      integer  not null
, daten     char(10) not null
,primary key (idt123var)
)
;
insert into
dbo.t123 (schl, daten)
   OUTPUT  INSERTED.schl,INSERTED.daten
     INTO  @t123var(schl, daten)
values (3, 'CCCC')
;
select * from @t123var
;
-------------------
delete from dbo.t123
  OUTPUT DELETED.schl, DELETED.daten
    INTO @t123var(schl, daten)
where  schl = 3
;
select * from @t123var
;
-------------------
declare @t123varaltneu TABLE
(
 idt123varaltneu integer not null IDENTITY(1,1)
,schlalt integer not null, datenalt char(10) not null
,schlneu integer not null, datenneu char(10) not null
)
;
select * from dbo.t123
;
update dbo.t123
SET dbo.t123.daten = 'A1A1A1', dbo.t123.schl = 111
     OUTPUT DELETED.schl , DELETED.daten
           ,INSERTED.schl, INSERTED.daten
       INTO @t123varaltneu
           (schlalt, datenalt
           ,schlneu, datenneu )
where schl = 1
;
select * from dbo.t123;
select * from @t123varaltneu
;
-------------------
go
```

6.9.2 Oracle: RETURNING INTO Clause

```
SET SERVEROUTPUT ON SIZE UNLIMITED;
set LINESIZE 200;
set null null;
SET AUTOCOMMIT OFF;

select buchnr, preis, titel
from tbuch where buchnr = 5
;

<<bbb>>DECLARE
  v_buchnr   NUMBER;
  v_preis    NUMBER(7,2);
  v_titel    VARCHAR2(127);
BEGin
 bbb.v_buchnr := 5;

 UPDATE tbuch
 SET preis    = preis + 10.00
   , titel    =  '* ' || titel || ' *'
 WHERE buchnr = bbb.v_buchnr
 RETURNING preis, titel
 INTO    bbb.v_preis ,bbb.v_titel
;

DBMS_OUTPUT.PUT_LINE('preis: ' || bbb.v_preis)
;
DBMS_OUTPUT.PUT_LINE('titel: ' || bbb.v_titel)
;
rollback
;
DBMS_OUTPUT.PUT_LINE('nach rollback');
END
;
/
select buchnr, preis, titel
from tbuch where buchnr = 5
;
```

“The returning clause specifies the values return from DELETE, EXECUTE IMMEDIATE, INSERT, and UPDATE statements. You can retrieve the column values into individual variables or into collections.”

6.9.3 DB2: SELECT * FROM FINAL/OLD TABLE

```
SELECT  * FROM FINAL TABLE
(
 INSERT INTO tisbn (buchnr, isbn, lfdnr)
            VALUES (27 , 'yyyyy', 1)
)
;
BUCHNR      ISBN       LFDNR
----------- ---------- -----
         27 yyyyy          1,

  1 Satz/Sätze ausgewählt.

SELECT  buchnr, preis FROM FINAL TABLE
(
 UPDATE tbuch set preis = preis + 10
 Where buchnr IN (4, 5, 6 )
)
;
BUCHNR      PREIS
----------- ---------
          5     13,50
          6     30,50

  2 Satz/Sätze ausgewählt.


SELECT  * FROM OLD TABLE
(
 DELETE from tisbn where buchnr = 8
)
;
BUCHNR      ISBN       LFDNR
----------- ---------- -----
          8 3472864303    1,
          8 34728643yx    2,

  2 Satz/Sätze ausgewählt.



SELECT  * FROM FINAL TABLE
(
 MERGE …
)
;
```

6.10 Die ACID Eigenschaften einer Transaktion

Anfang der Transaktion

Girokonto +100

Sparkonto -100

Ende der Transaktion

Eine Transaktion ist eine logische Verarbeitungseinheit von Daten, alle Veränderungen oder keine muss durchgeführt werden.

Die Daten werden von einem konsistenten Zustand in einen neuen konsistenten Zustand überführt. **Daten sind konsistent, wenn sie alle im System definierten Integritätsbedingungen (d.h. CONSTRAINTs) erfüllen**. Die Verantwortung für die Konsistenz der Daten liegt beim System.

Die Daten sollten natürlich von einem korrekten Zustand in einen neuen korrekten Zustand überführt werden. Daten sind korrekt, wenn sie die Realität richtig beschreiben. Die Verantwortung für die Korrektheit der Daten liegt beim Endanwender.

ACID Eigenschaften einer Transaktion

- **Atomicity Abgeschlossenheit „alles oder nichts“**
- **Consistency Konsistenzerhaltung**
- **Isolation Abgrenzung**
- **Durability Dauerhaftigkeit**

Eine Transaktion wird entweder durch COMMIT oder durch ROLLBACK beendet.

- *Die Eigenschaft Atomicity* ist, was eine einzelne SQL-Anweisung betrifft, vom System garantiert! ***Die Eigenschaft Atomicity ist aber, was eine Transaktion mit mehreren SQL-Anweisungen betrifft, ohne besondere Bemühungen des Endanwenders nicht gegeben!***
- *Die Eigenschaft Consistency wird vom System garantiert.*
- *Die Eigenschaft Isolation (konkurrierende Transaktionen beeinflussen sich gegenseitig nicht) ist eher eine erwünschte als eine garantierte Eigenschaft. Die Systeme stellen Mechanismen zur Verfügung – nämlich Isolation Levels kleiner als das Maximum – deren Effekt genau darin besteht, Isolation zu untergraben. Implementiert werden die verschiedenen Levels mit Hilfe von Sperrprotokollen bzw. Zeitstempelverfahren.* ***Abhängig von der Komplexität der SQL-Anweisungen (SELECT, INSERT, UPDATE, DELETE, MERGE) kann zur Verbesserung der Performance ein schwächerer Isolation Level als der Maximale gewählt werden, ohne dass die Konsistenz und Korrektheit der Daten beeinträchtigt wird.***
- *Die Eigenschaft Durability wird vom System garantiert.*

Auszug aus Wikipedia Oktober 2009:

ACID, deutsch auch *AKID*, ist ein Akronym in der Informatik. Es beschreibt erwünschte Eigenschaften von Transaktionen in Datenbankmanagement-.systemen (DBMS) und verteilten Systemen. Es steht für *atomicity*, *consistency*, *isolation* und *durability*. Man spricht im Deutschen auch von *AKID-Eigenschaften* (*Atomarität*, *Konsistenz*, *Isoliertheit* und *Dauerhaftigkeit*). Sie gelten als Voraussetzung für die Verlässlichkeit von Systemen. Das Akronym ACID zur Charakterisierung von Transaktionen wurde 1983 von den Informatikern Theo Härder und Andreas Reuter geprägt[1].

Atomarität

Von einer atomaren Operation spricht man, wenn die Transaktion entweder ganz oder gar nicht ausgeführt wird: Das DBMS verhält sich gegenüber dem Benutzer so, als ob die Transaktion eine einzelne elementare Operation wäre, die nicht von anderen Operationen unterbrochen werden kann. Praktisch werden die einzelnen Datenbankanweisungen, aus der sich die Transaktion zusammensetzt, natürlich nacheinander ausgeführt – sobald sich jedoch herausstellt, dass die Transaktion nicht abgeschlossen werden kann, wird ein Rollback durchgeführt, also alles bisher Erledigte wieder rückgängig gemacht.

Konsistenzerhaltung

Konsistenz heißt, dass eine Transaktion nach Beendigung einen konsistenten Datenzustand hinterlässt, falls die Datenbank vor der Transaktion auch konsistent war. Nach Beendigung der Transaktion gelten die inhärenten und explizit definierten Integritätsbedingungen, insbesondere die Schlüssel- und Fremdschlüsselbedingungen.

Isolation

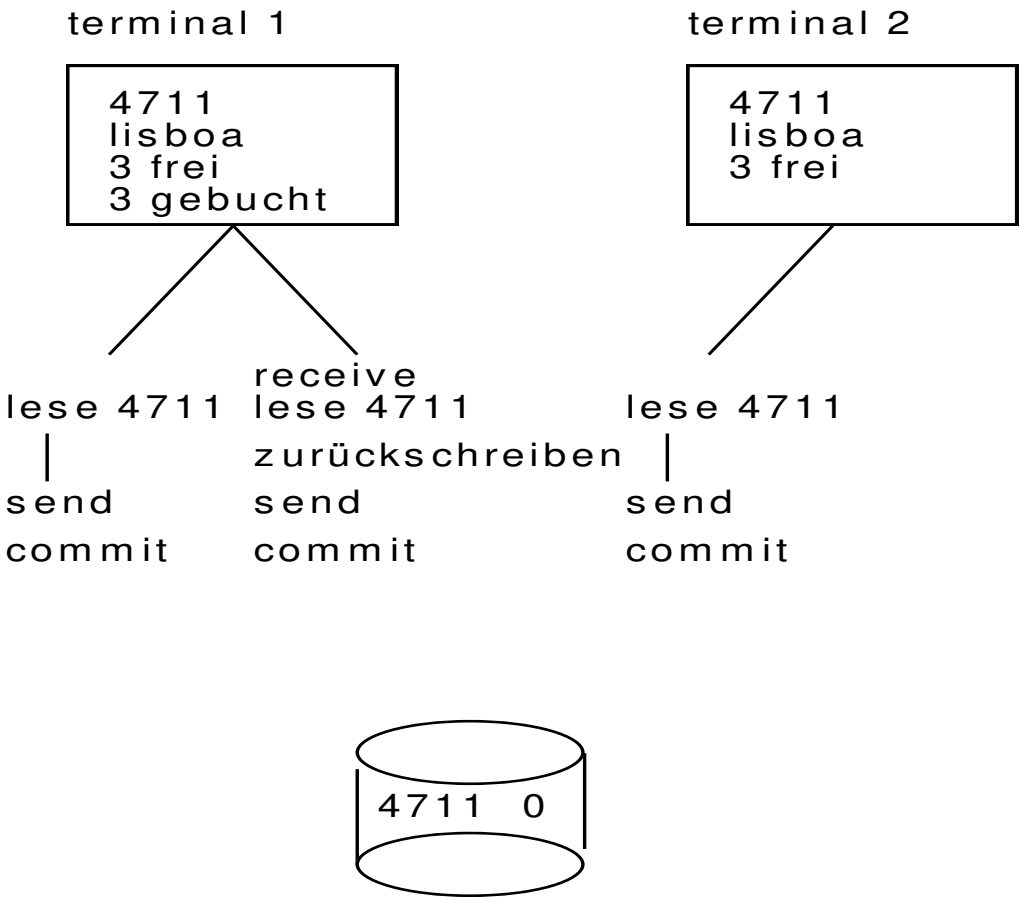
Durch das Prinzip der Isolation wird verhindert, dass sich in Ausführung befindliche Transaktionen gegenseitig beeinflussen. Realisiert wird dies beispielsweise durch spezielle Sperrprotokolle oder Zeitstempelverfahren.

Dauerhaftigkeit

Das Ergebnis einer Transaktion ist dauerhaft: Die Wirkung einer erfolgreich abgeschlossenen Transaktion bleibt dauerhaft (persistent) in der Datenbank erhalten, insbesondere auch nach Systemabstürzen.

6.11 die zwei Sperrprobleme

Leider ausgebucht!



Wir können zwei Sperrprobleme identifizieren:

Das Sperren von Daten über Dialogschritte hinweg, transaktionsübergreifend. Das Sperren über Dialogschritte hinweg ist eine Aufgabe der Anwendungen und wird hier nicht weiter diskutiert.

Das Sperren von Daten während einer Transaktion.

6.12 Dirty Read – der schmutzige Read

Trans1	Zeitpunkt	Trans2
	t1	**UPDATE** **X-Lock** **exklusive Sperre** **X-Lock**
Oracle Ein schmutziges Lesen ist nicht möglich!		**X-Lock** **X-Lock**

SQL Server
`SET IMPLICIT_TRANSACTIONS ON;`
`SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;`
 	
`SELECT ...`
 	
DB2 for Windows, UNIX, Linux
`UPDATE COMMAND OPTIONS USING C OFF;`
`CHANGE ISOLATION TO UR;`
 	
`SELECT ...`
`SELECT ... WITH UR;`
	
	t2
	
	t2
	t2
		X-Lock
		
		X-Lock

DB2 for z/OS SPUFI **„autocommit off“** `SELECT ... WITH UR;`	 t2	**X-Lock** **X-Lock**
	t3	**ROLLBACK**

"Transaktion Trans1 hat zum Zeitpunkt t2 Zugang zu nicht festgeschriebenen Daten"

Parameter für Trans1 mit dem Ziel den Dirty Read zu vermeiden:

Oracle:

Bei Oracle ist auf Grund der “statement-level read consistency“ ein schmutziges Lesen nicht möglich!

SQL Server:

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET TRANSACTION ISOLATION LEVEL SNAPSHOT;
```

DB2 for Windows, UNIX, Linux:

```
CHANGE ISOLATION TO CS;
SELECT ... WITH CS;
```

DB2 for z/OS SPUFI:

“ISOLATION CS im SPUFI Default Panel”

```
SELECT ... WITH CS;
```

6.13 Lost Update – der verlorene Update

Das folgende Coding SELECT/UPDATE ist abhängig vom Isolation Level.

Trans1	Zeitpunkt	Trans2

Oracle
`SET AUTOCOMMIT OFF;`
`SET TRANSACTION ISOLATION LEVEL READ COMMITTED;`
SQL Server
`SET IMPLICIT_TRANSACTIONS ON;`
`SET TRANSACTION ISOLATION LEVEL READ COMMITTED;`
DB2 for Windows, UNIX, Linux
`UPDATE COMMAND OPTIONS USING C OFF;`
`CHANGE ISOLATION TO CS;`
DB2 for z/OS SPUFI
„autocommit off“
“ISOLATION CS im SPUFI Default Panel”		

`SELECT konto`
`FROM tkonten`
`WHERE kontonr = 5`
es sind 3.50
Lesen ohne Sperren
	t1	

erhöhe um 200.00	t2	`SELECT konto`
		`FROM tkonten`
		`WHERE kontonr = 5`
		es sind 3.50
		Lesen ohne Sperren

`UPDATE tkonten SET`
`konto = 203.50`
`WHERE kontonr = 5`
Schreiben ohne Prüfen erfolgreich!
	t3	ziehe 50.00 ab

COMMIT	t4	

	t5	`UPDATE tkonten SET`
		`konto = - 46.50`
		`WHERE kontonr = 5`
		Schreiben ohne Prüfen erfolgreich!
		COMMIT

"Transaktion Trans1 verliert zum Zeitpunkt t5 einen Update"

Was ist das Problem? Lesen ohne Sperren und Schreiben ohne Prüfen!

Von allen Beteiligten muss entweder beim Lesen gesperrt werden oder beim Schreiben geprüft werden!

Parameter für alle Transaktionen mit dem Ziel, den verlorenen Update zu vermeiden:

Oracle (Schreiben mit Prüfen):

Bei SERIALIZABLE wird vom System beim Schreiben geprüft.

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
```

Die Klausel **SELECT…FOR UPDATE OF…** bietet die Möglichkeit, eine gelesene Zeile zu sperren, auch unter der Voraussetzung **SET AUTOCOMMIT ON**.

SQL Server (Schreiben mit Prüfen oder Lesen mit Sperren):

Bei SERIALIZABLE und REPEATABLE READ wird vom System gesperrt.

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
```

Bei SNAPSHOT wird vom System beim Schreiben geprüft.

```
SET TRANSACTION ISOLATION LEVEL SNAPSHOT;
```

Die Klausel SELECT…WITH(UPDLOCK) bietet die Möglichkeit, eine gelesene Zeile zu sperren.

DB2 (Lesen mit Sperren):

Bei RR und RS wird vom System gesperrt.

DB2 for Windows, UNIX, Linux:

```
CHANGE ISOLATION TO RR;
SELECT ... WITH RR;
CHANGE ISOLATION TO RS;
SELECT ... WITH RS;
```

DB2 for z/OS SPUFI:

"ISOLATION RR im SPUFI Default Panel"
"ISOLATION RS im SPUFI Default Panel"

```
SELECT ... WITH RR;
SELECT ... WITH RS;
```

Die Klausel SELECT…**FOR UPDATE OF…WITH RS** bietet die Möglichkeit, eine gelesene Zeile zu sperren.
ACHTUNG: *BEI* ***SELECT…FOR UPDATE…OF WITH CS*** *WIRD DIE ZEILE* ***NICHT GESPERRT****!*

Schreiben mit Prüfen im Coding, unabhängig vom Isolation Level

```
UPDATE  tkonten SET konto = 203.50
WHERE kontonr = 5 AND konto = 3.50;
```

Der verlorene Update ist nicht möglich, wenn eine andere, logisch nicht äquivalente Update-Anweisung benutzt wird.

```
UPDATE  tkonten SET konto = konto + 200.00
WHERE kontonr = 5;
```

Weitere Details (CURSOR FOR UPDATE, OPEN, FETCH, UPDATE WHERE CURRENT) sind für die Anwendungsentwicklung wichtig: wie müssen Update-Programme geschrieben werden, damit der LOST UPDATE nicht möglich ist!

6.14 Nonrepeatable Read und die nicht korrekte Analyse

KONTO 1 = 40, KONTO 2 = 50, KONTO 3 = 30

SELECT SUM(Kontostand) FROM Tkonten;

Trans1	Zeitpunkt	Trans2
Oracle SET AUTOCOMMIT OFF; SET TRANSACTION ISOLATION LEVEL READ COMMITTED; **SQL Server** SET IMPLICIT_TRANSACTIONS ON; SET TRANSACTION ISOLATION LEVEL READ COMMITTED; **DB2 for Windows, UNIX, Linux** UPDATE COMMAND OPTIONS USING C OFF; CHANGE ISOLATION TO CS; **DB2 for z/OS SPUFI** **„autocommit off“** “ISOLATION CS im SPUFI Default Panel”		
Lese KONTO 1 (40) SUMME = 40	t1	
Lese KONTO 2 (50) SUMME = 90	t2	
	t3	Update KONTO 3 (30 -> 20)
	t4	Update KONTO 1 (40 -> 50)
	t5	COMMIT
Lese KONTO 3(20) SUMME = 110	t6	

"Transaktion Trans1 kommt zu einem schlechten Ergebnis, es gab nie diese Summe auf den Konten!"

"Das Lesen der Konten ist nicht wiederholbar“.

Sofern die Select-Anweisung alleine auf der Table ist, liefert die Anweisung ein korrektes Ergebnis.

Parameter für Trans1 mit dem Ziel, diese nicht korrekte Analyse zu vermeiden:

Oracle:

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
```

Oracle hat bei `READ COMMITTED` auf Grund seiner “statement-level read consistency“ kein Problem mit der falschen Analyse.

SQL Server:

Es muss mit SNAPSHOT oder mindestens mit REPEATABLE READ gearbeitet werden.

```
SET TRANSACTION ISOLATION LEVEL SNAPSHOT
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
```

DB2 for Windows, UNIX, Linux:

Es muss mindestens mit RS gearbeitet werden.

```
CHANGE ISOLATION TO RS;
SELECT ... WITH RS;
CHANGE ISOLATION TO RR;
SELECT ... WITH RR;
```

DB2 for z/OS SPUFI:

Es muss mindestens mit RS gearbeitet werden.

ISOLATION RS im SPUFI Default Panel bzw. RS auf Anweisungsebene

```
SELECT ... WITH RS;
```

ISOLATION RR im SPUFI Default Panel bzw. RR auf Anweisungsebene

```
SELECT ... WITH RR;
```

6.15 Der Isolation Level und der SQL Standard

Der Isolation Level und der SQL Standard	dirty read	nonrepeatable read	phantom read
READ UNCOMMITTED	Y	Y	Y
READ COMMITTED	N	Y	Y
REPEATABLE READ	N	N	Y
SERIALIZABLE	N	N	N

Ein Isolation Level ist definiert durch die Phänomene, die sich in einer Transaktion nicht ereignen dürfen.

Verpflichtung der Implementierungen:

Bei ISOLATION LEVEL SERIALIZABLE muss – laut SQL Standard – die erfolgreiche Ausführung von konkurrierenden Transaktionen einer seriellen Ausführung entsprechen. The execution of concurrent SQL-transactions at isolation level SERIALIZABLE is guaranteed to be serializable.

6.16 “No updates will be lost.” Diese Behauptung ist Wunschdenken.

„‘The four isolation levels guarantee that [...] no updates will be lost.‘ Diese Behauptung ist Wunschdenken: Die Definitionen der vier Isolationsgrade an sich bieten keine solche Garantie.“

Das Zitat stammt aus: SQL – Der Standard SQL/92 mit den Erweiterungen CLI und PSM Deutsche Ausgabe des amerikanischen Klassikers Ausblick auf SQL3 Chris J. Date Hugh Darwen Addison Wesley Longman GmbH, 1998 Seite 87

Implementierungen

Isolation Level SQL Server	Isolation Level DB2	Isolation Level ORACLE
READ UNCOMMITTED ### !!!	UR uncommitted read ### !!!	
READ COMMITTED ### !!!	CS cursor stability ### !!!	READ COMMITTED !!!
REPEATABLE READ ###	RS read stability ###	
SERIALIZABLE	RR repeatable read	
		SERIALIZABLE ???
SNAPSHOT		

!!! **Achtung:** Der Lost Update ist bei READ UNCOMMITTED / UR bzw. READ COMMITTED / CS möglich (bei schlechtem Coding).

Achtung: Bei diesem ISOLATION LEVEL ist **nicht garantiert**, dass eine SELECT-Anweisung ein korrektes Ergebnis liefert.

??? Achtung: Bei ORACLE's SERIALIZABLE ist die Forderung fully serializable execution nicht erfüllt.

Beachten Sie bitte:

SQL Server READ UNCOMMITTED **entspricht** DB2 UR

SQL Server READ COMMITTED **entspricht** DB2 CS

SQL Server REPEATABLE READ **entspricht** DB2 RS

SQL Server SERIALIZABLE **entspricht** DB2 RR

Beachten Sie bitte:

SQL Server SERIALIZABLE und DB2 RR sind etwas anderes als ORACLE „SERIALIZABLE“

6.17 ISOLATION LEVEL SERIALIZABLE

In einem Datenbankmanagementsystem werden dieselben Daten "gleichzeitig" von verschiedenen Transaktionen bearbeitet.

Die verzahnte Ausführung konkurrierender Transaktionen sollte einer seriellen Ausführung dieser Transaktionen nacheinander entsprechen.

Dies sollte laut SQL Standard garantiert sein, wenn **alle Transaktionen** im ISOLATION LEVEL SERIALIZABLE ausgeführt werden.

SQL Server garantiert!
Der Isolation Level SERIALIZABLE des SQL Servers garantiert serialisierbar.

```
SET IMPLICIT_TRANSACTIONS ON;
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
```

DB2 garantiert!
IBM's Isolation Level RR garantiert serialisierbar (Beispiel für DB2 for Windows, UNIX, Linux).

```
UPDATE COMMAND OPTIONS USING C OFF;
CHANGE ISOLATION TO RR;
```

Oracle garantiert nicht!
Der sogenannte „Isolation Level Serializable" von Oracle garantiert das nicht.

```
SET AUTOCOMMIT OFF;
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
```

Da die Datenbanksysteme am Markt nur eine Illusion von sequentieller Ausführung aufrechterhalten ohne tatsächlich alle Transaktionen nacheinander einzeln abzuarbeiten, kann es vorkommen, dass eine Transaktion vom System abgebrochen werden muss. Ein Programm, dessen Transaktionen mit **Isolation Level Serializable** arbeiten, muss daher mit **Serialisationsfehlern** (Timeout, Deadlock) umgehen können.

Wenn irgendeine Transaktion aber in einem niedrigeren ISOLATION LEVEL ausgeführt wird, kann die Serialisierbarkeit durch eine Reihe verschiedener Möglichkeiten verletzt werden.

- Phantom Read – The Incorrect Analysis Problem
- Nonrepeatable Read – The Incorrect Analysis Problem
- Dirty Read – The Uncommitted Dependency Problem
- ***The Lost Update Problem: "Ein Änderung geht verloren".***
- **Dirty Update – The Uncommitted Dependency Problem**
- **A*chtung: Der Dirty Update ist bei DB2, Oracle, SQL Server nicht möglich.***

6.17.1 ORACLE's ISOLATION LEVEL SERIALIZABLE „garantiert serialisierbar nicht“

Trans1	Zeitpunkt	Trans2

Oracle
`SET AUTOCOMMIT OFF;`
`SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;`		

Lese PK = 5 gefunden	t1	
	t2	Lese FK=5 nicht gefunden
INSERT FK=5 COMMIT	t3	
	t4	DELETE PK=5

Wir nehmen an, dass die Primärschlüssel/Fremdschlüsselbeziehung nicht implementiert ist!

Selbst wenn beide Transaktionen mit Oracle’s ISOLATION LEVEL SERIALIZABLE arbeiten entspricht das Ergebnis keinem serialisierten Ablauf!

6.17.2 SQL Server's ISOLATION LEVEL SERIALIZABLE „garantiert serialisierbar“

Trans1
SQL Server
`SET IMPLICIT_TRANSACTIONS ON;`
`SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;`
	Zeitpunkt	**Trans2**
		SQL Server
		`SET IMPLICIT_TRANSACTIONS ON;`
		`SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;`

Lese PK = 5 gefunden S-Lock	t1	
	t2	Lese FK=5 nicht gefunden
INSERT FK=5 Wait	t3	
Wait Wait Wait	t4	 DELETE PK=5 Wait

Wir nehmen an, dass die Primärschlüssel/Fremdschlüsselbeziehung nicht implementiert ist!

Wenn beide Transaktionen mit SQL Server ISOLATION LEVEL SERIALIZABLE arbeiten entspricht das Ergebnis einem serialisierten Ablauf!

Es kommt zum Deadlock, eine der beiden Anweisungen INSERT bzw. DELETE wird nicht erfolgreich ausgeführt.

SQL Server ISOLATION LEVEL SNAPSHOT

Mit Hilfe von ALTER DATABASE kann die Snapshotisolation aktiviert werden.

```
ALTER DATABASE dbbuecher
    SET ALLOW_SNAPSHOT_ISOLATION ON;
```

Beachten Sie bitte, dass in der Sekundärliteratur Mythen verbreitet werden der Art: „SQL Server adds SNAPSHOT isolation that, in effect, provides alternate implementations of SERIALIZABLE ...“.

Wenn beide Transaktionen mit SQL Server ISOLATION LEVEL SNAPSHOT arbeiten entspricht das Ergebnis **keinem serialisierten Ablauf**!

Vergleichen Sie dazu bitte auch die Diskussion zu den Besonderheiten von ORACLE’s Implementierung.

6.17.3 DB2's Isolation Level RR „garantiert serialisierbar“

Trans1	Zeitpunkt	Trans2

DB2 for Windows, UNIX, Linux
`UPDATE COMMAND OPTIONS USING C OFF;`
`CHANGE ISOLATION TO RR;`
		DB2 for Windows, UNIX, Linux
		`UPDATE COMMAND OPTIONS USING C OFF;`
		`CHANGE ISOLATION TO RR;`

Lese PK = 5 gefunden S-Lock	t1	
	t2	Lese FK=5 nicht gefunden
INSERT FK=5 Wait	t3	
Wait Wait Wait	t4	 DELETE PK=5 Wait

Wir nehmen an, dass die Primärschlüssel/Fremdschlüsselbeziehung nicht implementiert ist!

Wenn beide Transaktionen mit DB2's Isolation Level RR arbeiten entspricht das Ergebnis einem serialisierten Ablauf!

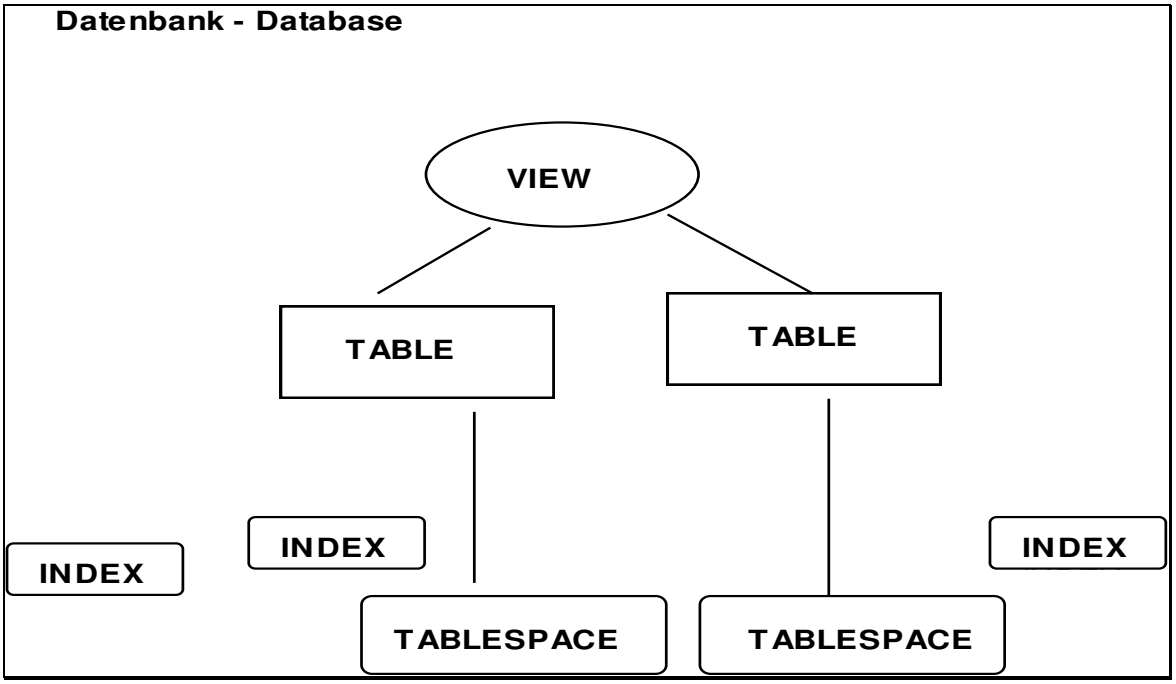
Es kommt zum Deadlock, eine der beiden Anweisungen INSERT bzw. DELETE wird nicht erfolgreich ausgeführt.

7

Datendefinition VIEW, der Katalog, Datenschutz, USER und ROLE

7 Datendefinition VIEW, der Katalog, Datenschutz, USER und ROLE

7.1 Externe Ebene und VIEW



7.2 CREATE VIEW

```
CREATE VIEW V9091
AS
      SELECT
         tbuch.Erschj AS Jahr
       , tbuch.Titel  AS Titel
       , tbuch.Buchnr AS Buchnr
       FROM   Tbuch
       WHERE  tbuch.Erschj = 1990
          OR  tbuch.Erschj = 1991
--WITH CHECK OPTION -- ist nicht default
;
SELECT   V9091.Buchnr, V9091.Jahr, V9091.Titel
FROM     V9091
ORDER BY V9091.Buchnr
;
    BUCHNR       JAHR TITEL
---------- ---------- --------------------
         9       1990 DB2 fuer Sie
        11       1990 Elvis in Heidelberg
```

- **Bei der Definition einer VIEW kann der Schlüssel nicht explizit definiert werden.** Es liegt in der Verantwortung des Anwenders, die zugrundeliegende SELECT-Anweisung so zu formulieren, dass die VIEW keine Duplikate präsentiert (im Beispiel ist die Spalte Buchnr der Schlüssel der VIEW V9091).
- **Es entsteht beim CREATE VIEW kein Duplikat der Daten!**
- **Die SELECT-Anweisung gegen die View wird mit Hilfe der Viewdefinition umgesetzt.**
- Das Datentyp der Spalten einer VIEW bestimmt sich aus dem Datentyp der Spalten des SELECT-Teils .
- Der Anwender kann nicht erkennen, ob er den SELECT auf eine TABLE oder auf eine VIEW absetzt.
- Eine VIEW besitzt für ihre Spalten auch Spaltennamen. Werden beim CREATE VIEW keine Spaltennamen angegeben, so werden die Namen der Spalten aus dem zugrunde liegenden SELECT-Teil verwendet. Spaltennamen müssen für alle Spalten angegeben werden, falls eine Funktion oder ein arithmetischer Ausdruck oder Konstanten verwendet werden.

  empfohlene Schreibweise:

```
CREATE VIEW Vbrutto
AS
  SELECT  tbuch.Buchnr       AS buchnr
        , tbuch.Titel        AS titel
        , tbuch.Preis * 1.15 AS bruttopreis
  FROM Tbuch
;
```

  nicht empfohlene Schreibweise:

```
CREATE VIEW Vbrutto (buchnr, titel, bruttopreis)
AS
  SELECT  Buchnr, Titel, Preis * 1.15
  FROM Tbuch
;
```

- **VIEWs können die Grundlage bilden zur Definition von weiteren VIEWs.**

```
CREATE VIEW Vxyz
AS
  SELECT
   Vbrutto.buchnr      as buchnummer
  ,vbrutto.bruttopreis as bruttowert
  FROM Vbrutto
;
```

7.3 VIEW und INSERT, UPDATE, DELETE

```
CREATE VIEW V9091
AS
      SELECT
         tbuch.Erschj AS Jahr
       , tbuch.Titel  AS Titel
       , tbuch.Buchnr AS Buchnr
       FROM   Tbuch
       WHERE  tbuch.Erschj = 1990
          OR  tbuch.Erschj = 1991
;
```

- INSERT über eine VIEW ist zum Beispiel nicht möglich, wenn die verborgenen Spalten mit NOT NULL definiert sind.
- INSERT über eine VIEW ist zum Beispiel möglich, wenn die verborgenen Spalten mit NOT NULL DEFAULT definiert sind.
- Ein INSERT mit folgenden Daten ist möglich! Die Zeile ist aber durch die View hindurch nicht sichtbar!

```
insert into v9091 (jahr, titel, buchnr)
values (1992, 'das verborgene Buch', 123)
;
```

7.4 CREATE VIEW ... WITH CHECK OPTION

```
CREATE VIEW V9091
AS
      SELECT
         tbuch.Erschj AS Jahr
       , tbuch.Titel  AS Titel
       , tbuch.Buchnr AS Buchnr
       FROM   Tbuch
       WHERE  tbuch.Erschj = 1990
          OR  tbuch.Erschj = 1991
WITH CHECK OPTION

;
```

- WITH CHECK OPTION: Bei INSERT bzw. UPDATE der VIEW wird vom System geprüft, ob die neuen Zeilen noch im Bereich der VIEW liegen.
- Über einen VIEW soll gewährleistet werden, dass eine Spalte nur bestimmte Werte enthalten darf:

```
CREATE VIEW ...
WHERE Geschlecht = 'w' OR Geschlecht = 'm'
WITH CHECK OPTION
;
```

7.5 VIEW Verwendung und Vorteile

Datenschutz

Der Endanwender kann nur die Daten sehen, für die er auf Grund der VIEW-definition berechtigt ist (Beispiel: alle Personalstammsätze mit Gehältern unter 5000 DM, ... über 5000 DM):

```
GRANT SELECT ON VIEW Vgehalt5000  TO PUBLIC;
GRANT SELECT ON VIEW Vgehalt10000 TO Sekretaer;
```

- horizontale Teilmengen (Zeile)
- vertikale Teilmengen (Spalte)
- row-and-column subset
- Statistiken mit Hilfe von Views
- VIEW für SELECT, VIEW für UPDATE
- VIEW with CHECK Option

die Handhabung der Datenbank wird erleichtert

Es ist nicht nötig, einen SELECT immer wieder einzugeben. Für immer wieder benötigte komplexe Abfragen wird eine VIEW definiert und getestet. Aus dieser VIEW können dann einzelne Zeilen selektiert werden. Der Endanwender benützt für komplexe Abfragen vordefinierte VIEWs.

logische Datenunabhängigkeit

Bei einer Änderung (Restrukturierung) der konzeptionellen Ebene muss nur die VIEW neu definiert werden. Die Anwendungsprogramme müssen nicht geändert werden (Wartungsfreundlichkeit).

7.6 Die Handhabung der Datenbank wird erleichtert

```
drop view buchautor1
;
create view buchautor1
as
  select  tautor.autornr AS autornr
         ,tautor.autor   AS autor1
         ,tbuch.buchnr   AS buchnr
         ,tbuch.titel    AS titel
  from tautor, tvautor, tbuch
  where tautor.autornr = tvautor.autornr
  and   tvautor.buchnr = tbuch.buchnr
  and   (tvautor.lfdnr = 0  OR tvautor.lfdnr = 1)
;
SELECT
       Autornr                    AS Autornr
     , SUBSTR    (Autor1, 1, 10)  AS Autor
--   , SUBSTRING(Autor1, 1, 10)   AS Autor
     , Buchnr                     AS Buchnr
     , SUBSTR   (Titel,  1, 6)    AS Titel
--   , SUBSTRING(Titel,  1, 6)    AS Titel
FROM buchautor1
WHERE UPPER(Autor1) LIKE 'BOELL%'
OR    LOWER(Titel)  LIKE '%der%'
;
```

Erstellen Sie die VIEW für folgende Sicht (keine Daten aus Tverlag):

```
select
titel, buchnr, isbn, ilnr, autor, autornr, alnr
from valles
order by
titel, buchnr, isbn, ilnr, autor, autornr, alnr
;
```

titel	buchnr	isbn	ilnr	autor	autornr	alnr
____	____	__________	____	Scheifele	6	____
a guide to db2	12	0201501139	1	C. J. Date	20	1
a guide to db2	12	0201501139	1	Colin J. White	21	2
Ansichten eines	5	__________	____	Boell	1	0
C	1	__________	____	BUSCH	100	1
C	1	__________	____	BUSCH	200	2
C	2	__________	____	BUSCH	200	0
Database System	18	__________	____	____	____	____
DB2 fuer Sie	9	__________	____	Emil Hack	10	1
DB2 fuer Sie	9	__________	____	Frieda Holz	11	2
der Butt	8	3472864303	1	Grass	2	0
der Butt	8	34728643yx	2	Grass	2	0
der Name der Ro	7	__________	____	Eco	3	0
die Blechtromme	6	__________	____	Grass	2	0
die Jüdin von T	27	__________	____	____	____	____
Elvis in Heidel	11	__________	____	____	____	____

```
drop view valles
;
Go
CREATE VIEW Valles
AS
  SELECT
   COALESCE ( SUBSTRING( Tbuch.Titel, 1, 15)
              , '____')            AS titel
  ,COALESCE ( CAST( Tbuch.Buchnr AS CHAR(11) )
              , '____' )           AS buchnr
  ,COALESCE ( Tisbn.isbn
              ,  '__________')     AS isbn
  ,COALESCE ( CAST( Tisbn.lfdnr AS CHAR(2) )
              , '____')            AS ilnr
  ,COALESCE ( SUBSTRING(Tautor.Autor, 1, 15)
              , '____')            AS autor
  ,COALESCE ( CAST( Tautor.Autornr AS CHAR(11) )
              , '____' )           AS autornr
  ,COALESCE ( CAST( Tvautor.Lfdnr AS CHAR(2) )
              , '____' )           As alnr
  FROM
   (( TBUCH LEFT OUTER JOIN TISBN
  ON TBUCH.BUCHNR = TISBN.BUCHNR
  )
  LEFT OUTER JOIN TVAUTOR
  ON TBUCH.BUCHNR = TVAUTOR.BUCHNR
  )
  FULL OUTER JOIN TAUTOR
  ON TVAUTOR.AUTORNR = TAUTOR.AUTORNR
  ;
```

7.7 VIEW und logische Datenunabhängigkeit

Tbuch

Buchnr	Erschj	Preis	Titel	Verlagnr

```
CREATE VIEW Vpreistitel
AS
SELECT
  tbuch.Buchnr, tbuch.Preis, tbuch.Titel
FROM Tbuch
;
```

Ttitel

Titelnr	Titel

Tbuch

Titelnr	Buchnr	Erschj	Preis	Verlagnr

```
CREATE VIEW Vpreistitel
AS
SELECT
  Tbuch.Buchnr, Tbuch.Preis, Ttitel.Titel
FROM   Tbuch, Ttitel
WHERE  Tbuch.Titelnr = Ttitel.Titelnr
;
```

Die neue VIEW ist theoretisch updatebar!

```
UPDATE Vpreistitel
SET Preis = 99.88
WHERE Buchnr = 5;
```

Microsofts SQL Server unterstützt solche Update-Operation.

Das folgende Beispiel wurde mit SQL Server getestet.

```
drop view vtbuchjointisbn
GO
create view vtbuchjointisbn
as
select
tbuch.buchnr
,tbuch.titel
,Tbuch.preis
,tisbn.isbn
,tisbn.lfdnr
from Tbuch inner join Tisbn
on Tbuch.Buchnr = Tisbn.Buchnr
;
go
select * from vtbuchjointisbn
;
update vtbuchjointisbn
set preis = preis +10
;
select * from vtbuchjointisbn
;
```

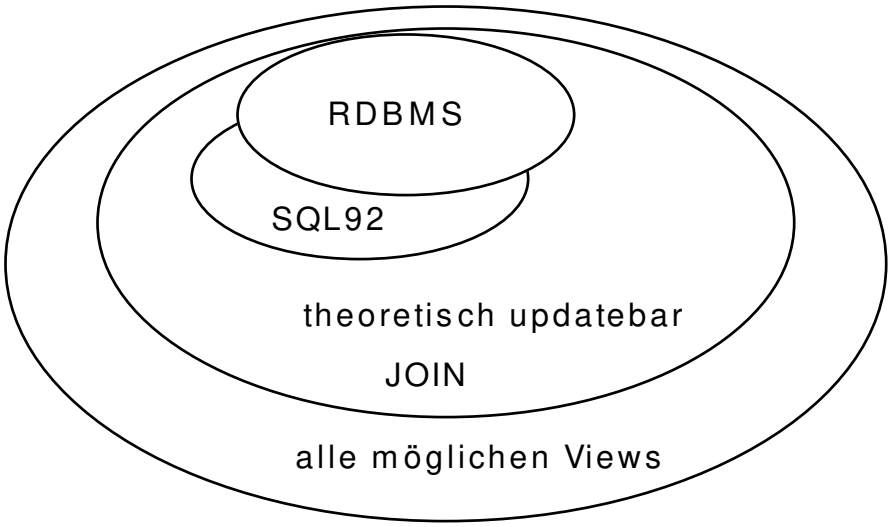
Sofern Ihr System solche Update-Operationen nicht unterstützt besteht die Alternative der **INSTEAD OFF-Trigger**. Der hauptsächliche Vorteil von INSTEAD OF-Triggern ist, dass sie das Aktualisieren von Sichten ermöglichen, die normalerweise nicht aktualisiert werden könnten.

7.8 VIEW und INSTEAD OFF-Trigger ein Beispiel

```
DROP trigger vtbuchjointisbn_trig1
GO
create trigger
 vtbuchjointisbn_trig1 ON vtbuchjointisbn
INSTEAD OF INSERT
AS
BEGIN
SET NOCOUNT ON
DECLARE @DBID INT
SELECT  @DBID = DB_ID()
DECLARE @DBNAME NVARCHAR(128)
SELECT  @DBNAME = DB_NAME()

print '>>>>>>trigger anfang'
IF (select count(*) from INSERTED )> 1
  begin
    PRINT ('bitte nur eine zeile')
    RAISERROR
    ( 'mehr als eine Zeile beim INSERT ist nicht erlaubt
     , die aktuelle Datenbank-Id ist: %d
     , der Name der Datenbank ist: %s.', 16, 1, @DBID, @DBNAME)
    return
  end
--
IF EXISTS( select tab1.buchnr, tab2.buchnr, tab2.ISBN
           from tbuch tab1, INSERTED tab2
           where tab1.buchnr = tab2.buchnr)
  Begin
    print 'wenigstens eine der buchnr war in tbuch schon
vorhanden'
    RAISERROR
    ( 'buchnr in tbuch schon vorhanden
      ,die aktuelle Datenbank-Id ist: %d
     , der Name der Datenbank ist: %s.'
     , 16, 1, @DBID, @DBNAME)
    return
  End
--
print '>>>>>>  insert tbuch und tisbn möglich'
insert into tbuch (buchnr, preis, titel)
    select buchnr, preis, titel
    FROM INSERTED
print '>>>>>>  insert tbuch'
insert into tisbn (buchnr, isbn, lfdnr)
    select buchnr, isbn, 1
    FROM INSERTED
print '>>>>>>  insert tisbn'

print '>>>>>>trigger ende'
END
Go
```

```
--Test des Triggers
SET implicit_transactions ON
SET NOCOUNT ON
GO
--select buchnr, preis, titel from tbuch
--select buchnr,preis,isbn, titel from vtbuchjointisbn
go

print '>>>1 insert vorher mehrere zeilen'
insert into vtbuchjointisbn (buchnr,titel, preis, isbn, lfdnr)
          select buchnr + 1000, titel, preis, 'AAA', 1   from
tbuch
print '>>>1 insert nachher'

print '>>>2 insert vorher schluessel schon vorhanden'
insert into vtbuchjointisbn (buchnr,titel, preis, isbn, lfdnr)
values (5, 'ABC-titel' , 12345.67, '3-isbn', 1)
print '>>>2 insert nachher'

print '>>>3 insert vorher schluessel noch nicht vorhanden'
insert into vtbuchjointisbn (buchnr,titel, preis, isbn, lfdnr)
values (555, 'ABC-titel' , 12345.67, '3-isbn', 1)
print '>>>3 insert nachher'

go
select buchnr, preis, titel from tbuch
order by Buchnr desc
select buchnr,preis,isbn, titel from vtbuchjointisbn
order by buchnr desc
go
rollback
GO
```

7.9 Einschränkungen für VIEWs



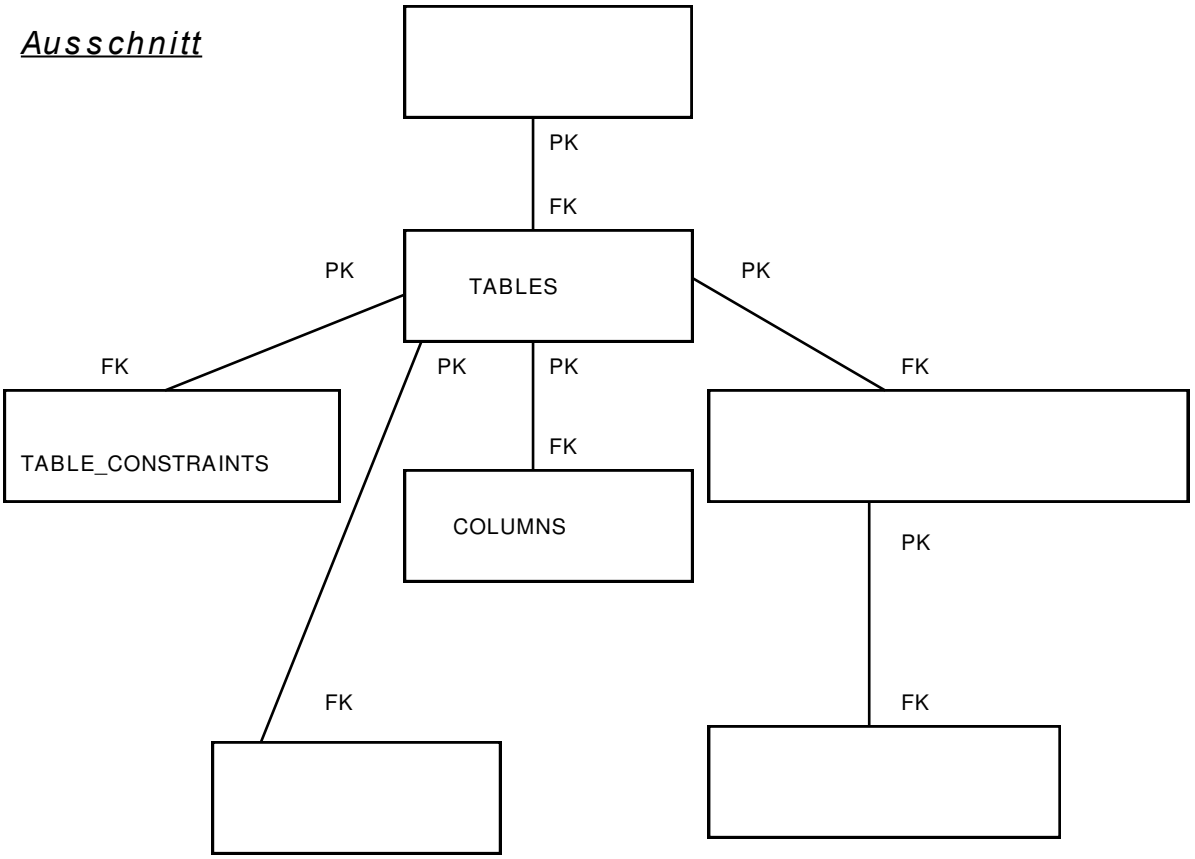
- Kein Produkt unterstützt update-Operationen auf der Menge der theoretisch updatebaren Views.
- Es gibt Views, die Joins sind und theoretisch updatebar.
- "The whole area of views (the retrieval aspect in particular) is one of the ugliest parts of the SQL language" C.J. DATE 1989, Seite 91

- In SQL:1992 ist eine VIEW updatebar, wenn die folgenden Bedingungen auf die VIEW-Definition zutreffen:

  1. The table expression that defines the scope of the view is a select expression; that is, it does not directly contain any of the key words JOIN, UNION, INTERSECT, or EXCEPT.
  2. The SELECT clause of that select expression does not directly contain the key word DISTINCT.
  3. ...
  4. The FROM clause of that select expression contains exactly on table reference.
  5. That table reference identifies either a base table or an updatable view.
  6. ...
  7. That select expression does not include a GROUP BY clause.
  8. That select expression does not include a HAVING clause.

  Aus C. J. DATE 1993, Seite 174

7.10 Der Katalog



- Der Katalog ist eine Metadatenbank, die Informationen über die generierten Objekte enthält. Die Einträge zusammengenommen beschreiben, was sinnvollerweise als Datenbank betrachtet werden kann. Der Begriff DATABASE ist formal im Standard nicht definiert.
- Der Katalog wird zum Beispiel verändert durch

  CREATE, ALTER, DROP von Objekten

  GRANT, REVOKE von Privilegien
- Laut Standard müssen bestimmte VIEWs vorhanden sein:
- DOMAINS TABLES VIEWS COLUMNS DOMAIN_CONSTRAINTS TABLE_CONSTRAINTS REFERENTIAL_CONSTRAINTS CHECK_CONSTRAINTS KEY_COLUMN_USAGE ASSERTIONS ...
- **Über SQL-Zugriffe sind Abfragen möglich!**
- SQL Server 2005 unterstützt diese VIEWs

```
select *
from information_schema.tables
select *
from information_schema.columns
select *
from information_schema.table_constraints
```

7.11 Katalogabfragen

Hinweis Oracle:

```
SELECT * FROM all_catalog WHERE OWNER = USER;
SELECT * FROM all_catalog WHERE OWNER = 'SL03';
SELECT * FROM all_catalog WHERE OWNER = 'sl03';
SELECT * FROM all_tab_columns WHERE ...
```

Hinweis DB2 for Windows NT:

```
SELECT * FROM SYSCAT.TABLES WHERE TABSCHEMA = USER;
SELECT * FROM SYSCAT.TABLES WHERE TABSCHEMA = 'DB2ADMIN';
SELECT * FROM SYSCAT.TABLES WHERE TABSCHEMA = 'db2admin';
SELECT * FROM SYSCAT.COLUMNS WHERE ...
```

Hinweis DB2 for z/OS:

```
SELECT * FROM SYSIBM.SYSTABLES WHERE CREATOR = USER;
SELECT * FROM SYSIBM.SYSCOLUMNS WHERE ...
```

SQL Server:

```
select * from sysobjects
         where sysobjects.uid = USER_ID('DBO');
```

Wir ermitteln aus dem Katalog die Datentypen der Spalten von Tbuch.

Oracle:

```
SELECT * FROM all_catalog
WHERE OWNER = USER;
SELECT * FROM all_tab_columns
WHERE OWNER = USER AND TABLE_NAME LIKE 'TBUCH%';
SELECT
 SUBSTR(OWNER,       1, 8)
,SUBSTR(TABLE_NAME,  1, 7)
,SUBSTR(COLUMN_NAME, 1, 7)
,SUBSTR(DATA_TYPE,   1, 9)
FROM all_tab_columns
WHERE OWNER = USER AND TABLE_NAME LIKE 'TBUCH%';
```

DB2 for Windows NT:

```
SELECT * FROM SYSCAT.TABLES
WHERE TABSCHEMA = USER;
SELECT * FROM SYSCAT.COLUMNS
WHERE TABSCHEMA = USER AND TABNAME LIKE 'TBUCH%';
```

DB2 for z/OS:

```
SELECT * FROM SYSIBM.SYSTABLES
WHERE CREATOR = USER;
SELECT * FROM SYSIBM.SYSCOLUMNS
WHERE WHERE TBCREATOR = USER AND TBNAME LIKE 'TBUCH%';
```

SQL Server:

```
select *
from syscolumns , sysobjects
where
    syscolumns.id  = sysobjects.id
and sysobjects.id  = object_id ('tbuch')
and sysobjects.UID = USER_ID ('DBO')
;
```

7.12 Das Oracle Data Dictionary

Im Data Dictionary (oft auch Katalog genannt) werden die Metadaten abgelegt:

- über jede TABLE, ihre Spaltennamen und Datentypen
- über jede VIEW, ihre Spaltennamen und Datentypen
- über die physische Speicherung der Daten
- ...
- über jeden Benutzer (USER) und seine Berechtigungen,

Das Data Dictionary ist in zwei Ebenen unterteilt:

- Strukturelle Ebene
  Hier werden die Base Tables und ihre Dateninhalte betrachtet. Der normale Benutzer hat hierauf keinen Zugriff.
- Externe Ebene
  Hier handelt es sich um eine Vielzahl von VIEWs, die jedem Benutzer mit Hilfe von SELECT zugänglich sind.

```
set linesize 32767
;
show user;
select table_name, num_rows
from user_tables
where table_name LIKE 'T%'
order by table_name
;
show user;
select owner, table_name
from all_tables
where
NOT
(
       owner   like 'SYS%'
  OR   OWNER   LIKE 'SYSTEM%'
)
AND    OWNER LIKE 'SL%'
order by owner, table_name
;
--select *
--from dba_tables
--;
```

Die Views des Data Dictionarys

Die externe Ebene des Data Dictionary bietet dem Benutzer die Möglichkeit, sich mittels SQL alle relevanten Informationen anzeigen zu lassen. Die Views des externen Ebene sind in drei Gruppen eingeteilt, die eine unterschiedliche Sichtweise erlauben:

USER_ Views dieser Gruppe präsentieren alle Objekte, deren OWNER (Eigentümer) der jeweilige USER (Benutzer) ist (die der USER selbst generiert hat).

```
SELECT * FROM USER_TABLES
WHERE TABLE_NAME = 'TBUCH'
;
SELECT * FROM USER_TAB_COLUMNS
WHERE TABLE_NAME = 'TBUCH'
;
SELECT * FROM USER_CONSTRAINTS
WHERE TABLE_NAME = 'TBUCH'
;
SELECT * FROM USER_VIEWS
WHERE VIEW_NAME = 'VBUCH'
;
```

ALL_ Views dieser Gruppe liefern Informationen über alle Objekte, auf die der Benutzer Zugriff hat. Die Anweisung für Tables lautet hier:

```
SELECT * FROM ALL_TABLES;
```

DBA_ Views mit diesem Präfix sind nur für Benutzer mit DBA-Privileg zugänglich. Diese Views bieten eine Gesamtsicht auf die Database. Alle Tabellen aller Benutzer im gesamten System zeigt die Anweisung:

```
SELECT * FROM DBA_TABLES;
```

Die Namen der einzelnen Views unterscheiden sich nur im Präfix und im Informationsgehalt, nicht jedoch in der grundsätzlichen Funktion.

Die Auflistung aller Dictionary Views und eine kurze Beschreibung bekommt man durch die Anweisung: `SELECT * FROM DICTIONARY;`

user_tab_columns

```
connect SL01/SL01;
show user;
set null "null"
set linesize 600;
col dlength format 9999
col dprec   format 9999
col dscale  format 9999

select
 substr(table_name,  1, 8)   as tname
,substr(column_name, 1, 8)   as cname
,substr(data_type,   1, 8)   as dtype
,DATA_LENGTH                 as dlength
,DATA_PRECISION              as dprec
,DATA_SCALE                  as dscale
from user_tab_columns
WHERE table_name = 'TVERLAG'
OR    table_name = 'TBUCH'
OR    table_name = 'TISBN'
OR    table_name = 'TAUTOR'
OR    table_name = 'TVAUTOR'
order by table_name, column_name;
```

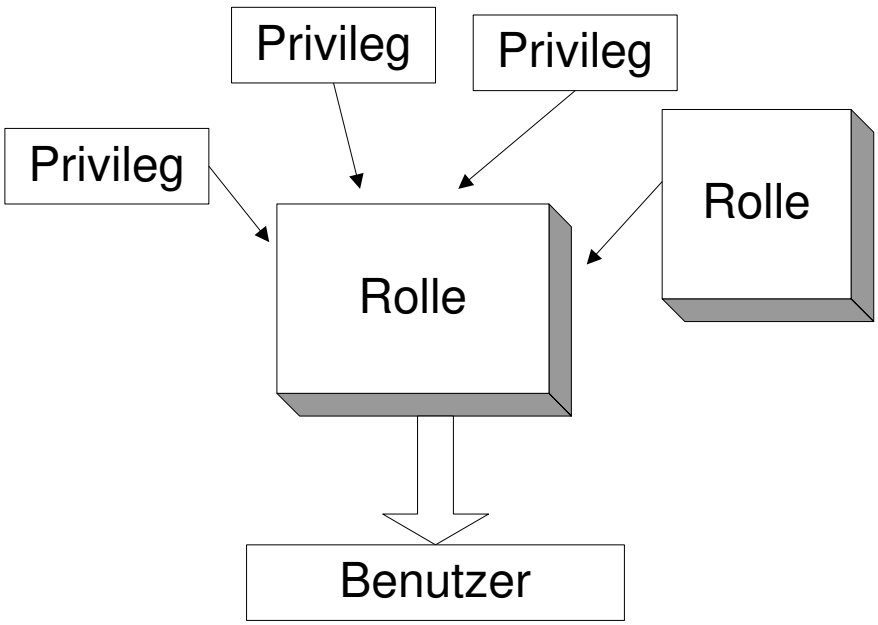
```
Connected.
USER is "SL01"

TNAME    CNAME    DTYPE    DLENGTH DPREC DSCALE
-------- -------- -------- ------- ----- ------
TAUTOR   AUTOR    VARCHAR2     240 null  null
TAUTOR   AUTORNR  NUMBER        22 null       0
TAUTOR   GEBURTSD DATE           7 null  null
TBUCH    BUCHNR   NUMBER        22 null       0
TBUCH    ERSCHJ   NUMBER        22     4      0
TBUCH    PREIS    NUMBER        22     7      2
TBUCH    TITEL    VARCHAR2     127 null  null
TBUCH    VERLAGNR NUMBER        22 null       0
TISBN    BUCHNR   NUMBER        22 null       0
TISBN    ISBN     CHAR          10 null  null
TISBN    LFDNR    NUMBER        22     1      0
TVAUTOR  AUTORNR  NUMBER        22 null       0
TVAUTOR  BUCHNR   NUMBER        22 null       0
TVAUTOR  LFDNR    NUMBER        22     1      0
TVAUTOR  PRAEMIE  NUMBER        22     9      3
TVERLAG  VERLAG   CHAR          20 null  null
TVERLAG  VERLAGNR NUMBER        22 null       0

17 rows selected.
```

user_constraints

```
connect SL01/SL01;
show user;
set null "null"
set linesize 600;

select
  substr(owner, 1, 6)           as owner
,substr(table_name, 1, 6)       as tname
,substr(constraint_name, 1, 10) as conname
,constraint_type                as contype
,delete_rule                    as delrule
,search_condition               as seacond
from user_constraints
WHERE
(
table_name = 'TVERLAG' OR
table_name = 'TBUCH'   OR
table_name = 'TISBN'   OR
table_name = 'TAUTOR'  OR
table_name = 'TVAUTOR'
)
and owner = user
order by owner, table_name, constraint_name;
```

```
Connected.
USER is "SL01"

OWNER  TNAME  CONNAME    C DELRULE   SEACOND
------ ------ ---------- - --------- --------------------------
SL01   TAUTOR SYS_C00710 C null      "AUTORNR" IS NOT NULL
SL01   TAUTOR SYS_C00710 C null      "AUTOR" IS NOT NULL
SL01   TAUTOR SYS_C00710 P null      null
SL01   TBUCH  FK_TBUCH_T R NO ACTION null
SL01   TBUCH  SYS_C00709 C null      "BUCHNR" IS NOT NULL
SL01   TBUCH  SYS_C00709 C null      "TITEL" IS NOT NULL
SL01   TBUCH  SYS_C00709 P null      null
SL01   TISBN  ALTKEY_TIS U null      null
SL01   TISBN  SYS_C00709 C null      "BUCHNR" IS NOT NULL
SL01   TISBN  SYS_C00709 C null      "ISBN" IS NOT NULL
SL01   TISBN  SYS_C00709 C null      "LFDNR" IS NOT NULL
SL01   TISBN  SYS_C00710 P null      null
SL01   TISBN  SYS_C00710 R CASCADE   null
SL01   TVAUTO ALTKEY_TVA U null      null
SL01   TVAUTO SYS_C00710 C null      "BUCHNR" IS NOT NULL
SL01   TVAUTO SYS_C00710 C null      "AUTORNR" IS NOT NULL
SL01   TVAUTO SYS_C00710 C null      "LFDNR" IS NOT NULL
SL01   TVAUTO SYS_C00710 P null      null
SL01   TVAUTO SYS_C00711 R CASCADE   null
SL01   TVAUTO SYS_C00711 R NO ACTION null
SL01   TVERLA SYS_C00709 C null      "VERLAGNR" IS NOT NULL
SL01   TVERLA SYS_C00709 C null      "VERLAG" IS NOT NULL
SL01   TVERLA SYS_C00709 P null      null

23 rows selected.
```

7.13 Datenschutz/Security, USER und ROLE



Das Konzept der ROLE vereinfacht die Verwaltung von Zugriffsrechten.

Eine ROLE ist eine Menge von System- und/ oder Objektprivilegien, die zu einer Gruppe zusammengefasst werden und unter einem Namen gemeinsam ansprechbar sind.

1) CREATE der ROLE, Namensvergabe und evtl. Passwort

2) GRANT der Privilegien an die ROLE

3) GRANT der ROLE an den Benutzer

--Beispiel Oracle

```
connect system as sysdba;
  DROP USER buecher CASCADE;
  CREATE USER buecher IDENTIFIED BY buecher;

  ----grant create session  to buecher with admin option;

  GRANT CONNECT TO buecher;
  GRANT RESOURCE TO buecher;

  GRANT CREATE ROLE TO buecher;
  GRANT DROP ANY ROLE TO buecher;

  DROP USER user001 CASCADE;
  CREATE USER user001 IDENTIFIED BY user001;
  GRANT CREATE SESSION TO user001;

  DROP USER user002 CASCADE;
  CREATE USER user002 IDENTIFIED BY user002;
  GRANT CREATE SESSION TO user002;

  GRANT CREATE SYNONYM TO user001;
  GRANT CREATE SYNONYM TO user002;
```

```
---------------------------
CONNECT buecher;

  CREATE TABLE buecher.Tverlag
  (
   Verlagnr   INTEGER    NOT NULL
  ,Verlag     CHAR(20)   NOT NULL
  ,PRIMARY KEY (Verlagnr)
  )
  ;

  CREATE TABLE buecher.Tbuch
  (
   Buchnr         INTEGER      NOT NULL
  ,Erschj         DECIMAL(4)
  ,Preis          DECIMAL(7,2)
  ,Verlagnr       INTEGER
  ,Titel          VARCHAR(127) NOT NULL
  ,PRIMARY KEY (Buchnr)
  )
  ;
  ALTER TABLE buecher.Tbuch ADD CONSTRAINT
                 FK_Tbuch_Tverlag
                 FOREIGN KEY (Verlagnr)
                 REFERENCES
                 buecher.Tverlag(Verlagnr)
                 ;
               --ON DELETE NO ACTION

  DROP ROLE   buecherrolesel;
  CREATE ROLE buecherrolesel;
  GRANT SELECT ON buecher.tverlag TO buecherrolesel;
  GRANT SELECT ON buecher.tbuch   TO buecherrolesel;


  GRANT buecherrolesel TO user001;
  GRANT buecherrolesel TO user002;
  ----GRANT SELECT ON buecher.Tverlag TO user001, user002;
  ----GRANT SELECT ON buecher.Tbuch TO user001, user002;

  DROP ROLE   buecherroleinsupd;
  cREATE ROLE buecherroleinsupd;
  GRANT INSERT, UPDATE  ON buecher.tverlag TO buecherrolein-
supd;

  GRANT buecherroleinsupd TO user001, user002;


  INsERT INTO buecher.Tverlag
             (verlagnr, verlag)
      values ( 1111   , 'Forkel');
  INSERT INTO buecher.Tbuch
             (buchnr, erschj, preis, titel, verlagnr)
       values(  8 ,  1977, 0.50 , 'der Butt', 1111);
```

```
--------------------------------
CONNECT user001;
  SET ROLE buecherrolesel;
  SELECt * FROM buecher.tverlag;
  SELECT * from buecher.tbuch;
  update buecher.tbuch set preis = 99.99;
  --nicht ausreichende berechtigung

  SET ROLE ALL EXCEPT buecherroleinsupd;
  SELECt * FROM buecher.tverlag;
  SELECT * from buecher.tbuch;
  DELETE FROM buecher.tbuch;
  --nicht ausreichende berechtigung

  set role buecherroleinsupd;
  update buecher.tbuch set preis = 99.99;
  -- tabelle oder view nicht vorhanden!
```

Oracle Data Dictionary Views für Roles

DBA_ROLES	Alle Rollen, die im System existieren.
DBA_ROLE_PRIVS	Rollen, die Usern und Rollen zugewiesen sind.
USER_ROLE_PRIVS	Dem aktuellen Benutzer zugewiesene Rollen.
ROLE_ROLE_PRIVS	Rollen, die Rollen zugewiesen sind.
ROLE_SYS_PRIVS	Systemprivilegien, die Rollen zugewiesen sind.
ROLE_TAB_PRIVS	Objektprivilegien, die Rollen zugewiesen sind.
SESSION_ROLES	Vom aktuellen Benutzer aktivierte Rollen.

7.14 Datenschutz/Security, GRANT und REVOKE

```
GRANT  SELECT ON buchautor1 TO PUBLIC;

REVOKE SELECT ON buchautor1 FROM PUBLIC;

GRANT ALL PRIVILEGES ON Tbuch TO PUBLIC;

GRANT ALL PRIVILEGES ON Vbuch TO PUBLIC;

GRANT SELECT ON Tbuch TO skis41t WITH GRANT OPTION;
GRANT SELECT ON Tbuch TO    SL02 WITH GRANT OPTION;

GRANT UPDATE ...
GRANT INSERT ...
GRANT DELETE ...
```

8

Aufgaben Interaktive SQL

8 Aufgaben Interaktive SQL

8.1 SELECT

8.1.1 Liste1

Buchnr, Erschj, Preis, Titel

sortiert nach Buchnr aufsteigend

BUCHNR	ERSCHJ	PREIS	TITEL
1	null	null	C
2	null	null	C
5	1988	3,5	Ansichten eines Clowns
6	1988	20,5	die Blechtrommel
7	1989	99,99	der Name der Rose
8	1977	,5	der Butt
9	1990	55	DB2 fuer Sie
11	1990	null	Elvis in Heidelberg
12	1989	null	a guide to db2
18	1989	99,99	Database Systems
27	null	99,99	die Jüdin von Toledo

8.1.2 Liste2

Erschj, Preis, Buchnr

sortiert nach Erschj aufsteigend, Preis aufsteigend, Buchnr aufsteigend

Oracle und DB2

```
    ERSCHJ      PREIS     BUCHNR
---------- ---------- ----------
      1977         ,5          8
      1988        3,5          5
      1988       20,5          6
      1989      99,99          7
      1989      99,99         18
      1989 null               12
      1990         55          9
      1990 null               11
null            99,99         27
null       null                1
null       null                2
```

SQL Server

```
Erschj Preis     Buchnr
------ --------- -----------
NULL   NULL      1
NULL   NULL      2
NULL   99.99     27
1977   .50       8
1988   3.50      5
1988   20.50     6
1989   NULL      12
1989   99.99     7
1989   99.99     18
1990   NULL      11
1990   55.00     9
```

8.1.3 Liste3

Erschj, Preis, Buchnr

sortiert nach Erschj absteigend, Preis absteigend, Buchnr absteigend

Oracle und DB2

ERSCHJ	PREIS	BUCHNR
null	null	2
null	null	1
null	99,99	27
1990	null	11
1990	55	9
1989	null	12
1989	99,99	18
1989	99,99	7
1988	20,5	6
1988	3,5	5
1977	,5	8

SQL Server

Erschj	Preis	Buchnr
1990	55.00	9
1990	NULL	11
1989	99.99	18
1989	99.99	7
1989	NULL	12
1988	20.50	6
1988	3.50	5
1977	.50	8
NULL	99.99	27
NULL	NULL	2
NULL	NULL	1

8.1.4 Liste4

Erschj, Preis, Buchnr

sortiert nach Preis absteigend, Erschj aufsteigend, Buchnr aufsteigend

Oracle und DB2

ERSCHJ	PREIS	BUCHNR
1989	null	12
1990	null	11
null	null	1
null	null	2
1989	99,99	7
1989	99,99	18
null	99,99	27
1990	55	9
1988	20,5	6
1988	3,5	5
1977	,5	8

SQL Server

Erschj	Preis	Buchnr
NULL	99.99	27
1989	99.99	7
1989	99.99	18
1990	55.00	9
1988	20.50	6
1988	3.50	5
1977	.50	8
NULL	NULL	1
NULL	NULL	2
1989	NULL	12
1990	NULL	11

8.1.5 Liste5

Titel, Buchnr

sortiert nach Titel, Buchnr

DB2 for Windows

```
TITEL                          BUCHNR
------------------------------ -----------
Ansichten eines Clowns                   5
C                                        1
C                                        2
DB2 fuer Sie                             9
Database Systems                        18
Elvis in Heidelberg                     11
a guide to db2                          12
der Butt                                 8
der Name der Rose                        7
die Blechtrommel                         6
die Jüdin von Toledo                    27
```

Oracle

```
alter session set nls_sort = german;
alter session set nls_comp = linguistic;
```

```
TITEL                            BUCHNR
------------------------------ ----------
a guide to db2                         12
Ansichten eines Clowns                  5
C                                       1
C                                       2
Database Systems                       18
DB2 fuer Sie                            9
der Butt                                8
der Name der Rose                       7
die Blechtrommel                        6
die Jüdin von Toledo                   27
Elvis in Heidelberg                    11
```

SQL Server

```
titel                     buchnr
------------------------- -----------
a guide to db2            12
Ansichten eines Clowns    5
C                         1
C                         2
Database Systems          18
DB2 fuer Sie              9
der Butt                  8
der Name der Rose         7
die Blechtrommel          6
die Jüdin von Toledo      27
Elvis in Heidelberg       11
```

8.1.6 Liste6

Erschj, Preis

Ausprägungen dieser Kombination

sortiert nach Erschj, Preis

Oracle und DB2

```
ERSCHJ      PREIS
---------- ----------
      1977         ,5
      1988        3,5
      1988       20,5
      1989      99,99
      1989 null
      1990         55
      1990 null
null            99,99
null       null
```

SQL Server

```
Erschj Preis
------ ---------
NULL   NULL
NULL   99.99
1977   .50
1988   3.50
1988   20.50
1989   NULL
1989   99.99
1990   NULL
1990   55.00
```

8.1.7 Liste7

Preis

Ausprägungen von Preisen

sortiert absteigend

Oracle und DB2

```
     PREIS
----------
null
     99,99
        55
      20,5
       3,5
        ,5

6 rows selected.
```

SQL Server

```
Preis
---------
99.99
55.00
20.50
3.50
.50
NULL

(6 row(s) affected)
```

8.1.8 Liste8

Buchnr, Titel

von Büchern, die 1988 erschienen sind

sortiert nach Buchnr

```
    BUCHNR TITEL
---------- --------------------------------------------------
         5 Ansichten eines Clowns
         6 die Blechtrommel
```

8.1.9 Liste9

Buchnr, Titel

von Büchern, die nicht 1988 erschienen sind

sortiert nach Buchnr aufsteigend

```
    BUCHNR TITEL
---------- --------------------------------------------------
         1 C
         2 C
         7 der Name der Rose
         8 der Butt
         9 DB2 fuer Sie
        11 Elvis in Heidelberg
        12 a guide to db2
        18 Database Systems
        27 die Jüdin von Toledo

9 rows selected.
```

8.1.10 Liste10

Buchnr

von Büchern, die 3.50 oder 55.00 kosten

sortiert nach Buchnr aufsteigend

```
BUCHNR
----------
         5
         9
```

8.1.11 Liste11

Buchnr

aller Bücher von 1977 *und* 1988

sortiert nach Buchnr aufsteigend

(Achtung: „und“ ist Alltagssprache)

```
BUCHNR
----------
         5
         6
         8
```

8.1.12 Liste12

Buchnr, Erschj, Preis

aller Bücher von 1990 oder teurer als 50.00

sortiert nach Buchnr aufsteigend

```
    BUCHNR     ERSCHJ      PREIS
---------- ---------- ----------
         7       1989      99,99
         9       1990         55
        11       1990 null
        18       1989      99,99
        27 null            99,99
```

8.1.13 Liste13

Buchnr, Erschj, Preis

aller Bücher von 1989 und mit Preis 99.99

sortiert nach Buchnr aufsteigend

```
    BUCHNR     ERSCHJ      PREIS
---------- ---------- ----------
         7       1989      99,99
        18       1989      99,99

2 rows selected.
```

8.1.14 Liste14

Buchnr, Erschj, Preis

aller Bücher von 1989 oder mit Preis 99.99

sortiert nach Buchnr aufsteigend

```
    BUCHNR     ERSCHJ      PREIS
---------- ---------- ----------
         7       1989      99,99
        12       1989 null
        18       1989      99,99
        27 null            99,99

4 rows selected.
```

8.1.15 Liste15

Buchnr, Erschj, Preis

aller Bücher von 1988 bzw. 1989 bzw. 1990

```
buchnr      erschj preis
----------- ------ ---------
5           1988   3.50
6           1988   20.50
7           1989   99.99
9           1990   55.00
11          1990   NULL
12          1989   NULL
18          1989   99.99
```

8.1.16 Liste16

Buchnr, Erschj, Preis

aller Bücher von 1988 bzw. 1989 bzw. 1990 und Preis über 10 Euro

```
buchnr      erschj preis
----------- ------ ---------
6           1988   20.50
7           1989   99.99
9           1990   55.00
18          1989   99.99
```

8.1.17 Liste17

Buchnr, Erschj, Preis

aller Bücher von 1988 bzw. 1989 bzw. 1990 und Preis nicht über 10 Euro

Variante IST

```
buchnr      erschj preis
----------- ------ ---------
5           1988   3.50
```

Variante SOLL (mit Berücksichtigung der NULL)

```
buchnr      erschj preis
----------- ------ ---------
5           1988   3.50
11          1990   NULL
12          1989   NULL
```

8.1.18 Liste18

Buchnr, Erschj, Preis

aller Bücher von 1988 bzw. 1989 bzw. 1990 oder Preis über 10 Euro

```
buchnr      erschj preis
----------- ------ ---------
5           1988   3.50
6           1988   20.50
7           1989   99.99
9           1990   55.00
11          1990   NULL
12          1989   NULL
18          1989   99.99
27          NULL   99.99
```

8.1.19 Liste19

Buchnr, Erschj, Preis

aller Bücher von 1988 bzw. 1989 bzw. 1990 oder Preis nicht über 10 Euro

Variante IST

```
buchnr      erschj preis
----------- ------ ---------
5           1988   3.50
6           1988   20.50
7           1989   99.99
8           1977   .50
9           1990   55.00
11          1990   NULL
12          1989   NULL
18          1989   99.99

(8 row(s) affected)
```

Variante SOLL

```
buchnr      erschj preis
----------- ------ ---------
1           NULL   NULL
2           NULL   NULL
5           1988   3.50
6           1988   20.50
7           1989   99.99
8           1977   .50
9           1990   55.00
11          1990   NULL
12          1989   NULL
18          1989   99.99

(10 row(s) affected)
```

8.1.20 Liste20

Buchnr, Titel

aller Bücher mit dem Titel der Butt

Was erwarten Sie bei folgenden Abfragen:

```
SELECT Buchnr, Titel FROM Tbuch
WHERE Titel = 'DER BUTT';

SELECT Buchnr, Titel FROM Tbuch
WHERE Titel = 'DER BUTT     ';

SELECT Buchnr, Titel FROM Tbuch
WHERE Titel = 'der Butt';

SELECT Buchnr, Titel FROM Tbuch
WHERE Titel = 'der Butt      ';

SELECT Buchnr, Titel FROM Tbuch
WHERE UPPER(Titel) = 'DER BUTT';

SELECT Buchnr, Titel FROM Tbuch
WHERE UPPER(Titel) = 'DER BUTT     ';
```

```
BUCHNR     TITEL
---------- -----------
         8 der Butt
1 row selected.
```

oder

```
BUCHNR     TITEL
---------- -----------
         8 DER BUTT
1 row selected.
```

oder

```
BUCHNR     TITEL
---------- -----------
no rows selected
```

8.1.21 Liste21

Preis, Buchnr

aller Bücher mit Preis zwischen 20.50 und 99.99

sortiert nach Preis aufsteigend, Buchnr aufsteigend

```
PREIS     BUCHNR
---------- ----------
      20,5          6
        55          9
     99,99          7
     99,99         18
     99,99         27

5 rows selected.
```

8.1.22 Liste22

Was erwarten Sie bei folgenden Abfragen, testen Sie Ihren Server!

Titel, Buchnr

aller Bücher mit Titel zwischen 'A' und 'D'

sortiert nach Titel, Buchnr

```
SELECT
  SUBSTR(Titel, 1, 30)    AS Titel --Oracle, DB2
--SUBSTRING(Titel, 1, 30) AS Titel --SQL Server
, Buchnr
FROM Tbuch
WHERE Titel BETWEEN 'A' AND 'D'
ORDER BY Titel, Buchnr
;
```

Titel, Buchnr

aller Bücher mit Titel zwischen 'a' und 'd'

sortiert nach Titel, Buchnr

```
SELECT
  SUBSTR(Titel, 1, 30)    AS Titel --Oracle, DB2
--SUBSTRING(Titel, 1, 30) AS Titel --SQL Server
, Buchnr
FROM Tbuch
WHERE Titel BETWEEN 'a' AND 'd'
ORDER BY Titel, Buchnr
;
```

8.1.23 Liste23

Was erwarten Sie bei folgenden Abfragen, testen Sie Ihren Server!

Titel, Buchnr
aller Bücher mit Titel zwischen 'A' und 'C'
sortiert nach Titel, Buchnr

Titel, Buchnr
aller Bücher mit Titel zwischen 'a' und 'c'
sortiert nach Titel, Buchnr

Titel, Buchnr
aller Bücher mit Titel zwischen 'A' und 'E'
sortiert nach Titel, Buchnr

Titel, Buchnr
aller Bücher mit Titel zwischen 'a' und 'e'
sortiert nach Titel, Buchnr

8.1.24 Liste24

Titel, Buchnr

aller Bücher deren Titel mit A, B, C oder D anfängt

sortiert nach Titel, Buchnr

Oracle und DB2 CASE SENSITIVE

```
TITEL                                BUCHNR
------------------------------ ----------
Ansichten eines Clowns                    5
C                                         1
C                                         2
Database Systems                         18
DB2 fuer Sie                              9

5 rows selected.
```

SQL Server CASE INSENSITIVE

```
Titel                          Buchnr
------------------------------ -----------
a guide to db2                 12
Ansichten eines Clowns         5
C                              1
C                              2
Database Systems               18
DB2 fuer Sie                   9
der Butt                       8
der Name der Rose              7
die Blechtrommel               6
die Jüdin von Toledo           27

(10 row(s) affected)
```

8.1.25 Liste25

Titel, Buchnr

aller Bücher deren Titel der, die oder das enthält

sortiert nach Titel, Buchnr

TITEL	BUCHNR
der Butt	8
der Name der Rose	7
die Blechtrommel	6
die Jüdin von Toledo	27

8.2 OR AND NOT – UNION INTERSECT EXCEPT

8.2.1 NOT B1 bzw. NOT B2

Aktuelle Daten von Tbuch:

buchnr	erschj	preis
100	1989	99.99
200	1989	22.22
300	2000	99.99
400	2000	22.22
500	1989	NULL
600	NULL	99.99
700	NULL	NULL
800	2000	NULL
900	NULL	22.22

B1: Tbuch.Erschj = 1989 B2: Tbuch.Preis = 99.99
SOLL: mit besonderer Berücksichtigung der NULL
Schreiben Sie die SELECT-Anweisungen

Bedingung für die Base-Table Tbuch		SOLL
B1 erschj=1989	100 200 500	
NOT B1		300 400 800 600 700 900
B2 Preis=99.99	100 300 600	
NOT B2		200 400 900 500 700 800

```
delete from tbuch;
INSERT INTO tbuch(buchnr,erschj, preis, titel) VALUES
 ( 100, 1989, 99.99, 'xyz');
INSERT INTO tbuch(buchnr,erschj, preis, titel) VALUES
 ( 200, 1989, 22.22, 'xyz');
INSERT INTO tbuch(buchnr,erschj, preis, titel) VALUES
 ( 300, 2000, 99.99, 'xyz');
INSERT INTO tbuch(buchnr,erschj, preis, titel) VALUES
 ( 400, 2000, 22.22, 'xyz');
INSERT INTO tbuch(buchnr,erschj, preis, titel) VALUES
 ( 500, 1989, NULL, 'xyz');
INSERT INTO tbuch(buchnr,erschj, preis, titel) VALUES
 ( 600, NULL, 99.99, 'xyz');
INSERT INTO tbuch(buchnr,erschj, preis, titel) VALUES
 ( 700, NULL, NULL, 'xyz');
INSERT INTO tbuch(buchnr,erschj, preis, titel) VALUES
 ( 800, 2000, NULL, 'xyz');
INSERT INTO tbuch(buchnr,erschj, preis, titel) VALUES
 ( 900, NULL, 22.22, 'xyz');
select buchnr, erschj, preis from tbuch order by buchnr;
```

```
--B1
--erschj = 1989
select buchnr, erschj, preis from tbuch
where Erschj = 1989
;
--NOT B1 --SOLL der Rest
```

```
--B2
--preis = 99.99
select buchnr, erschj, preis from tbuch
where preis = 99.99
;
--NOT B2 --SOLL der Rest
```

8.2.2 NOT (B1 AND B2) – (NOT B1) OR (NOT B2)

Aktuelle Daten von Tbuch:

buchnr	erschj	preis
100	1989	99.99
200	1989	22.22
300	2000	99.99
400	2000	22.22
500	1989	NULL
600	NULL	99.99
700	NULL	NULL
800	2000	NULL
900	NULL	22.22

B1: Tbuch.Erschj = 1989 B2: Tbuch.Preis = 99.99
SOLL: mit besonderer Berücksichtigung der NULL
Schreiben Sie die SELECT-Anweisungen

Bedingung für die Base-Table Tbuch		SOLL
B1 AND B2 1989 und 99.99	100	
NOT(B1 AND B2) äquivalent (NOT B1) OR (NOT B2)		200 300 400 500 600 700 800 900

```
--B1 AND B2
select buchnr, erschj, preis from tbuch
where erschj = 1989 AND preis = 99.99
;
--NOT (B1 AND B2) – (NOT B1) OR (NOT B2) --SOLL der Rest
```

8.2.3 NOT (B1 OR B2) – (NOT B1) AND (NOT B2)

Aktuelle Daten von Tbuch:

buchnr	erschj	preis
100	1989	99.99
200	1989	22.22
300	2000	99.99
400	2000	22.22
500	1989	NULL
600	NULL	99.99
700	NULL	NULL
800	2000	NULL
900	NULL	22.22

B1: Tbuch.Erschj = 1989 B2: Tbuch.Preis = 99.99
SOLL: mit besonderer Berücksichtigung der NULL
Schreiben Sie die SELECT-Anweisungen

Bedingung für die Base-Table Tbuch		SOLL
B1 OR B2 1989 oder 99.99	100 200 300 500 600	
NOT(B1 OR B2) äquivalent (NOT B1) AND (NOT B2)		400 700 800 900

```
--B1 OR B2
select buchnr, erschj, preis from tbuch
where erschj = 1989 OR preis = 99.99
;
--NOT (B1 OR B2) – (NOT B1) AND (NOT B2)--SOLL der Rest
```

8.2.4 entweder B1 oder B2 und die NULL?

Aktuelle Daten von Tbuch:

buchnr	erschj	preis
100	1989	99.99
200	1989	22.22
300	2000	99.99
400	2000	22.22
500	1989	NULL
600	NULL	99.99
700	NULL	NULL
800	2000	NULL
900	NULL	22.22

B1: Tbuch.Erschj = 1989 B2: Tbuch.Preis = 99.99
SOLL: mit besonderer Berücksichtigung der NULL
Schreiben Sie die SELECT-Anweisungen

Bedingung für die Base-Table Tbuch		SOLL
>>(entweder … oder)<< **>>entweder 1989 oder 99.99** **aber nicht beides gleichzeitig<<** (B1 OR B2) AND NOT (B1 AND B2) äquivalent (B1 AND (NOT B2)) OR (B2 AND (NOT B1))	200 300	
>>(entweder … oder) und die NULL?<<		200 300 500 600

```
--entweder … oder
--(B1 OR B2) AND NOT(B1 AND B2)
--(B1 AND (NOT B2)) OR (B2 AND (NOT B1))
select buchnr, erschj, preis from tbuch
where
    (
     (Erschj=1989 OR Preis=99.99)
      AND NOT
     (Erschj=1989 AND Preis=99.99)
    )
;
----äquivalent
select buchnr, erschj, preis from tbuch
where
  (erschj = 1989 and (preis <> 99.99))
OR
  (preis = 99.99 and (erschj<>1989))
;
```

```
--entweder … oder und die NULL?
select buchnr, erschj, preis from tbuch
where
    (
     (Erschj=1989 OR Preis=99.99)
      AND NOT
     (Erschj=1989 AND Preis=99.99)
    )
OR  (Erschj = 1989 and preis is null)
OR  (preis = 99.99 and erschj is null)
;
----äquivalent
select buchnr, erschj, preis from tbuch
where
  (erschj = 1989 and (preis <> 99.99 or preis IS NULL))
OR
  (preis = 99.99 and (erschj<>1989 or erschj IS NULL))
;
----äquivalent
     select buchnr, erschj, preis
     from tbuch
     WHERE Erschj = 1989 OR Preis = 99.99
EXCEPT --MINUS
     select buchnr, erschj, preis
     from tbuch
     WHERE Erschj = 1989 AND Preis = 99.99
;
```

8.3 MAX MIN AVG SUM COUNT und GROUP BY

8.3.1 Formulieren Sie Ihre Erwartung, testen Sie!

```
--AGG001a
SELECT abc.erschj        AS erschj
     , COUNT(abc.buchnr) AS countbuchnr
FROM TBUCH abc
WHERE abc.ERSCHJ IN (1977, 1989)
GROUP BY abc.erschj
;
--AGG001b
SELECT abc.erschj        AS erschj
     , COUNT(abc.buchnr) AS countbuchnr
FROM TBUCH abc
GROUP BY abc.erschj
HAVING abc.erschj IN (1977, 1989)
;
```

```
--AGG002a
SELECT abc.erschj        AS erschj
     , COUNT(abc.buchnr) AS countbuchnr
FROM TBUCH abc
WHERE abc.ERSCHJ IN (1977, 1989, NULL)
GROUP BY abc.erschj
;
--AGG002b
SELECT abc.erschj        AS erschj
     , COUNT(abc.buchnr) AS countbuchnr
FROM TBUCH abc
GROUP BY abc.erschj
HAVING abc.erschj IN (1977, 1989, NULL)
;
```

```
--AGG003a
SELECT abc.erschj        AS erschj
     , COUNT(abc.buchnr) AS countbuchnr
FROM TBUCH abc
WHERE abc.ERSCHJ IN (1977, 1989) OR abc.erschj IS NULL
GROUP BY abc.erschj
;
--AGG003b
SELECT abc.erschj        AS erschj
     , COUNT(abc.buchnr) AS countbuchnr
FROM TBUCH abc
GROUP BY abc.erschj
HAVING abc.erschj IN (1977, 1989, NULL) OR abc.erschj IS NULL
;
```

--AGG004

Natürlich ist es ein Unsinn, den Durchschnitt aller Buchnummern zu berechnen! Beachten Sie aber bitte die Listings!

```
SELECT
          MAX(TBUCH.BUCHNR) AS MAXBU
         ,MIN(TBUCH.BUCHNR) AS MINBU
         ,AVG(TBUCH.BUCHNR) AS AVGBU
         ,SUM(TBUCH.BUCHNR) AS SUMBU
         ,COUNT(buchnr)     AS ANZBU
FROM TBUCH
;
```

Oracle

```
     MAXBU      MINBU      AVGBU      SUMBU      ANZBU
---------- ---------- ---------- ---------- ----------
        27          1 9,63636364        106         11
1 row selected.
```

DB2

```
MAXBU       MINBU       AVGBU       SUMBU       ANZBU
----------- ----------- ----------- ----------- -----------
         27           1           9         106          11

  1 Satz/Sätze ausgewählt.
```

SQL Server

```
MAXBU       MINBU       AVGBU       SUMBU       ANZBU
----------- ----------- ----------- ----------- -----------
27          1           9           106         11

(1 row(s) affected)
```

8.3.2 GROUP BY HAVING

--sel001: Pro Erscheinungsjahr soll der größte und kleinste Preis, der Durchschnittspreis, die Summe der Preise und die Anzahl der Bücher angelistet werden.

ERSCHJ	MAXPREIS	MINPREIS	AVGPREIS	SUMPREIS	ANZBUECHER
1977	,5	,5	,5	,5	1
1988	20,5	3,5	12	24	2
1989	99,99	99,99	99,99	199,98	3
1990	55	55	55	55	2
null	99,99	99,99	99,99	99,99	3

--sel002: wie sel001, aber nur Erscheinungsjahre mit mindestens 2 Büchern und einem Durchschnittspreis unter 50 Euro.

ERSCHJ	MAXPREIS	MINPREIS	AVGPREIS	SUMPREIS	ANZBUECHER
1988	20,5	3,5	12	24	2

--sel003: wie sel002, aber nur Erscheinungsjahre mit mindestens 2 Büchern und einem Durchschnittspreis unter 50 Euro oder Erscheinungsjahre mit genau einem Buch.

ERSCHJ	MAXPREIS	MINPREIS	AVGPREIS	SUMPREIS	ANZBUECHER
1977	,5	,5	,5	,5	1
1988	20,5	3,5	12	24	2

--sel004: Verdichtung von Tvautor pro Buchnr (Anzahl der Autoren, Durchschnittspraemie, maximale Praemie)

```
buchnr      anzautoren  avgprprobuch                    maxprprobuch
----------- ----------- ------------------------------- ------------
1           2           NULL                                   NULL
2           1           NULL                                   NULL
5           1           NULL                                   NULL
6           1           NULL                                   NULL
7           1           NULL                                   NULL
8           1           20.000000                            20.000
9           2           254.000000                          498.000
12          2           30.000000                            30.000

(8 row(s) affected)
```

--sel005: Wie sel004 aber nur Bücher mit mehr als einem Autor.

```
buchnr      anzautoren  avgprprobuch                  maxprprobuch
----------- ----------- ----------------------------- ------------
1           2           NULL                             NULL
9           2           254.000000                       498.000
12          2           30.000000                        30.000

(3 row(s) affected)
```

--sel006: Wie sel005 aber nur Bücher mit wenigstens einer wohldefinierten Praemie.

```
buchnr      anzautoren  avgprprobuch                maxprprobuch
----------- ----------- ------------------------------------------
9           2           254.000000                      498.000
12          2           30.000000                        30.000

(2 row(s) affected)
```

8.3.3 GROUP BY „Redundanzen!"

```
SELECT
 tbuch.erschj                as erschj
,max(tbuch.preis)            as mapr
,min(tbuch.preis)            as mipr
,sum(tbuch.preis*1.19)       as suprbrutto
,avg(tbuch.preis*1.19)       as avprbrutto
,count(tbuch.buchnr)         as anz
,count(tbuch.preis)          as copreis
,count(distinct tbuch.preis) as codipreis
from tbuch
group by tbuch.erschj
HAVING
    avg(tbuch.preis*1.19) >  10.00
AND
    count(tbuch.buchnr)  = 2
AND
   max(tbuch.preis*1.19) - min(tbuch.preis*1.19)
   > 15.00
;
```

Aufgabe: Formulieren Sie die SELECT-Anweisung ohne HAVING!

Die Lösung finden Sie auf der nächsten Seite.

```
Select
 zwi.erschj                  as erschj
,zwi.mapr                    as mapr
,zwi.mipr                    as mipr
,zwi.suprbrutto              as suprbrutto
,zwi.avprbrutto              as avprbrutto
,zwi.anz                     as anz
,zwi.copreis                 as copreis
,zwi.codipreis               as codipreis
FROM
(
select
 erschj                     as erschj
,max(preis)                 as mapr
,min(preis)                 as mipr
,max(preis*1.19)            as maprbrutto
,min(preis*1.19)            as miprbrutto
,sum(preis*1.19)            as suprbrutto
,avg(preis*1.19)            as avprbrutto
,count(buchnr)              as anz
,count(preis)               as copreis
,count(distinct preis)      as codipreis
from tbuch
group by erschj
) zwi
WHERE
    zwi.avprbrutto >  10.00
AND
    zwi.anz = 2
AND
    maprbrutto - miprbrutto > 15.00
;
```

8.4 Inner Natural-Join, innerer natürlicher Verbund, INNER JOIN

Formulieren Sie die Inner Natural-Joins mit Hilfe von INNER JOIN!

8.4.1 Join1

```
SELECT
  Tbuch.Buchnr AS Buchnr
, Tbuch.Preis  AS Preis
, Tbuch.Erschj AS Erschj
, Tisbn.Lfdnr  AS Lfdnr
, Tisbn.ISBN
FROM   Tbuch, Tisbn
WHERE  Tbuch.Buchnr = Tisbn.Buchnr
;
BUCHNR      PREIS     ERSCHJ LFDNR ISBN
----------- --------- ------ ----- ----------
          8      0,50  1977,    1, 3472864303
          8      0,50  1977,    2, 34728643yx
         12         -  1989,    1, 0201501139
```

8.4.2 Join2

```
SELECT
 Tisbn.Buchnr     AS Buchnr
,Tisbn.Lfdnr      AS Lfdnr
,Tisbn.ISBN       AS Isbn
,Tvautor.Autornr  AS Autornr
FROM  Tisbn, Tvautor
WHERE Tisbn.Buchnr = Tvautor.Buchnr
;
BUCHNR      LFDNR ISBN       AUTORNR
----------- ----- ---------- -----------
          8    1, 3472864303           2
          8    2, 34728643yx           2
         12    1, 0201501139          20
         12    1, 0201501139          21
```

8.4.3 Join3

```
SELECT
 Tbuch.Buchnr      AS Buchnr
,Tbuch.Preis       AS Preis
,Tbuch.Erschj      AS Erschj
,Tisbn.Lfdnr       AS Lfdnr
,Tisbn.ISBN
,Tvautor.Autornr   AS Autornr
FROM Tisbn, Tbuch, Tvautor
WHERE  Tisbn.Buchnr = Tbuch.Buchnr
AND    Tbuch.Buchnr = Tvautor.Buchnr
;
BUCHNR      PREIS     ERSCHJ LFDNR ISBN       AUTORNR
----------- --------- ------ ----- ---------- -----------
          8      0,50  1977,    1, 3472864303           2
          8      0,50  1977,    2, 34728643yx           2
         12         -  1989,    1, 0201501139          20
         12         -  1989,    1, 0201501139          21
```

8.4.4 Join4

```
SELECT
 Tbuch.Buchnr      AS Buchnr
,Tbuch.Preis       AS Preis
,Tbuch.Erschj      AS Erschj
,Tvautor.Autornr   AS Autornr
,Tautor.Autor      AS Autor
FROM Tbuch, Tvautor, Tautor
WHERE  Tbuch.Buchnr = Tvautor.Buchnr
AND    Tvautor.Autornr = Tautor.Autornr
ORDER BY Buchnr, Autornr
;
BUCHNR      PREIS     ERSCHJ AUTORNR     AUTOR
----------- --------- ------ ----------- ----------------------
          1         -      -         100 BUSCH
          1         -      -         200 BUSCH
          2         -      -         200 BUSCH
          5      3,50  1988,           1 Boell
          6     20,50  1988,           2 Grass
          7     99,99  1989,           3 Eco
          8      0,50  1977,           2 Grass
          9     55,00  1990,          10 Emil Hack
          9     55,00  1990,          11 Frieda Holz
         12         -  1989,          20 C. J. Date
         12         -  1989,          21 Colin J. White
```

8.4.5 Join5

```
SELECT
 Tverlag.verlagnr    as verlagnr
,Tverlag.verlag      as verlag
,Tbuch.buchnr        as buchnr
FROM Tverlag, Tbuch
WHERE  Tverlag.verlagnr = Tbuch.verlagnr
ORDER BY verlagnr, buchnr
;
VERLAGNR    VERLAG                BUCHNR
----------- -------------------- -----------
       1111 Forkel                       8
       1111 Forkel                       9
```

8.4.6 Join6

```
SELECT
 Tverlag.verlagnr    as verlagnr
,Tverlag.verlag      as verlag
,Tbuch.buchnr        as buchnr
,Tisbn.isbn          as isbn
FROM Tverlag, Tbuch, Tisbn
WHERE  Tverlag.verlagnr = Tbuch.verlagnr
AND    Tbuch.buchnr = tisbn.buchnr
ORDER BY verlagnr, buchnr, isbn
;
SELECT
 Tverlag.verlagnr    as verlagnr
,Tverlag.verlag      as verlag
,Tbuch.buchnr        as buchnr
,tisbn.isbn
FROM Tverlag INNER JOIN Tbuch
     ON  Tverlag.verlagnr = Tbuch.verlagnr
              INNER JOIN tisbn
     ON tbuch.buchnr = tisbn.buchnr
ORDER BY verlagnr, buchnr, isbn
;
verlagnr    verlag               buchnr      isbn
----------- -------------------- ----------- ----------
1111        Forkel               8           3472864303
1111        Forkel               8           34728643yx
```

8.5 Join, Restriktion, Projektion ohne DISTINCT

8.5.1 JOI001: Buchnr, ISBN's von der Butt

```
BUCHNR      ISBN
----------- ----------
          8 3472864303
          8 34728643yx
```

8.5.2 JOI002: Buchnr, Autornr, Autor von Grass

```
BUCHNR      AUTORNR     AUTOR
----------- ----------- --------------
          6           2 Grass
          8           2 Grass
```

8.5.3 JOI003: Autornr, Autor, Titel, Buchnr von a guide to db2

```
AUTORNR     AUTOR                TITEL                BUCHNR
----------- -------------------- -------------------- -----------
         20 C. J. Date           a guide to db2               12
         21 Colin J. White       a guide to db2               12
```

8.5.4 JOI004: Buchnr, ISBN, Preis, Titel

```
BUCHNR      ISBN       PREIS     TITEL
----------- ---------- --------- --------------------
          8 3472864303      0,50 der Butt
          8 34728643yx      0,50 der Butt
         12 0201501139         - a guide to db2
```

8.5.5 JOI005: Buchnr, ISBN, Preis, Titel von der Butt

```
BUCHNR      ISBN       PREIS     TITEL
----------- ---------- --------- ----------
          8 3472864303      0,50 der Butt
          8 34728643yx      0,50 der Butt
```

8.5.6 JOI006: Buchnr, Preis, Titel von Büchern die nur einen Autor haben (Hinweis: Tvautor.Lfdnr = 0)

```
BUCHNR      PREIS     TITEL
----------- --------- ------------------------------
          2         - C
          5      3,50 Ansichten eines Clowns
          6     20,50 die Blechtrommel
          7     99,99 der Name der Rose
          8      0,50 der Butt
```

ACHTUNG: Wenn die Anwendungsprogramme bzw. das Datenbankmanagementsystem nicht garantieren, dass die Llfdnr=0 ist bei genau einem Autor, dann kommt es bei Lfdnr=0 zu inkonsistenten Listen.

Lösung ohne Tvautor.Lfdnr:

```
SELECT
  Tbuch.BUCHNR, Tbuch.PREIS, Tbuch.TITEL
FROM Tbuch
WHERE Tbuch.Buchnr IN
      (
       SELECT Tvautor.Buchnr
       FROM Tvautor
       GROUP BY Tvautor.Buchnr
       HAVING COUNT(tvautor.autornr) = 1
      )
;
```

8.5.7 JOI007: Buchnr, Preis, Titel, Autornr, Autor von Büchern die nur einen Autor haben (Hinweis: Tvautor.Lfdnr = 0)

```
BUCHNR      PREIS     TITEL      AUTORNR     AUTOR
----------- --------- ---------- ----------- ------------------
          2         - C                  200 BUSCH
          5      3,50 Ansichten            1 Boell
          6     20,50 die Blecht           2 Grass
          7     99,99 der Name d           3 Eco
          8      0,50 der Butt             2 Grass
```

8.5.8 JOI008: Buchnr, Preis, Titel, Autornr von Autoren mit dem Namen Boell

```
BUCHNR      PREIS     TITEL      AUTORNR
----------- --------- ---------- -----------
          5      3,50 Ansichten            1
```

8.5.9 JOI009: Buchnr, Preis, Autornr von Büchern mit einem 1. Autor des Namens Boell

```
BUCHNR      PREIS     AUTORNR
----------- --------- -----------

  0 Satz/Sätze ausgewählt.
```

8.5.10 JOI010: Buchnr, Preis, Autornr von Büchern bei denen wenigstens einer der Autoren den Namen Grass hat

```
BUCHNR      PREIS     AUTORNR
----------- --------- -----------
          6     20,50           2
          8      0,50           2
```

8.5.11 JOI011: Buchnr, Preis, Autornr aller Bücher mit Autor Grass und Titel Blechtrommel

```
BUCHNR      PREIS     AUTORNR
----------- --------- -----------
  0 Satz/Sätze ausgewählt.
```

8.5.12 JOI012: Buchnr, Preis, Autornr aller Bücher mit Autor Grass und Titel die Blechtrommel

```
DB2
BUCHNR      PREIS     AUTORNR
----------- --------- -----------
          6     20,50           2

ORACLE die Spalte Tbuch.Titel ist VARCHAR2
no rows selected


SQL Server
BUCHNR      PREIS     AUTORNR
----------- --------- -----------
6           20.50     2
```

8.6 left outer join

8.6.1 leftjoin1

Buchnr, Erschj aller Bücher,

Isbn und Lfdnr der Isbn (sofern vorhanden)

sortiert nach Buchnr, Lfdnr

```
BUCHNR      ERSCHJ ISBN       LFDNR
----------- ------ ---------- -----
          1      - -              -
          2      - -              -
          5  1988, -              -
          6  1988, -              -
          7  1989, -              -
          8  1977, 3472864303    1,
          8  1977, 34728643yx    2,
          9  1990, -              -
         11  1990, -              -
         12  1989, 0201501139    1,
         18  1989, -              -
         27      - -              -

  12 Satz/Sätze ausgewählt.
```

8.6.2 leftjoin2

Buchnr, Erschj aller Bücher,

Isbn und Lfdnr der Isbn (sofern vorhanden),

Autornr und Lfdnr des Autors sofern vorhanden

sortiert nach Buchnr, Lfdnr der Isbn, Autornr, Lfdnr des Autors

```
BUCHNR      ERSCHJ ISBN       ILFDNR AUTORNR     ALFDNR
----------- ------ ---------- ------ ----------- ------
          1      - -               -         100     1,
          1      - -               -         200     2,
          2      - -               -         200     0,
          5  1988, -               -           1     0,
          6  1988, -               -           2     0,
          7  1989, -               -           3     0,
          8  1977, 3472864303     1,          2     0,
          8  1977, 34728643yx     2,          2     0,
          9  1990, -               -          10     1,
          9  1990, -               -          11     2,
         11  1990, -               -           -      -
         12  1989, 0201501139     1,          20     1,
         12  1989, 0201501139     1,          21     2,
         18  1989, -               -           -      -
         27      - -               -           -      -

  15 Satz/Sätze ausgewählt.
```

8.6.3 Leftjoin3

Erschj, Buchnr aller Bücher,

Autor, Autornr und Lfdnr des Autors sofern vorhanden

sortiert nach Erschj, Buchnr, Autor, Autornr, Lfdnr des Autors

```
ERSCHJ BUCHNR      AUTOR      AUTORNR     ALFDNR
------ ----------- ---------- ----------- ------
 1977,           8 Grass                2      0,
 1988,           5 Boell                1      0,
 1988,           6 Grass                2      0,
 1989,           7 Eco                  3      0,
 1989,          12 C. J. Date          20      1,
 1989,          12 Colin J. W          21      2,
 1989,          18 -                    -      -
 1990,           9 Emil Hack           10      1,
 1990,           9 Frieda Hol          11      2,
 1990,          11 -                    -      -
    -            1 BUSCH              100      1,
    -            1 BUSCH              200      2,
    -            2 BUSCH              200      0,
    -           27 -                    -      -

  14 Satz/Sätze ausgewählt.
```

8.6.4 Leftjoin4

Autor, Autornr aller Autoren,

Erschj, Buchnr und Lfdnr des Autors sofern vorhanden

sortiert nach Autor, Autornr, Buchnr, Lfdnr des Autors

```
AUTOR      AUTORNR     ERSCHJ BUCHNR      ALFDNR
---------- ----------- ------ ----------- ------
Boell                1  1988,           5      0,
BUSCH              100      -           1      1,
BUSCH              200      -           1      2,
BUSCH              200      -           2      0,
C. J. Date          20  1989,          12      1,
Colin J. W          21  1989,          12      2,
Eco                  3  1989,           7      0,
Emil Hack           10  1990,           9      1,
Frieda Hol          11  1990,           9      2,
Grass                2  1988,           6      0,
Grass                2  1977,           8      0,
Scheifele            6      -           -      -

  12 Satz/Sätze ausgewählt.
```

8.7 GROUP BY und Join

8.7.1 GROUP BY und Join, Primärschlüssel und Fremdschlüssel

TABLE Tbuch

buchnr	titel
1	C
2	C
5	Ansichten eines Clowns
6	die Blechtrommel
7	der Name der Rose
8	der Butt
9	DB2 fuer Sie
11	Elvis in Heidelberg
12	a guide to db2
18	Database Systems
27	die Jüdin von Toledo

TABLE Tvautor

buchnr	autornr	lfdnr	praemie
1	100	1	NULL
1	200	2	NULL
2	200	0	NULL
5	1	0	NULL
6	2	0	NULL
7	3	0	NULL
8	2	0	20.000
9	10	1	10.000
9	11	2	498.000
12	20	1	30.000
12	21	2	NULL

Formulieren Sie die SELECT-Anweisung, die zu Buchnr und Titel die Summe der Praemien, den Durchschnitt der Praemien und die Anzahl der Autoren anlistet.

buchnr	titel	sumbu	avgbu	anzau
1	C	NULL	NULL	2
2	C	NULL	NULL	1
5	Ansic	NULL	NULL	1
6	die B	NULL	NULL	1
7	der N	NULL	NULL	1
8	der B	20.000	20.000000	1
9	DB2 f	508.000	254.000000	2
12	a gui	30.000	30.000000	2

--Lösung A: (tbuch join tvautor) group by

--Lösung B: tbuch join (tvautor group by) xxx

--Lösung C: WITH xxx AS

```
WITH
xxx AS
 (
select
  tvautor.buchnr                                  as buchnr
, sum(tvautor.praemie)                            as sumbu
, avg(tvautor.praemie)                            as avgbu
, count(tvautor.autornr)                          as anzau
from tvautor group by tvautor.buchnr
)
 select
  tbuch.buchnr                 as buchnr
, SUBSTRING(tbuch.titel, 1,5) as titel
----SUBSTR DB2 und Oracle
, xxx.sumbu
, xxx.avgbu
, xxx.anzau
from tbuch inner join xxx
on tbuch.buchnr= xxx.buchnr
order by buchnr
;
```

8.7.2 GROUP BY und Join, „eine Falle“

--Liste1: Buchnr, Titel und Anzahl der Isbn’s

```
buchnr      titel anzisbn
----------- ----- -----------
8           der B 2
12          a gui 1
```

--Liste2: Buchnr, Titel, Anzahl der Autoren, Summe und Durchschnitt der Praemien pro Buch

```
buchnr      titel anzautor    sumpraemie                    avgpraemie
----------- ----- ----------- ----------------------------- -------------
1           C     2           NULL                          NULL
2           C     1           NULL                          NULL
5           Ansic 1           NULL                          NULL
6           die B 1           NULL                          NULL
7           der N 1           NULL                          NULL
8           der B 1           20.000                        20.000000
9           DB2 f 2           508.000                       254.000000
12          a gui 2           30.000                        30.000000
```

--Liste3 und Anweisung3: Warum ist die folgende Select-Anweisung und die Liste nicht in Ordnung?

```
select
 tbuch.buchnr                      as buchnr
,SUBSTRING(tbuch.titel, 1, 5)      as titel
----SUBSTR DB2 und Oracle
,count(tvautor.autornr)            as anzautor
,sum(tvautor.praemie)              as sumpraemie
,avg(tvautor.praemie)              as avgpraemie
,count(tisbn.buchnr )              as anzisbn
from tbuch inner join tisbn
     on tbuch.buchnr = tisbn.buchnr
           inner join tvautor
     on tbuch.buchnr = tvautor.buchnr
group by tbuch.buchnr, tbuch.titel
;
buchnr      titel anzautor    sumpraemie     avgpraemie   anzisbn
----------- ----- ----------- -------------- ------------ ------
8           der B 2           40.000         20.000000    2
12          a gui 2           30.000         30.000000    2
```

Antwort zu Liste3 und Anweisung3:

Das Buchnr 8 hat zwei ISBN's aber einen Autor!

Das Buchnr 12 hat eine ISBN aber zwei Autoren!

Die Summe der Praemien von Buchnr 8 ist nicht korrekt!

Schreiben sie eine Select-Anweisung, die die korrekte Liste liefert.

```
buchnr      titel anzautor    sumpraemie     avgpraemie   anzisbn
----------- ----- ----------- -------------- ------------ ------
8           der B 1           20.000         20.000000    2
12          a gui 2           30.000         30.000000    1
```

8.7.3 GROUP BY "umfassend und verständlich?"

Für alle Bücher, die nach 1977 erschienen sind und bei denen die Summe der Autorenprämie über 500 liegt, hole **maximale Autorenprämie, Titel, Buchnr, Preis** sortiert nach maximaler Prämie, Titel und Buchnr.

Wir entwickeln die SELECT-Anweisung Schritt für Schritt.

Projektion und Selektion

```
SELECT
      Tbuch.Buchnr
     ,Tbuch.Preis
     ,Tbuch.Titel
FROM Tbuch
WHERE Tbuch.Erschj > 1977
;
```

Natural Join

```
SELECT
      Tbuch.Buchnr
     ,Tbuch.Preis
     ,Tbuch.Titel
     ,Tvautor.Praemie
    ,Tvautor.Autornr
FROM  Tbuch,Tvautor
WHERE Tbuch.Buchnr = Tvautor.Buchnr
AND   Tbuch.Erschj > 1977
;
```

Verdichtung mit GROUP BY, maximale Autorenprämie, Summe der Autorenprämie

```
SELECT
      Tbuch.Buchnr
     ,Tbuch.Preis
     ,Tbuch.Titel
     ,MAX(Tvautor.Praemie) AS Maxprae
     ,SUM(Tvautor.Praemie) AS Sumprae
FROM  Tbuch,Tvautor
WHERE Tbuch.Buchnr = Tvautor.Buchnr
AND   Tbuch.Erschj > 1977
GROUP BY Tbuch.Buchnr, Tbuch.Preis, Tbuch.Titel
;
```

Auswahl und Anordnung der Spalten, Auswahl von Zeilen mit HAVING und Anordnung der Zeilen mit ORDER BY

```
--Variante mit HAVING
SELECT
     MAX(Tvautor.Praemie) AS Maxprae
    ,Tbuch.Titel          AS Titel
    ,Tbuch.Buchnr         AS Buchnr
    ,Tbuch.Preis          AS Preis
FROM  Tbuch,Tvautor
WHERE Tbuch.Buchnr = Tvautor.Buchnr
AND   Tbuch.Erschj > 1977
GROUP BY Tbuch.Buchnr, Tbuch.Preis, Tbuch.Titel
HAVING SUM(Tvautor.Praemie) > 500.00
ORDER BY Maxprae, Titel, Buchnr
;
--Variante mit WHERE
SELECT tzwi.maxprae       AS maxprae
      ,Tzwi.Titel         AS Titel
      ,Tzwi.Buchnr        AS Buchnr
      ,Tzwi.Preis         AS Preis
FROM
(
    SELECT
     Tbuch.Buchnr         AS Buchnr
    ,Tbuch.Preis          AS Preis
    ,Tbuch.Titel          AS Titel
    ,MAX(Tvautor.Praemie) AS Maxprae
    ,SUM(Tvautor.Praemie) AS Sumprae
    FROM  Tbuch,Tvautor
    WHERE Tbuch.Buchnr = Tvautor.Buchnr
    AND   Tbuch.Erschj > 1977
    GROUP BY Tbuch.Buchnr, Tbuch.Preis, Tbuch.Titel
) Tzwi
WHERE Sumprae > 500.00
ORDER BY Maxprae, Titel, Buchnr
;
```

8.8 Subquery mit IN, Subquery mit EXISTS, Variante mit Join

Schreiben Sie eine SELECT-Anweisung

1 mit IN-Subquery

2 mit EXISTS-Subquery

3 mit Join und Projektion

8.8.1 Verlagnr, Verlag von Verlagen mit wenigstens einem Buch

```
verlagnr    verlag
----------- --------------------
1111        Forkel
```

8.8.2 Verlagnr, Verlag von Verlagen ohne Buch

```
INSERT INTO Tverlag( verlagnr, verlag)
             VALUES( 2222, 'RORORO'  )
;
----Wir fügen einen Verlag ohne Bücher ein.
----Achtung: tbuch.verlagnr kann eine NULL präsentieren

verlagnr    verlag
----------- --------------------
2222        rororo
```

8.8.3 Buchnr, Preis und Titel des Buches mit der ISBN 3472864303.

```
BUCHNR      PREIS     TITEL
----------- --------- --------------------
          8      0,50 der Butt
```

8.8.4 Buchnr, Preis und Titel der Bücher die wenigstens einen Autor haben.

```
BUCHNR      PREIS TITEL
---------- ---------- ------------------------------
         1 null       C
         2 null       C
         5        3,5 Ansichten eines Clowns
         6       20,5 die Blechtrommel
         7      99,99 der Name der Rose
         8         ,5 der Butt
         9         55 DB2 fuer Sie
        12 null       a guide to db2
```

8.8.5 Buchnr, Preis und Titel der Bücher die Autor 2 geschrieben hat.

```
    BUCHNR      PREIS TITEL
---------- ---------- ------------------------------
         6       20,5 die Blechtrommel
         8         ,5 der Butt
```

8.8.6 Buchnr, Preis und Titel der Bücher der Autoren mit Namen Grass.

```
BUCHNR      PREIS     TITEL
----------- --------- ------------------------------
          6     20,50 die Blechtrommel
          8      0,50 der Butt
```

8.8.7 Buchnr, Preis und Titel der Bücher ohne Autor.

```
BUCHNR      PREIS             TITEL
----------- ----------------- --------------------
11          NULL              Elvis in Heidelberg
18          99.99             Database Systems
27          99.99             die Jüdin von Toledo
```

8.8.8 Buchnr, Preis und Titel der Bücher ohne Praemie.

```
BUCHNR      preis      TITEL
----------- ---------- ------------------------
1           NULL       C
2           NULL       C
5           3.50       Ansichten eines Clowns
6           20.50      die Blechtrommel
7           99.99      der Name der Rose
11          NULL       Elvis in Heidelberg
18          99.99      Database Systems
27          99.99      die Jüdin von Toledo
```

8.8.9 Buchnr, Preis und Titel der Bücher ohne Praemie und ohne Isbn.

```
BUCHNR      preis      TITEL
----------- ---------- ------------------------
1           NULL       C
2           NULL       C
5           3.50       Ansichten eines Clowns
6           20.50      die Blechtrommel
7           99.99      der Name der Rose
11          NULL       Elvis in Heidelberg
18          99.99      Database Systems
27          99.99      die Jüdin von Toledo

--insert into Tisbn(buchnr, isbn, lfdnr)
--values ( 27, 'XXX' , 1);
--delete from Tisbn where Buchnr = 27;
```

8.8.10 Buchnr, Preis und Titel der Bücher ohne Praemie oder ohne Isbn.

```
BUCHNR      preis      TITEL
----------- ---------- ------------------------
1           NULL       C
2           NULL       C
5           3.50       Ansichten eines Clowns
6           20.50      die Blechtrommel
7           99.99      der Name der Rose
9           55.00      DB2 fuer Sie
11          NULL       Elvis in Heidelberg
18          99.99      Database Systems
27          99.99      die Jüdin von Toledo
```

8.9 Skalare Operatoren und Funktionen, Testen Sie ihren Server

8.9.1 Aufgabe

Ein String mit maximal 17 Ziffern vor dem Dezimalkomma
und maximal 5 Ziffern nach dem Dezimalkomma
soll gerundet auf 2 Nachkommastellen rechtsbündig
mit Vorzeichen und Dezimalpunkt
als String der Länge 33 präsentiert werden.

```
schl        wert                     wertrechts
----------- ------------------------ ---------------------------------
1           -12345678901234567,98764             -12345678901234567.99
2           +12345678901234567,98764             +12345678901234567.99
3               +3,50                                            +3.50
4               -3,50                                            -3.50
5                3,50                                            +3.50
6              ,456                                              +0.46
7              ,45                                               +0.45
8              ,4                                                +0.40
9                   +,888                                        +0.89
10              +,88                                             +0.88
11                   +,8                                         +0.80
12             -,777                                             -0.78
13                   -,77                                        -0.77
14             -,7                                               -0.70
```

Auf den folgenden Seiten finden Sie die SELECT-Anweisungen für SQL Server, DB2 bzw. Oracle.

Analysieren und testen Sie diese Anweisungen Schritt für Schritt.

```
drop table tnumber
;
create table tnumber
(
  schl integer  not null
 ,wert char(24) NOT NULL
 ,primary key (schl)
)
;
delete from tnumber
;
insert into tnumber (schl, wert) values
( 1 , '-12345678901234567,98764');
insert into tnumber (schl, wert) values
( 2 , '+12345678901234567,98764');

insert into tnumber (schl, wert) values
( 3, '    +3,50');
insert into tnumber (schl, wert) values
( 4, '    -3,50');
insert into tnumber (schl, wert) values
( 5, '     3,50');
insert into tnumber (schl, wert) values
( 6 , '  ,456  ');
insert into tnumber (schl, wert) values
( 7 , '  ,45  ');
insert into tnumber (schl, wert) values
( 8 , '  ,4 ');
insert into tnumber (schl, wert) values
( 9 , '       +,888  ');
insert into tnumber (schl, wert) values
( 10, '   +,88 ');
insert into tnumber (schl, wert) values
( 11 , '        +,8 ');
insert into tnumber (schl, wert) values
( 12 , '  -,777  ');
insert into tnumber (schl, wert) values
( 13 , '        -,77 ');
insert into tnumber (schl, wert) values
( 14 , '  -,7 ');
select * from tnumber
;
--weitere Testfälle
insert into tnumber (schl, wert) values
( 21, '    -0,0     ');
insert into tnumber (schl, wert) values
( 22, '    +0,0     ');
insert into tnumber (schl, wert) values
( 23, '     0,0     ');
insert into tnumber (schl, wert) values
( 24, '    -,0      ');
insert into tnumber (schl, wert) values
( 25, '    +,0      ');
insert into tnumber (schl, wert) values
( 26, '     ,0      ');
select * from tnumber;
```

8.9.2 SQL Server

```
WITH
zwitab0 AS
(
select schl
,wert
,REPLACE(wert, ',', '.') as wert0
 from tnumber
)
----select * from zwitab0;
,
zwitab1 as
(select schl
,wert
,RTRIM(LTRIM(wert0)) as wert1
from zwitab0
)
----select * from zwitab1;
,
zwitab2 as
(select schl
,wert
,CAST( wert1 AS DECIMAL(31,5))  as wert2
from zwitab1
)
---select * from zwitab2;
,
zwitab3 as
 ( select
   schl
  ,wert
  ,CAST( ROUND(wert2,2) as decimal(31,2)) as wert3
  from zwitab2
)
----select * from zwitab3;
,
zwitab4 as
(
select
  schl
, wert
, cast(wert3 as varchar(100))as wert4
 FROM zwitab3
)
---select * from zwitab4;
,
zwitab5 as
 (
select
  schl
 ,wert
 ,case
  when substring(wert4 , 1, 1) <> '-'
  then '+' + wert4
  else       wert4
  end as wert5
from zwitab4
)
```

```
---- select * from zwitab5;
,
zwitab6 as
(
select
schl
,wert
,wert5      as wert6
,LEN(wert5) as laenge6
from zwitab5
)
----select * from zwitab6;
,
zwitab7 as
(
select
 schl
,wert
,SUBSTRING( CAST( REPLICATE(' ', 1000) AS CHAR(33))
            , 1, 33-laenge6)
  + wert6                          as wert7
from zwitab6
)
select schl, wert, wert7 as wertrechts
from zwitab7
order by schl
;
```

8.9.3 DB2

```
---- sql server>>> DB2
---- substring >>> SUBSTR
---- len       >>> LENGTH
---- replicate >>> REPEAT
---- +         >>> ||
WITH
zwitab0 AS
(
select schl
,wert
,REPLACE(wert, ',', '.') as wert0
 from tnumber
)
---- select * from zwitab0;
,
zwitab1 as
(select schl
,wert
,RTRIM(LTRIM(wert0)) as wert1
from zwitab0
)
---- select * from zwitab1;
,
zwitab2 as
(select schl
,wert
,CAST( wert1 AS DECIMAL(31,5))  as wert2
from zwitab1
)
----select * from zwitab2;
,
zwitab3 as
 ( select
   schl
  ,wert
  ,CAST( ROUND(wert2,2) as decimal(31,2)) as wert3
  from zwitab2
)
---- select * from zwitab3;
,
zwitab4 as
(
select
  schl
, wert
, cast(wert3 as varchar(100))as wert4
 FROM zwitab3
)
----select * from zwitab4;
,
zwitab5 as
 (
select
  schl
 ,wert
 ,case
  when SUBSTR(wert4 , 1, 1) <> '-'
```

```
  then '+' || wert4
  else       wert4
  end as wert5
from zwitab4
)
---- select * from zwitab5;
,
zwitab6 as
(
select
schl
,wert
,wert5      as wert6
,LENGTH(wert5) as laenge6
from zwitab5
)
----select * from zwitab6;
,
zwitab7 as
(
select
 schl
,wert
,SUBSTR( CAST( REPEAT(' ', 1000) AS CHAR(33))
            , 1, 33-laenge6)
  || wert6                          as wert7
from zwitab6
)
select schl, wert, wert7 as wertrechts
from zwitab7
order by schl
;
```

8.9.4 Oracle

```
  WITH
zwitab0 AS
(
select schl
,wert
,REPLACE(wert, ',', '.') as wert0
 from tnumber
)
---- select * from zwitab0
,
zwitab1 as
(
 select schl
,wert
,RTRIM(LTRIM(wert0)) as wert1
,length(RTRIM(LTRIM(wert0)))  as laengewert1
from zwitab0
)
---- select * from zwitab1
,
zwitab2 as
(
select schl
,wert
,case when substr(wert1, 1, 1) = '.'
           then '0' || wert1
      when substr(wert1, 1, 2) = '-.'
           then REPLACE(wert1, '-.', '-0.')
      when substr(wert1, 1, 2) = '+.'
           then REPLACE(wert1, '+.', '0.')
      when wert1='0.'
           then '0.0'
      when substr(wert1, 1, 1) = '+'
      then substr(wert1, 2, laengewert1-1)
      else wert1
end   AS wert2
from zwitab1
)
---- select * from zwitab2
,
zwitab3 as
(select schl
,wert
,wert2
,TO_NUMBER( wert2 , '99999999999999999990.99999') as wert3
from zwitab2
)
----select * from zwitab3
,
zwitab4 as
(
select
schl
,wert
,CAST( ROUND(wert3,2) as decimal(31,2)) as wert4
from zwitab3
)
```

```
----select * from zwitab4
,
zwitab5 as
(
select
schl
,wert
,TO_CHAR(wert4, 'S9999999999999999990.99') as wert5
from zwitab4
)
select
schl
,wert
,wert5
from zwitab5;
```

9

Lösungen Interaktive SQL

9 Lösungen Interaktive SQL

9.1 SELECT

9.1.1 Liste1

Buchnr, Erschj, Preis, Titel

sortiert nach Buchnr aufsteigend

```
SELECT Buchnr, Erschj, Preis, Titel
FROM Tbuch
ORDER BY Buchnr
;
    BUCHNR     ERSCHJ      PREIS TITEL
---------- ---------- ---------- ----------------------
         1 null       null       C
         2 null       null       C
         5       1988        3,5 Ansichten eines Clowns
         6       1988       20,5 die Blechtrommel
         7       1989      99,99 der Name der Rose
         8       1977         ,5 der Butt
         9       1990         55 DB2 fuer Sie
        11       1990 null       Elvis in Heidelberg
        12       1989 null       a guide to db2
        18       1989      99,99 Database Systems
        27 null            99,99 die Jüdin von Toledo
```

9.1.2 Liste2

Erschj, Preis, Buchnr

sortiert nach Erschj aufsteigend, Preis aufsteigend, Buchnr aufsteigend

```
SELECT Erschj, Preis, Buchnr
FROM Tbuch
ORDER BY Erschj ASC, Preis ASC, Buchnr ASC
;
```

Oracle und DB2

```
    ERSCHJ      PREIS     BUCHNR
---------- ---------- ----------
      1977         ,5          8
      1988        3,5          5
      1988       20,5          6
      1989      99,99          7
      1989      99,99         18
      1989 null               12
      1990         55          9
      1990 null               11
null            99,99         27
null       null                 1
null       null                 2
```

SQL Server

```
Erschj Preis     Buchnr
------ --------- -----------
NULL   NULL      1
NULL   NULL      2
NULL   99.99     27
1977   .50       8
1988   3.50      5
1988   20.50     6
1989   NULL      12
1989   99.99     7
1989   99.99     18
1990   NULL      11
1990   55.00     9
```

9.1.3 Liste3

Erschj, Preis, Buchnr

sortiert nach Erschj absteigend, Preis absteigend, Buchnr absteigend

```
SELECT Erschj, Preis, Buchnr
FROM Tbuch
ORDER BY Erschj DESC, Preis DESC, Buchnr DESC
;
```

Oracle und DB2

```
    ERSCHJ      PREIS     BUCHNR
---------- ---------- ----------
null       null                2
null       null                1
null            99,99         27
      1990 null                11
      1990         55          9
      1989 null                12
      1989      99,99         18
      1989      99,99          7
      1988       20,5          6
      1988        3,5          5
      1977         ,5          8
```

SQL Server

```
Erschj Preis     Buchnr
------ --------- -----------
1990   55.00     9
1990   NULL      11
1989   99.99     18
1989   99.99     7
1989   NULL      12
1988   20.50     6
1988   3.50      5
1977   .50       8
NULL   99.99     27
NULL   NULL      2
NULL   NULL      1
```

9.1.4 Liste4

Erschj, Preis, Buchnr

sortiert nach Preis absteigend, Erschj aufsteigend, Buchnr aufsteigend

```
SELECT Erschj, Preis, Buchnr
FROM Tbuch
ORDER  BY Preis DESC, Erschj ASC,  Buchnr ASC
;
```

Oracle und DB2

```
    ERSCHJ      PREIS     BUCHNR
---------- ---------- ----------
      1989 null               12
      1990 null               11
null       null                1
null       null                2
      1989      99,99          7
      1989      99,99         18
null            99,99         27
      1990         55          9
      1988       20,5          6
      1988        3,5          5
      1977         ,5          8
```

SQL Server

```
Erschj Preis     Buchnr
------ --------- -----------
NULL   99.99     27
1989   99.99     7
1989   99.99     18
1990   55.00     9
1988   20.50     6
1988   3.50      5
1977   .50       8
NULL   NULL      1
NULL   NULL      2
1989   NULL      12
1990   NULL      11
```

9.1.5 Liste5

Titel, Buchnr

sortiert nach Titel, Buchnr

DB2 for Windows

```
select titel, buchnr from tbuch
order by titel, buchnr
;
```

```
TITEL                          BUCHNR
------------------------------ -----------
Ansichten eines Clowns                   5
C                                        1
C                                        2
DB2 fuer Sie                             9
Database Systems                        18
Elvis in Heidelberg                     11
a guide to db2                          12
der Butt                                 8
der Name der Rose                        7
die Blechtrommel                         6
die Jüdin von Toledo                    27
```

Oracle

```
alter session set nls_sort = german;
alter session set nls_comp = linguistic;
select titel, buchnr from tbuch
order by titel, buchnr
;
```

```
TITEL                            BUCHNR
------------------------------ ----------
a guide to db2                         12
Ansichten eines Clowns                  5
C                                       1
C                                       2
Database Systems                       18
DB2 fuer Sie                            9
der Butt                                8
der Name der Rose                       7
die Blechtrommel                        6
die Jüdin von Toledo                   27
Elvis in Heidelberg                    11
```

SQL Server

```
select titel, buchnr from tbuch
order by titel, buchnr
;
```

```
titel                     buchnr
------------------------- -----------
a guide to db2            12
Ansichten eines Clowns    5
C                         1
C                         2
Database Systems          18
DB2 fuer Sie              9
der Butt                  8
der Name der Rose         7
die Blechtrommel          6
die Jüdin von Toledo      27
Elvis in Heidelberg       11
```

9.1.6 Liste6

Erschj, Preis

Ausprägungen dieser Kombination

sortiert nach Erschj, Preis

```
SELECT DISTINCT
  Erschj, Preis
FROM Tbuch
ORDER  BY Erschj, Preis
;
```

Oracle und DB2

```
ERSCHJ      PREIS
---------- ----------
      1977         ,5
      1988        3,5
      1988       20,5
      1989      99,99
      1989 null
      1990         55
      1990 null
null            99,99
null       null
```

SQL Server

```
Erschj Preis
------ ---------
NULL   NULL
NULL   99.99
1977   .50
1988   3.50
1988   20.50
1989   NULL
1989   99.99
1990   NULL
1990   55.00
```

9.1.7 Liste7

Preis

Ausprägungen von Preisen

sortiert absteigend

```
SELECT DISTINCT
  Preis
FROM Tbuch
ORDER  BY Preis DESC
;
```

Oracle und DB2

```
     PREIS
----------
null
     99,99
        55
      20,5
       3,5
        ,5

6 rows selected.
```

SQL Server

```
Preis
---------
99.99
55.00
20.50
3.50
.50
NULL

(6 row(s) affected)
```

9.1.8 Liste8

Buchnr, Titel

von Büchern, die 1988 erschienen sind

sortiert nach Buchnr

```
SELECT Buchnr, Titel
FROM Tbuch
WHERE Erschj = 1988
ORDER BY Buchnr
;
    BUCHNR TITEL
---------- --------------------------------------------------
         5 Ansichten eines Clowns
         6 die Blechtrommel
```

9.1.9 Liste9

Buchnr, Titel

von Büchern, die nicht 1988 erschienen sind

sortiert nach Buchnr aufsteigend

```
SELECT Buchnr, Titel
FROM Tbuch
WHERE
   (  NOT (Erschj = 1988) OR   Erschj IS NULL  )
ORDER BY Buchnr
;
    BUCHNR TITEL
---------- --------------------------------------------------
         1 C
         2 C
         7 der Name der Rose
         8 der Butt
         9 DB2 fuer Sie
        11 Elvis in Heidelberg
        12 a guide to db2
        18 Database Systems
        27 die Jüdin von Toledo

9 rows selected.
```

9.1.10 Liste10

Buchnr

von Büchern, die 3.50 oder 55.00 kosten

sortiert nach Buchnr aufsteigend

```
SELECT Buchnr
FROM Tbuch
WHERE Preis = 3.50 OR Preis = 55.00
ORDER BY Buchnr
;
BUCHNR
----------
         5
         9
```

9.1.11 Liste11

Buchnr

aller Bücher von 1977 *und* 1988

sortiert nach Buchnr aufsteigend

(Achtung: „und“ ist Alltagssprache)

```
SELECT Buchnr
FROM Tbuch
WHERE Erschj = 1977 OR Erschj = 1988
ORDER BY Buchnr
;
BUCHNR
----------
         5
         6
         8
```

9.1.12 Liste12

Buchnr, Erschj, Preis

aller Bücher von 1990 oder teurer als 50.00

sortiert nach Buchnr aufsteigend

```
SELECT Buchnr, Erschj, Preis
FROM Tbuch
WHERE Erschj = 1990 OR Preis > 50.00
ORDER BY Buchnr
;
    BUCHNR     ERSCHJ      PREIS
---------- ---------- ----------
         7       1989      99,99
         9       1990         55
        11       1990 null
        18       1989      99,99
        27 null            99,99
```

9.1.13 Liste13

Buchnr, Erschj, Preis

aller Bücher von 1989 und mit Preis 99.99

sortiert nach Buchnr aufsteigend

```
SELECT Buchnr, Erschj, Preis
FROM Tbuch
WHERE      (Erschj = 1989 AND Preis = 99.99)
ORDER BY Buchnr
;

    BUCHNR     ERSCHJ      PREIS
---------- ---------- ----------
         7       1989      99,99
        18       1989      99,99

2 rows selected.
```

9.1.14 Liste14

Buchnr, Erschj, Preis

aller Bücher von 1989 oder mit Preis 99.99

sortiert nach Buchnr aufsteigend

```
SELECT Buchnr, Erschj, Preis
FROM Tbuch
WHERE      (Erschj = 1989 OR Preis = 99.99)
ORDER BY Buchnr
;

    BUCHNR     ERSCHJ      PREIS
---------- ---------- ----------
         7       1989      99,99
        12       1989 null
        18       1989      99,99
        27 null            99,99

4 rows selected.
```

9.1.15 Liste15

Buchnr, Erschj, Preis

aller Bücher von 1988 bzw. 1989 bzw. 1990

```
select buchnr, erschj, preis
from tbuch
where (erschj = 1988 or erschj = 1989 or erschj = 1990)
;
select buchnr, erschj, preis
from tbuch
where (erschj IN (1988,1989,1990))
;
buchnr      erschj preis
----------- ------ ---------
5           1988   3.50
6           1988   20.50
7           1989   99.99
9           1990   55.00
11          1990   NULL
12          1989   NULL
18          1989   99.99
```

9.1.16 Liste16

Buchnr, Erschj, Preis

aller Bücher von 1988 bzw. 1989 bzw. 1990 und Preis über 10 Euro

```
select buchnr, erschj, preis
from tbuch
where
       (erschj = 1988 or erschj = 1989 or erschj = 1990)
   and
      (preis > 10.00)
;
buchnr      erschj preis
----------- ------ ---------
6           1988   20.50
7           1989   99.99
9           1990   55.00
18          1989   99.99
```

9.1.17 Liste17

Buchnr, Erschj, Preis

aller Bücher von 1988 bzw. 1989 bzw. 1990 und Preis nicht über 10 Euro

Variante IST

```
select buchnr, erschj, preis
from tbuch
where
       (erschj = 1988 or erschj = 1989 or erschj = 1990)
   and
        NOT (preis > 10.00)
;
buchnr      erschj preis
----------- ------ ---------
5           1988   3.50
```

Variante SOLL (mit Berücksichtigung der NULL)

```
select buchnr, erschj, preis
from tbuch
where
       (erschj = 1988 or erschj = 1989 or erschj = 1990)
   and
       ( NOT (preis > 10.00) OR preis IS NULL)
;
buchnr      erschj preis
----------- ------ ---------
5           1988   3.50
11          1990   NULL
12          1989   NULL
```

9.1.18 Liste18

Buchnr, Erschj, Preis

aller Bücher von 1988 bzw. 1989 bzw. 1990 oder Preis über 10 Euro

```
select buchnr, erschj, preis
from tbuch
where
       (erschj = 1988 or erschj = 1989 or erschj = 1990)
   OR
      (preis > 10.00)
;
buchnr      erschj preis
----------- ------ ---------
5           1988   3.50
6           1988   20.50
7           1989   99.99
9           1990   55.00
11          1990   NULL
12          1989   NULL
18          1989   99.99
27          NULL   99.99

(8 row(s) affected)
```

9.1.19 Liste19

Buchnr, Erschj, Preis

aller Bücher von 1988 bzw. 1989 bzw. 1990 oder Preis nicht über 10 Euro

Variante IST

```
select buchnr, erschj, preis
from tbuch
where
       (erschj = 1988 or erschj = 1989 or erschj = 1990)
   OR
        NOT(preis > 10.00)
;
buchnr      erschj preis
----------- ------ ---------
5           1988   3.50
6           1988   20.50
7           1989   99.99
8           1977   .50
9           1990   55.00
11          1990   NULL
12          1989   NULL
18          1989   99.99

(8 row(s) affected)
```

Variante SOLL

```
select buchnr, erschj, preis
from tbuch
where
       (erschj = 1988 or erschj = 1989 or erschj = 1990)
   OR
       ( NOT(preis > 10.00) OR preis IS NULL )
;
buchnr      erschj preis
----------- ------ ---------
1           NULL   NULL
2           NULL   NULL
5           1988   3.50
6           1988   20.50
7           1989   99.99
8           1977   .50
9           1990   55.00
11          1990   NULL
12          1989   NULL
18          1989   99.99

(10 row(s) affected)
```

9.1.20 Liste20

9.1.21 Liste21

Preis, Buchnr

aller Bücher mit Preis zwischen 20.50 und 99.99

sortiert nach Preis aufsteigend, Buchnr aufsteigend

```
SELECT Preis, Buchnr
FROM Tbuch
WHERE Preis BETWEEN 20.50 AND 99.99
ORDER BY Preis, Buchnr
;

PREIS     BUCHNR
---------- ----------
      20,5          6
        55          9
     99,99          7
     99,99         18
     99,99         27

5 rows selected.
```

9.1.22 Liste22

9.1.23 Liste23

9.1.24 Liste24

Titel, Buchnr

aller Bücher deren Titel mit A, B, C oder D anfängt

sortiert nach Titel, Buchnr

Oracle und DB2

```
SELECT SUBSTR(Titel, 1,30) AS Titel, Buchnr
FROM Tbuch
WHERE Titel LIKE 'A%'
   OR Titel LIKE 'B%'
   OR Titel LIKE 'C%'
   OR Titel LIKE 'D%'
ORDER BY Titel, Buchnr
;

TITEL                            BUCHNR
------------------------------ ----------
Ansichten eines Clowns                5
C                                     1
C                                     2
Database Systems                     18
DB2 fuer Sie                          9

5 rows selected.
```

SQL Server

```
SELECT SUBSTRING(Titel, 1,30) AS Titel, Buchnr
FROM Tbuch
WHERE Titel LIKE 'A%'
   OR Titel LIKE 'B%'
   OR Titel LIKE 'C%'
   OR Titel LIKE 'D%'
ORDER BY Titel, Buchnr
;
Titel                          Buchnr
------------------------------ -----------
a guide to db2                 12
Ansichten eines Clowns         5
C                              1
C                              2
Database Systems               18
DB2 fuer Sie                   9
der Butt                       8
der Name der Rose              7
die Blechtrommel               6
die Jüdin von Toledo           27

(10 row(s) affected)
```

9.1.25 Liste25

Titel, Buchnr

aller Bücher deren Titel der, die oder das enthält

sortiert nach Titel, Buchnr

Oracle und DB2

```
SELECT SUBSTR(Titel, 1,30) AS Titel, Buchnr
FROM Tbuch
WHERE Titel LIKE '%der%'
OR    Titel LIKE '%die%'
OR    Titel LIKE '%das%'
ORDER BY Titel, Buchnr
;
```

SQL Server

```
SELECT SUBSTRING(Titel, 1,30) AS Titel, Buchnr
FROM Tbuch
WHERE Titel LIKE '%der%'
OR    Titel LIKE '%die%'
OR    Titel LIKE '%das%'
ORDER BY Titel, Buchnr
;
```

```
TITEL                           BUCHNR
------------------------------ ----------
der Butt                                8
der Name der Rose                       7
die Blechtrommel                        6
die Jüdin von Toledo                   27
```

9.2 OR AND NOT – UNION INTERSECT EXCEPT

9.2.1 NOT B1 bzw. NOT B2

Aktuelle Daten von Tbuch:

```
buchnr      erschj                                 preis
----------- -------------------------------------- -------
100         1989                                   99.99
200         1989                                   22.22
300         2000                                   99.99
400         2000                                   22.22
----------- -------------------------------------- -------
500         1989                                   NULL
600         NULL                                   99.99
700         NULL                                   NULL
800         2000                                   NULL
900         NULL                                   22.22
```

B1: Tbuch.Erschj = 1989 B2: Tbuch.Preis = 99.99
SOLL: mit besonderer Berücksichtigung der NULL
Schreiben Sie die SELECT-Anweisungen

Bedingung für die Base-Table Tbuch		SOLL
B1 erschj=1989	100 200 500	
NOT B1		300 400 800 600 700 900
B2 Preis=99.99	100 300 600	
NOT B2		200 400 900 500 700 800

```
delete from tbuch;
INSERT INTO tbuch(buchnr,erschj, preis, titel) VALUES
 ( 100, 1989, 99.99, 'xyz');
INSERT INTO tbuch(buchnr,erschj, preis, titel) VALUES
 ( 200, 1989, 22.22, 'xyz');
INSERT INTO tbuch(buchnr,erschj, preis, titel) VALUES
 ( 300, 2000, 99.99, 'xyz');
INSERT INTO tbuch(buchnr,erschj, preis, titel) VALUES
 ( 400, 2000, 22.22, 'xyz');
INSERT INTO tbuch(buchnr,erschj, preis, titel) VALUES
 ( 500, 1989, NULL, 'xyz');
INSERT INTO tbuch(buchnr,erschj, preis, titel) VALUES
 ( 600, NULL, 99.99, 'xyz');
INSERT INTO tbuch(buchnr,erschj, preis, titel) VALUES
 ( 700, NULL, NULL, 'xyz');
INSERT INTO tbuch(buchnr,erschj, preis, titel) VALUES
 ( 800, 2000, NULL, 'xyz');
INSERT INTO tbuch(buchnr,erschj, preis, titel) VALUES
 ( 900, NULL, 22.22, 'xyz');
select buchnr, erschj, preis from tbuch order by buchnr;
```

```
--B1
--erschj = 1989
select buchnr, erschj, preis from tbuch
where Erschj = 1989
;
--NOT B1 --SOLL der Rest
select buchnr, erschj, preis from tbuch
where (not erschj = 1989 or erschj is null)
;
----äquivalent
  select buchnr, erschj, preis from tbuch
EXCEPT --MINUS  Oracle
  select buchnr, erschj, preis from tbuch
  where Erschj = 1989
;
------------------------------
--B2
--preis = 99.99
select buchnr, erschj, preis from tbuch
where preis = 99.99
;
--NOT B2 --SOLL der Rest
select buchnr, erschj, preis from tbuch
where (not preis = 99.99 or preis is null)
;
----äquivalent
  select buchnr, erschj, preis from tbuch
EXCEPT --MINUS  Oracle
  select buchnr, erschj, preis from tbuch
  where preis = 99.99
;
------------------------------
```

9.2.2 NOT (B1 AND B2) – (NOT B1) OR (NOT B2)

Aktuelle Daten von Tbuch:

buchnr	erschj	preis
100	1989	99.99
200	1989	22.22
300	2000	99.99
400	2000	22.22
500	1989	NULL
600	NULL	99.99
700	NULL	NULL
800	2000	NULL
900	NULL	22.22

B1: Tbuch.Erschj = 1989 B2: Tbuch.Preis = 99.99

SOLL: mit besonderer Berücksichtigung der NULL

Schreiben Sie die SELECT-Anweisungen

Bedingung für die Base-Table Tbuch		**SOLL**
B1 AND B2 1989 und 99.99	100	
NOT(B1 AND B2) äquivalent (NOT B1) OR (NOT B2)		200 300 400 500 600 700 800 900

```
--B1 AND B2
select buchnr, erschj, preis from tbuch
where erschj = 1989 AND preis = 99.99
;
--NOT (B1 AND B2) - (NOT B1) OR (NOT B2) --SOLL der Rest
select buchnr, erschj, preis from tbuch
where
NOT (Erschj=1989 AND Preis=99.99)
OR erschj IS NULL
OR preis  IS NULL
;
----äquivalent
select buchnr, erschj, preis from tbuch
where
 (erschj <> 1989 or erschj is null)
 OR
 (preis <> 99.99 or preis is null)
;
----äquivalent
  select buchnr, erschj, preis from tbuch
EXCEPT
  select buchnr, erschj, preis from tbuch
  where Erschj = 1989 and preis =99.99
;
```

9.2.3 NOT (B1 OR B2) – (NOT B1) AND (NOT B2)

Aktuelle Daten von Tbuch:

buchnr	erschj	preis
100	1989	99.99
200	1989	22.22
300	2000	99.99
400	2000	22.22
500	1989	NULL
600	NULL	99.99
700	NULL	NULL
800	2000	NULL
900	NULL	22.22

B1: Tbuch.Erschj = 1989 B2: Tbuch.Preis = 99.99

SOLL: mit besonderer Berücksichtigung der NULL

Schreiben Sie die SELECT-Anweisungen

Bedingung für die Base-Table Tbuch		SOLL
B1 OR B2 1989 oder 99.99	100 200 300 500 600	
NOT(B1 OR B2) äquivalent (NOT B1) AND (NOT B2)		400 700 800 900

```
--B1 OR B2
select buchnr, erschj, preis from tbuch
where erschj = 1989 OR preis = 99.99
;
--NOT (B1 OR B2) – (NOT B1) AND (NOT B2)--SOLL der Rest
select buchnr, erschj, preis from tbuch
where
 NOT (Erschj=1989 OR Preis=99.99        )
 OR  (Erschj IS NULL AND Preis IS NULL  )
 OR  (Erschj <> 1989 AND preis IS NULL  )
 OR  (Erschj IS NULL AND preis <> 99.99 )
;
----äquivalent
select buchnr, erschj, preis from tbuch
where
 (erschj <> 1989 or erschj is null)
 AND
 (preis <> 99.99 or preis is null)
;
----äquivalent
  select buchnr, erschj, preis from tbuch
EXCEPT
  select buchnr, erschj, preis from tbuch
  where Erschj = 1989 OR preis =99.99
;
```

9.2.4 entweder B1 oder B2 und die NULL?

Aktuelle Daten von Tbuch:

```
buchnr      erschj                                 preis
----------- -------------------------------------- -------
100         1989                                   99.99
200         1989                                   22.22
300         2000                                   99.99
400         2000                                   22.22
----------- -------------------------------------- -------
500         1989                                   NULL
600         NULL                                   99.99
700         NULL                                   NULL
800         2000                                   NULL
900         NULL                                   22.22
```

B1: Tbuch.Erschj = 1989 B2: Tbuch.Preis = 99.99

SOLL: mit besonderer Berücksichtigung der NULL

Schreiben Sie die SELECT-Anweisungen

Bedingung für die Base-Table Tbuch		**SOLL**
>>(entweder … oder)<< **>>entweder 1989 oder 99.99** **aber nicht beides gleichzeitig<<** (B1 OR B2) AND NOT (B1 AND B2) äquivalent (B1 AND (NOT B2)) OR (B2 AND (NOT B1))	200 300	
>>(entweder … oder) und die NULL?<<		200 300 500 600

```
--entweder … oder
--(B1 OR B2) AND NOT(B1 AND B2)
--(B1 AND (NOT B2)) OR (B2 AND (NOT B1))
select buchnr, erschj, preis from tbuch
where
    (
     (Erschj=1989 OR Preis=99.99)
      AND NOT
     (Erschj=1989 AND Preis=99.99)
    )
;
----äquivalent
select buchnr, erschj, preis from tbuch
where
  (erschj = 1989 and (preis <> 99.99))
OR
  (preis = 99.99 and (erschj<>1989))
;
```

```
--entweder … oder und die NULL?
select buchnr, erschj, preis from tbuch
where
    (
     (Erschj=1989 OR Preis=99.99)
      AND NOT
     (Erschj=1989 AND Preis=99.99)
    )
OR  (Erschj = 1989 and preis is null)
OR  (preis = 99.99 and erschj is null)
;
----äquivalent
select buchnr, erschj, preis from tbuch
where
  (erschj = 1989 and (preis <> 99.99 or preis IS NULL))
OR
  (preis = 99.99 and (erschj<>1989 or erschj IS NULL))
;
----äquivalent
     select buchnr, erschj, preis
     from tbuch
     WHERE Erschj = 1989 OR Preis = 99.99
EXCEPT --MINUS
     select buchnr, erschj, preis
     from tbuch
     WHERE Erschj = 1989 AND Preis = 99.99
;
```

9.3 MAX MIN AVG SUM COUNT und GROUP BY

9.3.1 Formulieren Sie Ihre Erwartung, testen Sie!

```
--AGG001a
SELECT abc.erschj        AS erschj
     , COUNT(abc.buchnr) AS countbuchnr
FROM TBUCH abc
WHERE abc.ERSCHJ IN (1977, 1989)
GROUP BY abc.erschj
;
--AGG001b
SELECT abc.erschj        AS erschj
     , COUNT(abc.buchnr) AS countbuchnr
FROM TBUCH abc
GROUP BY abc.erschj
HAVING abc.erschj IN (1977, 1989)
;
erschj                                   countbuchnr
---------------------------------------- -----------
1977                                     1
1989                                     3

--AGG002a
SELECT abc.erschj        AS erschj
     , COUNT(abc.buchnr) AS countbuchnr
FROM TBUCH abc
WHERE abc.ERSCHJ IN (1977, 1989, NULL)
GROUP BY abc.erschj
;
--AGG002b
SELECT abc.erschj        AS erschj
     , COUNT(abc.buchnr) AS countbuchnr
FROM TBUCH abc
GROUP BY abc.erschj
HAVING abc.erschj IN (1977, 1989, NULL)
;
erschj                                   countbuchnr
---------------------------------------- -----------
1977                                     1
1989                                     3

--AGG003a
SELECT abc.erschj        AS erschj
     , COUNT(abc.buchnr) AS countbuchnr
FROM TBUCH abc
WHERE abc.ERSCHJ IN (1977, 1989) OR abc.erschj IS NULL
GROUP BY abc.erschj
;
--AGG003b
SELECT abc.erschj        AS erschj
     , COUNT(abc.buchnr) AS countbuchnr
FROM TBUCH abc
GROUP BY abc.erschj
HAVING abc.erschj IN (1977, 1989, NULL) OR abc.erschj IS NULL
;
erschj                                   countbuchnr
---------------------------------------- -----------
NULL                                     3
1977                                     1
1989                                     3
```

```
--AGG004
```

Natürlich ist es ein Unsinn, den Durchschnitt aller Buchnummern zu berechnen! Beachten Sie aber bitte die Listings!

```
SELECT
          MAX(TBUCH.BUCHNR) AS MAXBU
         ,MIN(TBUCH.BUCHNR) AS MINBU
         ,AVG(TBUCH.BUCHNR) AS AVGBU
         ,SUM(TBUCH.BUCHNR) AS SUMBU
         ,COUNT(buchnr)     AS ANZBU
FROM TBUCH
;
```

Oracle

```
     MAXBU      MINBU      AVGBU      SUMBU      ANZBU
---------- ---------- ---------- ---------- ----------
        27          1 9,63636364        106         11
1 row selected.
```

DB2

```
MAXBU       MINBU       AVGBU       SUMBU       ANZBU
----------- ----------- ----------- ----------- -----------
         27           1           9         106          11

  1 Satz/Sätze ausgewählt.
```

SQL Server

```
MAXBU       MINBU       AVGBU       SUMBU       ANZBU
----------- ----------- ----------- ----------- -----------
27          1           9           106         11

(1 row(s) affected)
```

9.3.2 GROUP BY HAVING

--sel001: Pro Erscheinungsjahr soll der grösste und kleinste Preis, der Durchschnittspreis, die Summe der Preise und die Anzahl der Bücher angelistet werden.

```
SELECT    ERSCHJ
         ,MAX(TBUCH.PREIS) AS MAXpreis
         ,MIN(TBUCH.PREIS) AS MINpreis
         ,AVG(TBUCH.PREIS) AS AVGpreis
         ,SUM(TBUCH.PREIS) AS SUMpreis
         ,COUNT(buchnr)    AS ANZbuecher
FROM TBUCH
GROUP BY ERSCHJ
;
    ERSCHJ   MAXPREIS   MINPREIS   AVGPREIS   SUMPREIS ANZBUECHER
---------- ---------- ---------- ---------- ---------- ----------
      1977         ,5         ,5         ,5         ,5          1
      1988       20,5        3,5         12         24          2
      1989      99,99      99,99      99,99     199,98          3
      1990         55         55         55         55          2
null            99,99      99,99      99,99      99,99          3
```

--sel002: wie sel001, aber nur Erscheinungsjahre mit mindestens 2 Büchern und einem Durchschnittspreis unter 50 Euro.

```
SELECT   Tbuch.erschj     AS erschj
         ,MAX(TBUCH.PREIS) AS MAXpreis
         ,MIN(TBUCH.PREIS) AS MINpreis
         ,AVG(TBUCH.PREIS) AS AVGpreis
         ,SUM(TBUCH.PREIS) AS SUMpreis
         ,COUNT(tbuch.buchnr)    AS ANZbuecher
FROM TBUCH
GROUP BY tbuch.ERSCHJ
HAVING COUNT(TBUCH.BUCHNR) > 1
   AND AVG(TBUCH.PREIS)    < 50.00
;
SELECT
 zwi.erschj
,zwi.MAXpreis
,zwi.MINpreis
,zwi.AVGpreis
,zwi.SUMpreis
,zwi.ANZbuecher
FROM
(
 SELECT   tbuch.ERSCHJ     AS erschj
         ,MAX(TBUCH.PREIS) AS MAXpreis
         ,MIN(TBUCH.PREIS) AS MINpreis
         ,AVG(TBUCH.PREIS) AS AVGpreis
         ,SUM(TBUCH.PREIS) AS SUMpreis
         ,COUNT(tbuch.buchnr)    AS ANZbuecher
FROM TBUCH
GROUP BY ERSCHJ
) zwi
WHERE zwi.ANZbuecher > 1
   AND zwi.avgpreis  < 50.00
;
    ERSCHJ   MAXPREIS   MINPREIS   AVGPREIS   SUMPREIS ANZBUECHER
---------- ---------- ---------- ---------- ---------- ----------
      1988       20,5        3,5         12         24          2
```

--sel003: wie sel002, aber nur Erscheinungsjahre mit mindestens 2 Büchern und einem Durchschnittspreis unter 50 Euro oder Erscheinungsjahre mit genau einem Buch.

```
SELECT   Tbuch.erschj      AS erschj
         ,MAX(TBUCH.PREIS) AS MAXpreis
         ,MIN(TBUCH.PREIS) AS MINpreis
         ,AVG(TBUCH.PREIS) AS AVGpreis
         ,SUM(TBUCH.PREIS) AS SUMpreis
         ,COUNT(tbuch.buchnr)    AS ANZbuecher
FROM TBUCH
GROUP BY tbuch.ERSCHJ
HAVING
(
       COUNT(TBUCH.BUCHNR) > 1
   AND AVG(TBUCH.PREIS)    < 50.00
)  OR  COUNT(tbuch.buchnr) =1
;
SELECT
 zwi.erschj
,zwi.MAXpreis
,zwi.MINpreis
,zwi.AVGpreis
,zwi.SUMpreis
,zwi.ANZbuecher
FROM
(
 SELECT   tbuch.ERSCHJ     AS erschj
         ,MAX(TBUCH.PREIS) AS MAXpreis
         ,MIN(TBUCH.PREIS) AS MINpreis
         ,AVG(TBUCH.PREIS) AS AVGpreis
         ,SUM(TBUCH.PREIS) AS SUMpreis
         ,COUNT(tbuch.buchnr)    AS ANZbuecher
FROM TBUCH
GROUP BY ERSCHJ
) zwi
WHERE
(
       zwi.ANZbuecher > 1
   AND zwi.avgpreis  < 50.00
)
    OR zwi.anzbuecher = 1
;
    ERSCHJ   MAXPREIS   MINPREIS   AVGPREIS   SUMPREIS ANZBUECHER
---------- ---------- ---------- ---------- ---------- ----------
      1977         ,5         ,5         ,5         ,5          1
      1988       20,5        3,5         12         24          2
```

--sel004: Verdichtung von Tvautor pro Buchnr (Anzahl der Autoren, Durchschnittspraemie, maximale Praemie)

```
select
  tvautor.buchnr           as buchnr
 ,COUNT(tvautor.autornr)   as anzautoren
 ,AVG(tvautor.praemie)     as avgprprobuch
 ,MAX(tvautor.praemie)     as maxprprobuch
FROM tvautor
GROUP BY tvautor.buchnr
;
buchnr      anzautoren  avgprprobuch                    maxprprobuch
----------- ----------- ------------------------------- ------------
1           2           NULL                                  NULL
2           1           NULL                                  NULL
5           1           NULL                                  NULL
6           1           NULL                                  NULL
7           1           NULL                                  NULL
8           1           20.000000                             20.000
9           2           254.000000                           498.000
12          2           30.000000                             30.000
(8 row(s) affected)
```

--sel005: Wie sel004 aber nur Bücher mit mehr als einem Autor.

```
select
  tvautor.buchnr           as buchnr
 ,COUNT(tvautor.autornr)   as anzautoren
 ,AVG(tvautor.praemie)     as avgprprobuch
 ,MAX(tvautor.praemie)     as maxprprobuch
FROM tvautor
GROUP BY tvautor.buchnr
HAVING COUNT(tvautor.autornr) > 1
;
SELECT
*
FROM
(
select
  tvautor.buchnr           as buchnr
 ,COUNT(tvautor.autornr)   as anzautoren
 ,AVG(tvautor.praemie)     as avgprprobuch
 ,MAX(tvautor.praemie)     as maxprprobuch
FROM tvautor
GROUP BY tvautor.buchnr
)zwi
WHERE zwi.anzautoren > 1
;

buchnr      anzautoren  avgprprobuch                  maxprprobuch
----------- ----------- ------------------------------ ------------
1           2           NULL                              NULL
9           2           254.000000                        498.000
12          2           30.000000                         30.000

(3 row(s) affected)
```

--sel006: Wie sel005 aber nur Bücher mit wenigstens einer wohldefinierten Praemie.

```
select
  tvautor.buchnr            as buchnr
 ,COUNT(tvautor.autornr)    as anzautoren
 ,AVG(tvautor.praemie)      as avgprprobuch
 ,MAX(tvautor.praemie)      as maxprprobuch
FROM tvautor
GROUP BY tvautor.buchnr
HAVING COUNT(tvautor.autornr) > 1
   AND COUNT(tvautor.praemie) > 0
;
SELECT
 zwi.buchnr
,zwi.anzautoren
,zwi.avgprprobuch
,zwi.maxprprobuch
FROM
(
select
  tvautor.buchnr            as buchnr
 ,COUNT(tvautor.autornr)    as anzautoren
 ,AVG(tvautor.praemie)      as avgprprobuch
 ,MAX(tvautor.praemie)      as maxprprobuch
 ,COUNT(tvautor.praemie)    as countpraemie
FROM tvautor
GROUP BY tvautor.buchnr
)zwi
WHERE zwi.anzautoren > 1
  AND zwi.countpraemie > 0
;

buchnr      anzautoren  avgprprobuch                maxprprobuch
----------- ----------- ---------------------------------------
9           2           254.000000                     498.000
12          2           30.000000                       30.000

(2 row(s) affected)
```

9.3.3 GROUP BY „Redundanzen!"

9.4 Inner Natural-Join, innerer natürlicher Verbund, INNER JOIN

Formulieren Sie die Inner Natural-Joins mit Hilfe von INNER JOIN!

9.4.1 Join1

```
SELECT
  Tbuch.Buchnr AS Buchnr
, Tbuch.Preis  AS Preis
, Tbuch.Erschj AS Erschj
, Tisbn.Lfdnr  AS Lfdnr
, Tisbn.ISBN
FROM   Tbuch, Tisbn
WHERE  Tbuch.Buchnr = Tisbn.Buchnr
;
SELECT
  Tbuch.Buchnr AS Buchnr
, Tbuch.Preis  AS Preis
, Tbuch.Erschj AS Erschj
, Tisbn.Lfdnr  AS Lfdnr
, Tisbn.ISBN
FROM   Tbuch INNER JOIN Tisbn
       ON Tbuch.Buchnr = Tisbn.Buchnr
;
BUCHNR      PREIS     ERSCHJ LFDNR ISBN
----------- --------- ------ ----- ----------
          8      0,50  1977,     1, 3472864303
          8      0,50  1977,     2, 34728643yx
         12         -  1989,     1, 0201501139
```

Beide SELECT-Anweisungen generieren die gleiche Liste.

9.4.2 Join2

```
SELECT
 Tisbn.Buchnr     AS Buchnr
,Tisbn.Lfdnr      AS Lfdnr
,Tisbn.ISBN       AS Isbn
,Tvautor.Autornr  AS Autornr
FROM  Tisbn, Tvautor
WHERE Tisbn.Buchnr = Tvautor.Buchnr
;
SELECT
 Tisbn.Buchnr     AS Buchnr
,Tisbn.Lfdnr      AS Lfdnr
,Tisbn.ISBN       AS Isbn
,Tvautor.Autornr  AS Autornr
FROM  Tisbn INNER JOIN Tvautor
      ON Tisbn.Buchnr = Tvautor.Buchnr
;
BUCHNR      LFDNR ISBN       AUTORNR
----------- ----- ---------- -----------
          8     1, 3472864303           2
          8     2, 34728643yx           2
         12     1, 0201501139          20
         12     1, 0201501139          21
```

9.4.3 Join3

```
SELECT
 Tbuch.Buchnr      AS Buchnr
,Tbuch.Preis       AS Preis
,Tbuch.Erschj      AS Erschj
,Tisbn.Lfdnr       AS Lfdnr
,Tisbn.ISBN
,Tvautor.Autornr   AS Autornr
FROM Tisbn, Tbuch, Tvautor
WHERE  Tisbn.Buchnr = Tbuch.Buchnr
AND    Tbuch.Buchnr = Tvautor.Buchnr
;
SELECT
 Tbuch.Buchnr      AS Buchnr
,Tbuch.Preis       AS Preis
,Tbuch.Erschj      AS Erschj
,Tisbn.Lfdnr       AS Lfdnr
,Tisbn.ISBN
,Tvautor.Autornr   AS Autornr
FROM Tisbn INNER JOIN Tbuch
     ON Tisbn.Buchnr = Tbuch.Buchnr
           INNER JOIN Tvautor
     ON Tbuch.Buchnr = Tvautor.Buchnr
;
BUCHNR      PREIS     ERSCHJ LFDNR ISBN       AUTORNR
----------- --------- ------ ----- ---------- -----------
          8      0,50  1977,     1, 3472864303           2
          8      0,50  1977,     2, 34728643yx           2
         12         -  1989,     1, 0201501139          20
         12         -  1989,     1, 0201501139          21
```

9.4.4 Join4

```
SELECT
 Tbuch.Buchnr      AS Buchnr
,Tbuch.Preis       AS Preis
,Tbuch.Erschj      AS Erschj
,Tvautor.Autornr   AS Autornr
,Tautor.Autor      AS Autor
FROM Tbuch, Tvautor, Tautor
WHERE  Tbuch.Buchnr = Tvautor.Buchnr
AND    Tvautor.Autornr = Tautor.Autornr
ORDER BY Buchnr, Autornr
;
SELECT
 Tbuch.Buchnr      AS Buchnr
,Tbuch.Preis       AS Preis
,Tbuch.Erschj      AS Erschj
,Tvautor.Autornr   AS Autornr
,Tautor.Autor      AS Autor
FROM Tbuch INNER JOIN Tvautor
     ON Tbuch.Buchnr = Tvautor.Buchnr
           INNER JOIN Tautor
     ON Tvautor.Autornr = Tautor.Autornr
ORDER BY Buchnr, Autornr
;
BUCHNR      PREIS     ERSCHJ AUTORNR     AUTOR
----------- --------- ------ ----------- ----------------------
          1         -      -         100 BUSCH
          1         -      -         200 BUSCH
          2         -      -         200 BUSCH
          5      3,50  1988,           1 Boell
          6     20,50  1988,           2 Grass
          7     99,99  1989,           3 Eco
          8      0,50  1977,           2 Grass
          9     55,00  1990,          10 Emil Hack
          9     55,00  1990,          11 Frieda Holz
         12         -  1989,          20 C. J. Date
         12         -  1989,          21 Colin J. White
```

9.4.5 Join5

```
SELECT
 Tverlag.verlagnr    as verlagnr
,Tverlag.verlag      as verlag
,Tbuch.buchnr        as buchnr
FROM Tverlag, Tbuch
WHERE  Tverlag.verlagnr = Tbuch.verlagnr
ORDER BY verlagnr, buchnr
;
SELECT
 Tverlag.verlagnr    as verlagnr
,Tverlag.verlag      as verlag
,Tbuch.buchnr        as buchnr
FROM Tverlag INNER JOIN Tbuch
     ON  Tverlag.verlagnr = Tbuch.verlagnr
ORDER BY verlagnr, buchnr
;
VERLAGNR    VERLAG               BUCHNR
----------- -------------------- -----------
       1111 Forkel                       8
       1111 Forkel                       9
```

9.4.6 Join6

```
SELECT
 Tverlag.verlagnr    as verlagnr
,Tverlag.verlag      as verlag
,Tbuch.buchnr        as buchnr
,Tisbn.isbn          as isbn
FROM Tverlag, Tbuch, Tisbn
WHERE  Tverlag.verlagnr = Tbuch.verlagnr
AND    Tbuch.buchnr = tisbn.buchnr
ORDER BY verlagnr, buchnr, isbn
;
SELECT
 Tverlag.verlagnr    as verlagnr
,Tverlag.verlag      as verlag
,Tbuch.buchnr        as buchnr
,tisbn.isbn
FROM Tverlag INNER JOIN Tbuch
     ON  Tverlag.verlagnr = Tbuch.verlagnr
              INNER JOIN tisbn
     ON tbuch.buchnr = tisbn.buchnr
ORDER BY verlagnr, buchnr, isbn
;
verlagnr    verlag               buchnr      isbn
----------- -------------------- ----------- ----------
1111        Forkel               8           3472864303
1111        Forkel               8           34728643yx
```

9.5 Join, Restriktion, Projektion ohne DISTINCT

9.5.1 JOI001: Buchnr, ISBN's von der Butt

```
BUCHNR      ISBN
----------- ----------
          8 3472864303
          8 34728643yx
SELECT
  TISBN.BUCHNR, TISBN.ISBN
FROM TISBN, TBUCH
WHERE  TISBN.BUCHNR = TBUCH.BUCHNR
AND    TBUCH.TITEL  = 'der Butt'
;
SELECT
  TISBN.BUCHNR, TISBN.ISBN
FROM TISBN JOIN TBUCH
     ON TISBN.BUCHNR = TBUCH.BUCHNR
WHERE  TBUCH.TITEL = 'der Butt'
;
```

9.5.2 JOI002: Buchnr, Autornr, Autor von Grass

```
BUCHNR      AUTORNR     AUTOR
----------- ----------- --------------
          6           2 Grass
          8           2 Grass

SELECT
  TVAUTOR.BUCHNR, TAUTOR.AUTORNR, TAUTOR.AUTOR
FROM   TVAUTOR, TAUTOR
WHERE  TVAUTOR.AUTORNR = TAUTOR.AUTORNR
AND    TAUTOR.AUTOR    = 'Grass'
;
SELECT
  TVAUTOR.BUCHNR, TAUTOR.AUTORNR, TAUTOR.AUTOR
FROM   TVAUTOR INNER JOIN TAUTOR
       ON TVAUTOR.AUTORNR = TAUTOR.AUTORNR
WHERE  TAUTOR.AUTOR   = 'Grass'
;
```

9.5.3 JOI003: Autornr, Autor, Titel, Buchnr von a guide to db2

```
AUTORNR     AUTOR                TITEL                BUCHNR
----------- -------------------- -------------------- -----------
         20 C. J. Date           a guide to db2               12
         21 Colin J. White       a guide to db2               12
```

```
SELECT
       Tautor.Autornr
     , SUBSTRING(TAUTOR.AUTOR,   1, 20) AS Autor
     , SUBSTRING(TBUCH.TITEL,    1, 20) AS Titel
     ----SUBSTR DB2 und Oracle
     , Tbuch.Buchnr
FROM   TAUTOR, TVAUTOR, TBUCH
WHERE  TAUTOR.AUTORNR = TVAUTOR.AUTORNR
AND    TVAUTOR.BUCHNR = TBUCH.BUCHNR
AND    TBUCH.TITEL    = 'a guide to db2'
;
SELECT
       Tautor.Autornr
     , SUBSTRING(TAUTOR.AUTOR,   1, 20) AS Autor
     , SUBSTRING(TBUCH.TITEL,    1, 20) AS Titel
     ----SUBSTR DB2 und Oracle
     , Tbuch.Buchnr
FROM ((  TAUTOR INNER JOIN TVAUTOR
         ON TAUTOR.AUTORNR = TVAUTOR.AUTORNR )
         INNER JOIN TBUCH ON TVAUTOR.BUCHNR = TBUCH.BUCHNR)
WHERE TBUCH.TITEL    = 'a guide to db2'
;
```

9.5.4 JOI004: Buchnr, ISBN, Preis, Titel

```
BUCHNR      ISBN       PREIS     TITEL
----------- ---------- --------- --------------------
          8 3472864303      0,50 der Butt
          8 34728643yx      0,50 der Butt
         12 0201501139         - a guide to db2
```

```
SELECT
       TBUCH.BUCHNR
     , TISBN.ISBN
     , TBUCH.PREIS
     , TBUCH.TITEL
FROM   TBUCH, TISBN
WHERE TBUCH.BUCHNR = TISBN.BUCHNR
;
SELECT
       TBUCH.BUCHNR
     , TISBN.ISBN
     , TBUCH.PREIS
     , TBUCH.TITEL
FROM   TBUCH INNER JOIN TISBN
       ON TBUCH.BUCHNR = TISBN.BUCHNR
;
```

9.5.5 JOI005: Buchnr, ISBN, Preis, Titel von der Butt

```
BUCHNR      ISBN       PREIS     TITEL
----------- ---------- --------- ----------
          8 3472864303      0,50 der Butt
          8 34728643yx      0,50 der Butt
```

```
SELECT
       TBUCH.BUCHNR
     , TISBN.ISBN
     , TBUCH.PREIS
     , TBUCH.TITEL
FROM   TBUCH, TISBN
WHERE  TBUCH.BUCHNR = TISBN.BUCHNR
AND    TBUCH.TITEL = 'der Butt'
;
SELECT
       TBUCH.BUCHNR
     , TISBN.ISBN
     , TBUCH.PREIS
     , TBUCH.TITEL
FROM   TBUCH INNER JOIN TISBN
       ON TBUCH.BUCHNR = TISBN.BUCHNR
WHERE  TBUCH.TITEL = 'der Butt'
;
```

9.5.6 JOI006: Buchnr, Preis, Titel von Büchern die nur einen Autor haben (Hinweis: Tvautor.Lfdnr = 0)

```
BUCHNR      PREIS     TITEL
----------- --------- ------------------------------
          2         - C
          5      3,50 Ansichten eines Clowns
          6     20,50 die Blechtrommel
          7     99,99 der Name der Rose
          8      0,50 der Butt
```

```
SELECT
  TBUCH.BUCHNR, TBUCH.PREIS, TBUCH.TITEL
FROM   TBUCH, TVAUTOR
WHERE  TBUCH.BUCHNR = TVAUTOR.BUCHNR
AND    TVAUTOR.LFDNR  = 0
;
SELECT
  TBUCH.BUCHNR, TBUCH.PREIS, TBUCH.TITEL
FROM   TBUCH INNER JOIN TVAUTOR
       ON TBUCH.BUCHNR = TVAUTOR.BUCHNR
WHERE  TVAUTOR.LFDNR  = 0
;
SELECT
  TBUCH.BUCHNR, TBUCH.PREIS, TBUCH.TITEL
FROM   TBUCH
WHERE  TBUCH.BUCHNR IN
(
 select tvautor.buchnr
 from tvautor
 where Tvautor.lfdnr = 0
)
;
```

ACHTUNG: Wenn die Anwendungsprogramme bzw. das Datenbankmanagementsystem nicht garantieren, dass die Llfdnr=0 ist bei genau einem Autor, dann kommt es bei Lfdnr=0 zu inkonsistenten Listen.

Lösung ohne Tvautor.Lfdnr:

```
SELECT
  Tbuch.BUCHNR, Tbuch.PREIS, Tbuch.TITEL
FROM Tbuch
WHERE Tbuch.Buchnr IN
      (
       SELECT Tvautor.Buchnr
       FROM Tvautor
       GROUP BY Tvautor.Buchnr
       HAVING COUNT(tvautor.autornr) = 1
      )
;
```

9.5.7 JOI007: Buchnr, Preis, Titel, Autornr, Autor von Büchern die nur einen Autor haben (Hinweis: Tvautor.Lfdnr = 0)

```
BUCHNR      PREIS     TITEL      AUTORNR     AUTOR
----------- --------- ---------- ----------- ------------------
          2         - C                  200 BUSCH
          5      3,50 Ansichten            1 Boell
          6     20,50 die Blecht           2 Grass
          7     99,99 der Name d           3 Eco
          8      0,50 der Butt             2 Grass
```

```
SELECT
      Tbuch.BUCHNR
    , Tbuch.PREIS
    , SUBSTRING(Tbuch.TITEL, 1, 10) AS Titel
     ----SUBSTR DB2 und Oracle
    , TAUTOR.AUTORNR
    , Tautor.AUTOR
FROM   TBUCH, TVAUTOR, TAUTOR
WHERE  TBUCH.BUCHNR       = TVAUTOR.BUCHNR
AND    TVAUTOR.AUTORNR    = TAUTOR.AUTORNR
AND    TVAUTOR.LFDNR      = 0
;
SELECT
      Tbuch.BUCHNR
    , Tbuch.PREIS
    , SUBSTRING(Tbuch.TITEL, 1, 10) AS Titel
     ----SUBSTR DB2 und Oracle
    , TAUTOR.AUTORNR
    , tautor.AUTOR
FROM   TBUCH INNER JOIN TVAUTOR
       ON TBUCH.BUCHNR    = TVAUTOR.BUCHNR
       INNER JOIN TAUTOR
       ON TVAUTOR.AUTORNR = TAUTOR.AUTORNR
WHERE  TVAUTOR.LFDNR      = 0
;
```

9.5.8 JOI008: Buchnr, Preis, Titel, Autornr von Autoren mit dem Namen Boell

```
BUCHNR      PREIS     TITEL      AUTORNR
----------- --------- ---------- -----------
          5      3,50 Ansichten            1
```

```
SELECT
      Tbuch.BUCHNR
    , Tbuch.PREIS
    , SUBSTRING(Tbuch.TITEL, 1, 10) AS Titel
     ----SUBSTR DB2 und Oracle
    , TVAUTOR.AUTORNR
FROM   TBUCH, TVAUTOR, TAUTOR
WHERE  TBUCH.BUCHNR       = TVAUTOR.BUCHNR
AND    TVAUTOR.AUTORNR    = TAUTOR.AUTORNR
AND    TAUTOR.AUTOR       = 'Boell'
;
SELECT
      Tbuch.BUCHNR
    , Tbuch.PREIS
    , SUBSTRING(Tbuch.TITEL, 1, 10) AS Titel
     ----SUBSTR DB2 und Oracle
    , TVAUTOR.AUTORNR
FROM   TBUCH INNER JOIN TVAUTOR
       ON TBUCH.BUCHNR       = TVAUTOR.BUCHNR
       INNER JOIN TAUTOR
       ON TVAUTOR.AUTORNR    = TAUTOR.AUTORNR
WHERE  TAUTOR.AUTOR          = 'Boell'
;
```

9.5.9 JOI009: Buchnr, Preis, Autornr von Büchern mit einem 1. Autor des Namens Boell

```
BUCHNR      PREIS     AUTORNR
----------- --------- -----------
  0 Satz/Sätze ausgewählt.
```

```
SELECT
      Tbuch.BUCHNR
    , Tbuch.PREIS
    , TVAUTOR.AUTORNR
FROM   TBUCH, TVAUTOR, TAUTOR
WHERE  TBUCH.BUCHNR       = TVAUTOR.BUCHNR
AND    TVAUTOR.AUTORNR    = TAUTOR.AUTORNR
AND    TAUTOR.AUTOR       = 'Boell'
AND    TVAUTOR.Lfdnr      = 1
;
SELECT
      Tbuch.BUCHNR
    , Tbuch.PREIS
    , TVAUTOR.AUTORNR
FROM   TBUCH INNER JOIN TVAUTOR
       ON TBUCH.BUCHNR       = TVAUTOR.BUCHNR
       INNER JOIN TAUTOR
       ON TVAUTOR.AUTORNR    = TAUTOR.AUTORNR
WHERE  TAUTOR.AUTOR          = 'Boell'
AND    TVAUTOR.Lfdnr         = 1
;
```

9.5.10 JOI010: Buchnr, Preis, Autornr von Büchern bei denen wenigstens einer der Autoren den Namen Grass hat

```
BUCHNR      PREIS     AUTORNR
----------- --------- -----------
          6     20,50           2
          8      0,50           2
```

```
SELECT
  TBUCH.BUCHNR, PREIS, TAUTOR.AUTORNR
FROM   TBUCH, TVAUTOR, TAUTOR
WHERE  TBUCH.BUCHNR       = TVAUTOR.BUCHNR
AND    TVAUTOR.AUTORNR    = TAUTOR.AUTORNR
AND    TAUTOR.AUTOR       = 'Grass'
;
```

9.5.11 JOI011: Buchnr, Preis, Autornr aller Bücher mit Autor Grass und Titel Blechtrommel

```
BUCHNR      PREIS     AUTORNR
----------- --------- -----------
  0 Satz/Sätze ausgewählt.
```

```
SELECT TBUCH.BUCHNR, PREIS, TAUTOR.AUTORNR
FROM   TBUCH, TVAUTOR, TAUTOR
WHERE  TBUCH.BUCHNR       = TVAUTOR.BUCHNR
AND    TVAUTOR.AUTORNR    = TAUTOR.AUTORNR
AND    TAUTOR.AUTOR       = 'Grass'
AND    TBUCH.TITEL        = 'Blechtrommel'
;
```

9.5.12 JOI012: Buchnr, Preis, Autornr aller Bücher mit Autor Grass und Titel die Blechtrommel

```
SELECT TBUCH.BUCHNR, PREIS, TAUTOR.AUTORNR
FROM   TBUCH, TVAUTOR, TAUTOR
WHERE  TBUCH.BUCHNR       = TVAUTOR.BUCHNR
AND    TVAUTOR.AUTORNR    = TAUTOR.AUTORNR
AND    TAUTOR.AUTOR       = 'Grass'
AND    TBUCH.TITEL        = 'die Blechtrommel      '
;
```

DB2

```
BUCHNR      PREIS     AUTORNR
----------- --------- -----------
          6     20,50           2
```

ORACLE die Spalte Tbuch.Titel ist VARCHAR2

no rows selected

SQL Server

```
BUCHNR      PREIS     AUTORNR
----------- --------- -----------
6           20.50     2
```

9.6 left outer join

9.6.1 leftjoin1

Buchnr, Erschj aller Bücher,

Isbn und Lfdnr der Isbn (sofern vorhanden)

sortiert nach Buchnr, Lfdnr

```
SELECT
 Tbuch.Buchnr   AS Buchnr
,Tbuch.Erschj   AS Erschj
,Tisbn.Isbn     AS Isbn
,Tisbn.Lfdnr    AS Lfdnr
FROM Tbuch LEFT OUTER JOIN Tisbn
ON Tbuch.Buchnr = Tisbn.Buchnr
ORDER BY
 Buchnr
,Tisbn.Lfdnr
;
```

```
BUCHNR      ERSCHJ ISBN        LFDNR
----------- ------ ---------- -----
          1      - -              -
          2      - -              -
          5   1988, -              -
          6   1988, -              -
          7   1989, -              -
          8   1977, 3472864303    1,
          8   1977, 34728643yx    2,
          9   1990, -              -
         11   1990, -              -
         12   1989, 0201501139    1,
         18   1989, -              -
         27      - -              -

  12 Satz/Sätze ausgewählt.
```

9.6.2 leftjoin2

Buchnr, Erschj aller Bücher,

Isbn und Lfdnr der Isbn (sofern vorhanden),

Autornr und Lfdnr des Autors sofern vorhanden

sortiert nach Buchnr, Lfdnr der Isbn, Autornr, Lfdnr des Autors

```
SELECT
 Tbuch.Buchnr    AS Buchnr
,Tbuch.Erschj    AS Erschj
,Tisbn.Isbn      AS Isbn
,Tisbn.Lfdnr     AS Ilfdnr
,Tvautor.Autornr AS Autornr
,Tvautor.Lfdnr   AS Alfdnr
FROM (
     Tbuch LEFT OUTER JOIN Tisbn
     ON Tbuch.Buchnr = Tisbn.Buchnr
     )     LEFT OUTER JOIN Tvautor
     ON Tbuch.Buchnr = Tvautor.Buchnr
ORDER BY
 Buchnr
,Tisbn.Lfdnr
,Tvautor.Autornr
,Tvautor.Lfdnr
;
```

```
BUCHNR      ERSCHJ ISBN        ILFDNR AUTORNR     ALFDNR
----------- ------ ---------- ------ ----------- ------
          1      - -               -         100     1,
          1      - -               -         200     2,
          2      - -               -         200     0,
          5  1988, -               -           1     0,
          6  1988, -               -           2     0,
          7  1989, -               -           3     0,
          8  1977, 3472864303     1,          2     0,
          8  1977, 34728643yx     2,          2     0,
          9  1990, -               -          10     1,
          9  1990, -               -          11     2,
         11  1990, -               -           -      -
         12  1989, 0201501139     1,         20     1,
         12  1989, 0201501139     1,         21     2,
         18  1989, -               -           -      -
         27      - -               -           -      -

  15 Satz/Sätze ausgewählt.
```

9.6.3 Leftjoin3

Erschj, Buchnr aller Bücher,

Autor, Autornr und Lfdnr des Autors sofern vorhanden

sortiert nach Erschj, Buchnr, Autor, Autornr, Lfdnr des Autors

```
SELECT
 Tbuch.Erschj    AS Erschj
,Tbuch.Buchnr    AS Buchnr
,SUBSTRING(Tautor.Autor, 1, 10) AS Autor
----SUBSTR DB2 und Oracle
,Tvautor.Autornr AS Autornr
,Tvautor.Lfdnr   AS Alfdnr
FROM (
     Tbuch LEFT OUTER JOIN Tvautor
     ON Tbuch.Buchnr = Tvautor.Buchnr
     )     LEFT OUTER JOIN Tautor
     ON Tvautor.Autornr = Tautor.Autornr
ORDER BY
 Tbuch.Erschj
,Tbuch.Buchnr
,Tautor.Autor
,Tautor.Autornr
,Tvautor.Lfdnr
;
```

```
ERSCHJ BUCHNR      AUTOR      AUTORNR     ALFDNR
------ ----------- ---------- ----------- ------
 1977,           8 Grass                2     0,
 1988,           5 Boell                1     0,
 1988,           6 Grass                2     0,
 1989,           7 Eco                  3     0,
 1989,          12 C. J. Date          20     1,
 1989,          12 Colin J. W          21     2,
 1989,          18 -                    -      -
 1990,           9 Emil Hack           10     1,
 1990,           9 Frieda Hol          11     2,
 1990,          11 -                    -      -
     -           1 BUSCH               100     1,
     -           1 BUSCH               200     2,
     -           2 BUSCH               200     0,
     -          27 -                    -      -

  14 Satz/Sätze ausgewählt.
```

9.6.4 Leftjoin4

Autor, Autornr aller Autoren,

Erschj, Buchnr und Lfdnr des Autors sofern vorhanden

sortiert nach Autor, Autornr, Buchnr, Lfdnr des Autors

```
SELECT
 SUBSTRING(Tautor.Autor, 1, 10) AS Autor
----SUBSTR DB2 und Oracle
,Tautor.Autornr AS Autornr
,Tbuch.Erschj    AS Erschj
,Tbuch.Buchnr  AS Buchnr
,Tvautor.Lfdnr   AS Alfdnr
FROM (
     Tautor LEFT OUTER JOIN Tvautor
     ON Tautor.Autornr = Tvautor.Autornr
     )      LEFT OUTER JOIN Tbuch
     ON Tvautor.Buchnr = Tbuch.Buchnr
ORDER BY
 Tautor.Autor
,Tautor.Autornr
,Tvautor.Buchnr
,Tvautor.Lfdnr
;
```

AUTOR	AUTORNR	ERSCHJ	BUCHNR	ALFDNR
Boell	1	1988,	5	0,
BUSCH	100	-	1	1,
BUSCH	200	-	1	2,
BUSCH	200	-	2	0,
C. J. Date	20	1989,	12	1,
Colin J. W	21	1989,	12	2,
Eco	3	1989,	7	0,
Emil Hack	10	1990,	9	1,
Frieda Hol	11	1990,	9	2,
Grass	2	1988,	6	0,
Grass	2	1977,	8	0,
Scheifele	**6**	**-**	**-**	**-**

12 Satz/Sätze ausgewählt.

9.7 GROUP BY und Join

9.7.1 GROUP BY und Join, Primärschlüssel und Fremdschlüssel

TABLE Tbuch

buchnr	titel
1	C
2	C
5	Ansichten eines Clowns
6	die Blechtrommel
7	der Name der Rose
8	der Butt
9	DB2 fuer Sie
11	Elvis in Heidelberg
12	a guide to db2
18	Database Systems
27	die Jüdin von Toledo

TABLE Tvautor

buchnr	autornr	lfdnr	praemie
1	100	1	NULL
1	200	2	NULL
2	200	0	NULL
5	1	0	NULL
6	2	0	NULL
7	3	0	NULL
8	2	0	20.000
9	10	1	10.000
9	11	2	498.000
12	20	1	30.000
12	21	2	NULL

Formulieren Sie die SELECT-Anweisung, die zu Buchnr und Titel die Summe der Praemien, den Durchschnitt der Praemien und die Anzahl der Autoren anlistet.

buchnr	titel	sumbu	avgbu	anzau
1	C	NULL	NULL	2
2	C	NULL	NULL	1
5	Ansic	NULL	NULL	1
6	die B	NULL	NULL	1
7	der N	NULL	NULL	1
8	der B	20.000	20.000000	1
9	DB2 f	508.000	254.000000	2
12	a gui	30.000	30.000000	2

--Lösung A: (tbuch join tvautor) group by

```
select
  tbuch.buchnr                          as buchnr
, SUBSTRING(tbuch.titel, 1,5)           as titel
----SUBSTR DB2 und Oracle
, sum(tvautor.praemie)                  as sumbu
, avg(tvautor.praemie)                  as avgbu
, count(tvautor.autornr)                as anzau
from tbuch, tvautor
where tbuch.buchnr= tvautor.buchnr
group by tbuch.buchnr, tbuch.titel
order by buchnr
;
```

--Lösung B: tbuch join (tvautor group by) xxx

```
select
  tbuch.buchnr                 as buchnr
, SUBSTRING(tbuch.titel, 1,5) as titel
----SUBSTR DB2 und Oracle
, xxx.sumbu
, xxx.avgbu
, xxx.anzau
from tbuch inner join
(
select
  tvautor.buchnr                                as buchnr
, sum(tvautor.praemie)                          as sumbu
, avg(tvautor.praemie)                          as avgbu
, count(tvautor.autornr)                        as anzau
from tvautor group by tvautor.buchnr
) xxx
on tbuch.buchnr= xxx.buchnr
order by buchnr
;
```

--Lösung C: WITH xxx AS

```
WITH
xxx AS
 (
select
  tvautor.buchnr                                as buchnr
, sum(tvautor.praemie)                          as sumbu
, avg(tvautor.praemie)                          as avgbu
, count(tvautor.autornr)                        as anzau
from tvautor group by tvautor.buchnr
)
 select
  tbuch.buchnr                 as buchnr
, SUBSTRING(tbuch.titel, 1,5) as titel
----SUBSTR DB2 und Oracle
, xxx.sumbu
, xxx.avgbu
, xxx.anzau
from tbuch inner join xxx
on tbuch.buchnr= xxx.buchnr
order by buchnr
;
```

9.7.2 GROUP BY und Join, „eine Falle“

--Liste1: Buchnr, Titel und Anzahl der Isbn’s

```
select
 tbuch.buchnr                       as buchnr
,SUBSTRING(tbuch.titel, 1, 5)       as titel
----SUBSTR DB2 und Oracle
,count(tisbn.buchnr )               as anzisbn
from tbuch inner join tisbn
     on tbuch.buchnr = tisbn.buchnr
group by tbuch.buchnr, tbuch.titel
;
buchnr      titel anzisbn
----------- ----- -----------
8           der B 2
12          a gui 1
```

--Liste2: Buchnr, Titel, Anzahl der Autoren, Summe und Durchschnitt der Praemien pro Buch

```
select
 tbuch.buchnr                       as buchnr
,SUBSTRING(tbuch.titel, 1, 5)       as titel
----SUBSTR DB2 und Oracle
,count(tvautor.autornr)             as anzautor
,sum(tvautor.praemie)               as sumpraemie
,avg(tvautor.praemie)               as avgpraemie
from tbuch inner join tvautor
     on tbuch.buchnr = tvautor.buchnr
group by tbuch.buchnr, tbuch.titel
;
buchnr      titel anzautor    sumpraemie                    avgpraemie
----------- ----- ----------- ----------------------------- -------------
1           C     2           NULL                          NULL
2           C     1           NULL                          NULL
5           Ansic 1           NULL                          NULL
6           die B 1           NULL                          NULL
7           der N 1           NULL                          NULL
8           der B 1           20.000                        20.000000
9           DB2 f 2           508.000                       254.000000
12          a gui 2           30.000                        30.000000
```

--Liste3 und Anweisung3: Warum ist die folgende Select-Anweisung und die Liste nicht in Ordnung?

```
select
 tbuch.buchnr                       as buchnr
,SUBSTRING(tbuch.titel, 1, 5)       as titel
----SUBSTR DB2 und Oracle
,count(tvautor.autornr)             as anzautor
,sum(tvautor.praemie)               as sumpraemie
,avg(tvautor.praemie)               as avgpraemie
,count(tisbn.buchnr )               as anzisbn
from tbuch inner join tisbn
     on tbuch.buchnr = tisbn.buchnr
           inner join tvautor
     on tbuch.buchnr = tvautor.buchnr
group by tbuch.buchnr, tbuch.titel
;
buchnr      titel anzautor    sumpraemie     avgpraemie   anzisbn
----------- ----- ----------- -------------- ------------ ------
8           der B 2           40.000         20.000000    2
12          a gui 2           30.000         30.000000    2
```

Antwort zu Liste3 und Anweisung3:

Das Buchnr 8 hat zwei ISBN's aber einen Autor!

Das Buchnr 12 hat eine ISBN aber zwei Autoren!

Die Summe der Praemien von Buchnr 8 ist nicht korrekt!

Schreiben sie eine Select-Anweisung, die die korrekte Liste liefert.

```
buchnr      titel anzautor    sumpraemie     avgpraemie   anzisbn
----------- ----- ----------- -------------- ------------ ------
8           der B 1           20.000         20.000000    2
12          a gui 2           30.000         30.000000    1
```

```
select
 zwi.buchnr      as buchnr
,zwi.titel       as titel
,zwi.anzautor    as anzautor
,zwi.sumpraemie  as sumpraemie
,zwi.avgpraemie  as avgpraemie
,zwa.anzisbn     as anzisbnb
FROM
(
 select
 tbuch.buchnr                    as buchnr
,SUBSTRING(tbuch.titel, 1, 5)    as titel
----SUBSTR DB2 und Oracle
,count(tvautor.autornr)          as anzautor
,sum(tvautor.praemie)            as sumpraemie
,avg(tvautor.praemie)            as avgpraemie
from tbuch inner join tvautor
     on tbuch.buchnr = tvautor.buchnr
group by tbuch.buchnr, tbuch.titel
)zwi
 INNER JOIN
(
 select
 tisbn.buchnr         as buchnr
,count(tisbn.buchnr ) as anzisbn
from tisbn
group by tisbn.buchnr
) zwa
on zwi.buchnr = zwa.buchnr
```

9.7.3 GROUP BY "umfassend und verständlich?"

9.8 Subquery mit IN, Subquery mit EXISTS, Variante mit Join

9.8.1 Verlagnr, Verlag von Verlagen mit wenigstens einem Buch

```
verlagnr    verlag
----------- --------------------
1111        Forkel
```

```
-- Variante1 Subquery mit IN
SELECT
  tverlag.verlagnr, tverlag.verlag
FROM   tverlag
WHERE  tverlag.verlagnr IN
                (
                  SELECT tbuch.verlagnr
                  FROM   tbuch
                 )
;
-- Variante2 Subquery mit EXISTS
SELECT
  tverlag.verlagnr, tverlag.verlag
FROM tverlag
WHERE EXISTS (
               SELECT * FROM tbuch
               WHERE tbuch.verlagnr = tverlag.verlagnr
             )
;
-- Variante3 Join und Projektion
SELECT DISTINCT
  tverlag.verlagnr, tverlag.verlag
FROM tverlag INNER JOIN tbuch
     ON tverlag.verlagnr = tbuch.verlagnr
;
-- Variante4
WITH zwi AS
(
  SELECT          tverlag.verlagnr FROM tverlag
  INTERSECT
  SELECT DISTINCT tbuch.verlagnr FROM tbuch
)
SELECT
  tverlag.verlagnr, tverlag.verlag
FROM tverlag INNER JOIN zwi
     ON tverlag.verlagnr = zwi.verlagnr
;
```

9.8.2 Verlagnr, Verlag von Verlagen ohne Buch

INSERT INTO Tverlag(verlagnr, verlag)
VALUES(2222, 'RORORO')
;
----Wir fügen einen Verlag ohne Bücher ein.
----Achtung: tbuch.verlagnr kann eine NULL präsentieren

```
verlagnr    verlag
----------- --------------------
2222        rororo
```

```
-- Variante1 Subquery mit NOT IN
SELECT
  tverlag.verlagnr, tverlag.verlag
FROM   tverlag
WHERE  tverlag.verlagnr NOT IN
                (
                 SELECT tbuch.verlagnr
                 FROM   tbuch
                 WHERE tbuch.verlagnr IS NOT NULL
                )
;
-- Variante2 Subquery mit NOT EXISTS
SELECT
  tverlag.verlagnr, tverlag.verlag
FROM tverlag
WHERE NOT EXISTS (
                  SELECT * FROM tbuch
                  WHERE tbuch.verlagnr = tverlag.verlagnr
                 )
 ;
-- Variante3 Left Join und Projektion und Restriktion
SELECT
  tverlag.verlagnr, tverlag.verlag
FROM tverlag LEFT OUTER JOIN tbuch
     ON tverlag.verlagnr = tbuch.verlagnr
WHERE tbuch.buchnr IS NULL
;
-- Variante4
WITH zwi AS
(
  SELECT          tverlag.verlagnr FROM tverlag
  EXCEPT ----MINUS Oracle
  SELECT DISTINCT tbuch.verlagnr FROM tbuch
)
SELECT
  tverlag.verlagnr, tverlag.verlag
FROM tverlag INNER JOIN zwi
     ON tverlag.verlagnr = zwi.verlagnr
;
```

9.8.3 Buchnr, Preis und Titel des Buches mit der ISBN 3472864303.

```
BUCHNR      PREIS     TITEL
----------- --------- --------------------
          8      0,50 der Butt

SELECT
  TBUCH.BUCHNR, TBUCH.PREIS, TBUCH.TITEL
FROM   TBUCH
WHERE  TBUCH.BUCHNR IN
                (
                  SELECT TISBN.BUCHNR
                  FROM   TISBN
                  WHERE  TISBN.ISBN   = '3472864303'
                )
;
SELECT
  TBUCH.BUCHNR, TBUCH.PREIS, TBUCH.TITEL
FROM TBUCH
WHERE EXISTS (
               SELECT * FROM TISBN
               WHERE TISBN.BUCHNR = TBUCH.BUCHNR
               AND TISBN.ISBN   = '3472864303'
             )
;
SELECT
  TBUCH.BUCHNR, TBUCH.PREIS, TBUCH.TITEL
FROM TBUCH INNER JOIN TISBN
     ON TBUCH.BUCHNR = TISBN.BUCHNR
WHERE TISBN.ISBN = '3472864303'
;
WITH zwi AS
(
  SELECT          tbuch.buchnr FROM tbuch
  INTERSECT
  SELECT          tisbn.buchnr FROM tisbn
  WHERE tisbn.isbn = '3472864303'
)
SELECT
  tbuch.buchnr, tbuch.preis, tbuch.titel
FROM tbuch INNER JOIN zwi
     ON tbuch.buchnr = zwi.buchnr
;
```

9.8.4 Buchnr, Preis und Titel der Bücher die wenigstens einen Autor haben.

```
BUCHNR      PREIS TITEL
---------- ---------- ------------------------------
         1 null       C
         2 null       C
         5        3,5 Ansichten eines Clowns
         6       20,5 die Blechtrommel
         7      99,99 der Name der Rose
         8         ,5 der Butt
         9         55 DB2 fuer Sie
        12 null       a guide to db2
```

```
SELECT
  TBUCH.BUCHNR, TBUCH.PREIS, TBUCH.TITEL
FROM TBUCH
WHERE TBUCH.BUCHNR IN
                      (
                          SELECT TVAUTOR.BUCHNR
                          FROM TVAUTOR
                      )
;
SELECT
  XYZ.BUCHNR, XYZ.PREIS, XYZ.TITEL
FROM TBUCH XYZ
WHERE EXISTS (
                SELECT * FROM TVAUTOR
                WHERE TVAUTOR.BUCHNR = XYZ.BUCHNR
             )
;
SELECT DISTINCT
  TBUCH.BUCHNR, TBUCH.PREIS, TBUCH.TITEL
FROM TBUCH INNER JOIN TVAUTOR
     ON TBUCH.BUCHNR = TVAUTOR.BUCHNR
;
WITH zwi AS
(
  SELECT          tbuch.buchnr   FROM tbuch
  INTERSECT
  SELECT DISTINCT tvautor.buchnr FROM tvautor
)
SELECT
  tbuch.buchnr, tbuch.preis, tbuch.titel
FROM tbuch INNER JOIN zwi
     ON tbuch.buchnr = zwi.buchnr
;
```

9.8.5 Buchnr, Preis und Titel der Bücher die Autor 2 geschrieben hat.

```
    BUCHNR      PREIS TITEL
---------- ---------- ------------------------------
         6       20,5 die Blechtrommel
         8         ,5 der Butt
```

```
SELECT
  TBUCH.BUCHNR, TBUCH.PREIS, TBUCH.TITEL
FROM TBUCH
WHERE TBUCH.BUCHNR IN
                (
                 SELECT Tvautor.Buchnr
                 FROM   Tvautor
                 WHERE  Tvautor.Autornr=2
                )
;
SELECT
  XYZ.BUCHNR, XYZ.PREIS, XYZ.TITEL
FROM TBUCH XYZ
WHERE EXISTS (
               SELECT * FROM TVAUTOR
               WHERE TVAUTOR.BUCHNR = XYZ.BUCHNR
               AND TVAUTOR.AUTORNR = 2
             )
;
SELECT
  TBUCH.BUCHNR, TBUCH.PREIS, TBUCH.TITEL
FROM TBUCH INNER JOIN TVAUTOR
     ON TBUCH.BUCHNR = TVAUTOR.BUCHNR
WHERE  TVAUTOR.AUTORNR = 2
;
```

9.8.6 Buchnr, Preis und Titel der Bücher der Autoren mit Namen Grass.

```
BUCHNR      PREIS     TITEL
----------- --------- ------------------------------
          6     20,50 die Blechtrommel
          8      0,50 der Butt
```

```
SELECT
  TBUCH.BUCHNR, TBUCH.PREIS, TBUCH.TITEL
FROM TBUCH
WHERE TBUCH.BUCHNR IN
         (
           SELECT TVAUTOR.BUCHNR
           FROM TVAUTOR
           WHERE TVAUTOR.AUTORNR IN
               (
                SELECT TAUTOR.AUTORNR FROM TAUTOR
                WHERE TAUTOR.AUTOR = 'Grass'
               )
         )
;
SELECT
  XYZ.BUCHNR, XYZ.PREIS, XYZ.TITEL
FROM TBUCH XYZ
WHERE EXISTS
        (
         SELECT * FROM  TVAUTOR ABC
          WHERE ABC.BUCHNR = XYZ.BUCHNR
          AND   EXISTS
              (
                SELECT * FROM  TAUTOR
                WHERE TAUTOR.AUTORNR = ABC.AUTORNR
                AND   TAUTOR.AUTOR   = 'Grass'
              )
        )
;
SELECT DISTINCT
  TBUCH.BUCHNR, TBUCH.PREIS, TBUCH.TITEL
FROM
     (TBUCH INNER JOIN TVAUTOR
     ON TBUCH.BUCHNR = TVAUTOR.BUCHNR)
            INNER JOIN TAUTOR
     ON TVAUTOR.AUTORNR = TAUTOR.AUTORNR
WHERE TAUTOR.AUTOR = 'Grass'
;
```

9.8.7 Buchnr, Preis und Titel der Bücher ohne Autor.

```
BUCHNR      PREIS             TITEL
----------- ----------------- --------------------
11          NULL              Elvis in Heidelberg
18          99.99             Database Systems
27          99.99             die Jüdin von Toledo

SELECT
  TBUCH.BUCHNR, TBUCH.PREIS, TBUCH.TITEL
FROM   TBUCH
WHERE  TBUCH.BUCHNR NOT IN
                (
                  SELECT tvautor.BUCHNR
                  FROM   tvautor
                )
;
SELECT
  XYZ.BUCHNR, XYZ.PREIS, XYZ.TITEL
FROM TBUCH XYZ
WHERE NOT EXISTS (
                   SELECT * FROM tvautor
                   WHERE tvautor.BUCHNR = XYZ.BUCHNR
                  )
;
SELECT
  TBUCH.BUCHNR, TBUCH.PREIS, TBUCH.TITEL
FROM TBUCH LEFT OUTER JOIN tvautor
     ON TBUCH.BUCHNR = tvautor.BUCHNR
WHERE tvautor.autornr IS NULL
;
```

9.8.8 Buchnr, Preis und Titel der Bücher ohne Praemie.

```
BUCHNR      preis      TITEL
----------- ---------- ------------------------
1           NULL       C
2           NULL       C
5           3.50       Ansichten eines Clowns
6           20.50      die Blechtrommel
7           99.99      der Name der Rose
11          NULL       Elvis in Heidelberg
18          99.99      Database Systems
27          99.99      die Jüdin von Toledo
```

```
SELECT
  TBUCH.BUCHNR, TBUCH.PREIS, TBUCH.TITEL
FROM   TBUCH
WHERE  TBUCH.BUCHNR NOT IN
                (
                  SELECT tvautor.BUCHNR
                  FROM   tvautor
                  where Tvautor.Praemie is not null
                )
;
SELECT
  XYZ.BUCHNR, XYZ.PREIS, XYZ.TITEL
FROM TBUCH XYZ
WHERE NOT EXISTS (
                  SELECT * FROM tvautor
                  WHERE tvautor.BUCHNR = XYZ.BUCHNR
                  and Tvautor.Praemie is not null
                 )
;
SELECT
  TBUCH.BUCHNR, TBUCH.PREIS, TBUCH.TITEL
FROM TBUCH LEFT OUTER JOIN
     (
      select buchnr, Autornr, praemie
      from tvautor
      where Tvautor.Praemie is not null
     ) zwi
     ON TBUCH.BUCHNR = zwi.BUCHNR
WHERE zwi.autornr is null
;
```

9.8.9 Buchnr, Preis und Titel der Bücher ohne Praemie und ohne Isbn.

```
BUCHNR      preis      TITEL
----------- ---------- ------------------------
1           NULL       C
2           NULL       C
5           3.50       Ansichten eines Clowns
6           20.50      die Blechtrommel
7           99.99      der Name der Rose
11          NULL       Elvis in Heidelberg
18          99.99      Database Systems
27          99.99      die Jüdin von Toledo

--insert into Tisbn(buchnr, isbn, lfdnr)
--values ( 27, 'XXX' , 1)

SELECT
  TBUCH.BUCHNR, TBUCH.PREIS, TBUCH.TITEL
FROM   TBUCH
WHERE  TBUCH.BUCHNR NOT IN
                (
                 SELECT tvautor.BUCHNR
                  FROM   tvautor
                  where Tvautor.Praemie is not null
                )
AND    TBUCH.BUCHNR NOT IN
                (
                  SELECT tisbn.BUCHNR
                  FROM   tisbn
                )
;
SELECT
  XYZ.BUCHNR, XYZ.PREIS, XYZ.TITEL
FROM TBUCH XYZ
WHERE NOT EXISTS (
                  SELECT * FROM tvautor
                  WHERE tvautor.BUCHNR = XYZ.BUCHNR
                  and Tvautor.Praemie is not null
                  )
AND   NOT EXISTS (
                  SELECT * FROM tisbn
                  WHERE tisbn.BUCHNR = XYZ.BUCHNR
                 )
;
```

```
SELECT
  TBUCH.BUCHNR, TBUCH.PREIS, TBUCH.TITEL
FROM TBUCH LEFT OUTER JOIN
     (
      select buchnr, Autornr, praemie
      from tvautor
      where Tvautor.Praemie is not null
     ) zwi
     ON TBUCH.BUCHNR = zwi.BUCHNR
          LEFT OUTER JOIN
     (
      select buchnr, isbn
      FROM tisbn
     ) zwi22
     ON Tbuch.Buchnr = zwi22.buchnr
WHERE zwi.autornr is null AND zwi22.ISBN is null
;

--delete from Tisbn where Buchnr = 27
--;
```

9.8.10 Buchnr, Preis und Titel der Bücher ohne Praemie oder ohne Isbn.

```
BUCHNR      preis      TITEL
----------- ---------- ----------------------
1           NULL       C
2           NULL       C
5           3.50       Ansichten eines Clowns
6           20.50      die Blechtrommel
7           99.99      der Name der Rose
9           55.00      DB2 fuer Sie
11          NULL       Elvis in Heidelberg
18          99.99      Database Systems
27          99.99      die Jüdin von Toledo
```

```
SELECT TBUCH.BUCHNR, TBUCH.PREIS, TBUCH.TITEL
FROM   TBUCH
WHERE  TBUCH.BUCHNR NOT IN
                (
                  SELECT tvautor.BUCHNR
                  FROM   tvautor
                  where Tvautor.Praemie is not null
                )
OR    TBUCH.BUCHNR NOT IN
                (
                  SELECT tisbn.BUCHNR
                  FROM   tisbn
                )
;
SELECT XYZ.BUCHNR, XYZ.PREIS, XYZ.TITEL
FROM TBUCH XYZ
WHERE NOT EXISTS (
                  SELECT * FROM tvautor
                  WHERE tvautor.BUCHNR = XYZ.BUCHNR
                  and Tvautor.Praemie is not null
                  )
OR    NOT EXISTS (
                   SELECT * FROM tisbn
                   WHERE tisbn.BUCHNR = XYZ.BUCHNR
                  )
;
```

```
SELECT DISTINCT
TBUCH.BUCHNR, TBUCH.PREIS, TBUCH.TITEL
FROM TBUCH LEFT OUTER JOIN
     (select buchnr, Autornr, praemie
      from tvautor
      where Tvautor.Praemie is not null
     ) zwi
     ON TBUCH.BUCHNR = zwi.BUCHNR
           LEFT OUTER JOIN
     (select buchnr, isbn
      FROM tisbn
     ) zwi22
     ON Tbuch.Buchnr = zwi22.buchnr
WHERE zwi.autornr is null OR zwi22.ISBN is null
;
```

9.9 Skalare Operatoren und Funktionen, Testen Sie ihren Server

10

Anhang 1: Literatur

10 Anhang 1: Literatur

10.1 Bücher zum Thema Datenbankmanagement und Relationale Theorie

Informationssysteme und Datenbanken
C. A. Zehnder
B. G. Teubner Stuttgart 1998

An Introduction to Database Systems
C. J. Date

Addison-Wesley
Eighth Edition 2003

Practical Issues in Database Management
A Reference for the Thinking Practitioner

Fabian Pascal
Addison-Wesley 2000

SQL and Relational Theory
How to Write Accurate SQL Code

C. J. Date
O'Reilly 2009

TEMPORAL DATA AND THE RELATIONAL MODEL

A Detailed Investigation into the Application of Interval and Relation Theory to the Problem of Temporal Database Management

C. J. Date Hugh Darwen Nikos A. Lorentzos
Morgan Kaufmann Publishers 2003

und vor allem im Internet **DATABASE DEBUNKINGS**
www.dbdebunk.com

10.2 Bücher über SQL und Datenbanken

Relationale Datenbanken und SQL
Konzepte der Entwicklung und Anwendung

G. Matthiessen Michael Unterstein

Addison-Wesley 2003

SQL Der Schlüssel zu relationalen Datenbanken

G. Kuhlmann F. Müllmerstadt

rororo 2004

SQL IN A NUTSHELL
A Desktop Quick Reference

Kevin E. Kline with Daniel Kline & Brand Hunt

Second Edition

O'Reilly 2004

SQL Pocket Guide

Jonathan Gennick

Third Edition

O'Reilly 2010

10.3 Der SQL Standard

ISO/IEC FCD 9075-1:2011, Information technology – Database languages – SQL – Part

1: Framework (SQL/Framework)

ISO/IEC FCD 9075-2:2011, Information technology – Database languages – SQL – Part

2: Foundation (SQL/Foundation)

ISO/IEC FCD 9075-3:2008, Information technology – Database languages – SQL – Part

3: Call-Level Interface (SQL/CLI)

ISO/IEC FCD 9075-4:2011, Information technology – Database languages – SQL – Part

4: Persistent Stored Modules (SQL/PSM)

ISO/IEC FCD 9075-9:2008, Information technology – Database languages – SQL – Part

9: Management of External Data (SQL/MED)

ISO/IEC FCD 9075-10:2008, Information technology – Database languages – SQL – Part

10: Object Language Bindings (SQL/OLB)

ISO/IEC FCD 9075-11:2011, Information technology – Database languages – SQL – Part

11: Information and Definition Schemas (SQL/Schemata)

ISO/IEC FCD 9075-13:2008, Information technology – Database languages – SQL – Part

13: Java Routines and Types (SQL/JRT)

ISO/IEC FCD 9075-14:2011, Information technology – Database languages – SQL – Part

14: XML-Related Specifications (SQL/XML)

10.4 Bücher über den SQL Standard

A Guide to the SQL standard
C. J. Date with H.Darwen
Addison Wesley Longman, Inc.
4. Edition 1997 Covers CLI and PSM

SQL – Der Standard SQL/92 mit den Erweiterungen CLI und PSM Deutsche Ausgabe des amerikanischen Klassikers Ausblick auf SQL3
Chris J. Date Hugh Darwen
Addison Wesley Longman GmbH, 1998

Understanding the new SQL: a complete guide
Jim Melton, Alan R. Simon
Morgan Kaufmann Publishers
1993

SQL – The Standard Handbook based on the new SQL standard
(ISO 9075:1992(E))
S. J. Cannan, G. A. M. Otten
McGraw-Hill, 1993

Understanding SQL's Stored Procedures:
A Complete Guide to SQL/PSM
Jim Melton
Morgan Kaufmann Publishers 1998

SQL:1999
Understanding Relational Language Components
Jim Melton Alan R. Simon
Morgan Kaufmann Publishers 2002

Einführung in den Sprachkern von SQL-99
Wolfgang Panny mit Alfred Taudes
Springer-Verlag Berlin Heidelberg New York 2000

Advanced SQL:1999
Understanding Object-Relational and Other Advanced Features
Jim Melton
Morgan Kaufmann Publishers 2003

SQL:1999 & SQL:2003
Objektrelationes SQL, SQLJ & SQL/XML
Can Türker
dpunkt.verlag 2003

10.5 Microsoft SQL Server

Es gibt viele Bücher

Microsoft SQL Server 2012 T-SQL Fundamentals (**MSPress**)
By: **Itzik Ben-Gan**
Technical Editors: Gianluca Hotz, Herbert Albert
Publication Date: July 5, 2012
ISBN-10: 0735658145, ISBN-13: 978-0735658141, 440 pages

Microsoft SQL Server 2012 High-Performance T-SQL Using Window Functions (MSPress)
By: **Itzik Ben-Gan**
Technical Editor: **Adam Machanic**
Publication Date: April 10, 2012
ISBN-10: 0735658366, ISBN-13: 978-0735658363, 221 pages

Inside Microsoft SQL Server 2008: T-SQL Querying (**MSPress**)
By: Itzik Ben-Gan, Lubor Kollar, Dejan Sarka, Steve Kass
Technical Editors: Steve Kass, Umachandar Jayachandran
Foreword by: César Galindo-Legaria
Publication Date: March 25, 2009
ISBN-10: 0735626030, ISBN-13: 978-0735626034
Format: Paperback, 832 pages

10.6 Oracle

Es gibt viele Bücher.

10.7 DB2

Es gibt viele Bücher.

10.8 Bücher über SQL und JAVA

Understanding SQL and Java Together
A Guide to SQLJ, JDBC, and Related Technologies
Melton J. Eisenberg A.
Morgan Kaufmann Publishers 2000

S. White, M. Fisher, R. Cattell, G. Hamilton, M. Hapner
JDBC API Tutorial and Reference, Second Edition
Universal Data Access for the Java 2 Plattform
Addison-Wesley-Longmann 1999

Datenbanken & Java
JDBC, SQLJ, ODJG und JDO
Gunter Saake Kai-Uwe Sattler
dpunkt Verlag 2003

10.9 IBM DB2 SQL Reference for Cross-Platform Development Version 3.1

IBM Document No. SC26-3316-00 Formal Register of Existing Differences in SQL, September 1993 (diese Broschüre beschreibt im Detail die verschiedenen IBM-Dialekte im Vergleich mit SQL92 ENTRY LEVEL)

SQL Reference for Cross-Platform Development Version 3.1 (DB2 for: z/OS V9, iSeries V6.1, LUW V9.5)

Introduction

The *SQL Reference for Cross-Platform Development* uses the familiar format of the product SQL Reference manuals and includes information on the following topics:

- Database concepts
- Built-in functions
- Statements and queries
- SQL procedural language control statements
- Limits
- Programming in C, Java™, COBOL, REXX, and external routines in general
- SQL return codes (SQLSTATEs)
- CCSIDs

New in Version 3.1

The addition of newer versions of the DB2 products means the following great SQL features are now included:

- DECFLOAT data type
- Additional timestamp format
- Client special registers
- ROW CHANGE expressions
- New built-in functions
- order-by-clause and fetch-first-clause in a subselect
- SELECT FROM INSERT
- Full outer join
- ALTER FUNCTION
- IMPLICITLY HIDDEN columns
- row-change-timestamp columns

- RESTRICT on DROP of functions and procedures
- CURRENT DECFLOAT ROUNDING MODE special register and associated SET statement
- Other portability enhancements

10.10 IBM DB2 SQL Reference for Cross-Platform Development Version 4.0

SQL Reference for Cross-Platform Development Version 4.0

This book defines the DB2 SQL language elements that are common to the IBM®

DB2 Family of relational database products across the following environments:

Environment IBM Relational Database Product Short Name

z/OS® DB2 for z/OS Version 10 DB2 for z/OS

i operating system DB2 for IBM i Version 7.1 DB2 for i

Linux

UNIX

Windows

DB2 for the Linux, UNIX and Windows Platforms Version 9.7 DB2 for LUW

What's new for this book

The major new features covered in this book include:

- Support for the XML data type
- Datetime constants
- Support for extended indicator variables
- Enhancements to untyped parameter marker support
- Support for the XMLCAST specification
- New built-in functions:
- Aggregate Functions: XMLAGG
- Datetime Scalar Functions: ADD_MONTHS, EXTRACT, LAST_DAY, MONTHS_BETWEEN, NEXT_DAY, ROUND_TIMESTAMP and TRUNC_TIMESTAMP
- MQSeries Scalar Functions: MQREAD, MQREADCLOB, MQRECEIVE, MQRECEIVECLOB and MQSEND
- Numeric Scalar Functions: BITAND, BITANDNOT, BITNOT, BITOR, BITXOR
- String Scalar Functions: TRIM

- XML Scalar Functions: XMLATTRIBUTES, XMLCOMMENT, XMLCONCAT, XMLDOCUMENT, XMLELEMENT, XMLFOREST, XMLNAMESPACES, XMLPARSE, XMLPI, XMLSERIALIZE and XMLTEXT
- Table Functions: MQREADALL, MQREADALLCLOB, MQRECEIVEALL and MQRECEIVEALLCLOB
- TIMESTAMP_FORMAT and VARCHAR_FORMAT enhancements
- Support for CROSS JOIN
- Support for retrieving result sets in embedded SQL (ALLOCATE CURSOR and ASSOCIATE LOCATORS)
- Support for expressions in a CALL statement
- Support for identity columns and constraints on partitioned tables
- Support for distinct types in a temporary table
- Support for MERGE
- Support for CONCURRENT ACCESS RESOLUTION in PREPARE
- Support for the SET host variable statement
- Support for additional control statements in SQL functions
- Increases in several SQL limits

11

Anhang 2: Datums- und Zeitangaben

11 Anhang 2: Datums- und Zeitangaben

11.1 idiosyncrasy = persönliche Eigenart

„No vendor supports SQL-92 at the Full SQL level of conformance. All products include idiosyncrasies in their temporal support that render porting to other DBMSs difficult“.

Da jedes SQL-Datenbanksystem am Markt seine besonderen Eigenarten in Bezug auf Datums- und Zeitangaben besitzt, und die Produkte den SQL-Standard nur teilweise unterstützen, werden in diesem Kapitel für den interessierten Leser die Eigenarten für die Produkte DB2, Oracle und SQL Server dargestellt.

Die folgenden Seiten sind – was die Darstellung des SQL-Standards betrifft – eine knappe Zusammenfassung von 'Kapitel 17 Datums- und Zeitangaben' des Buches von Chris J. Date und Hugh Darwen, der deutschen Ausgabe des amerikanischen Klassikers über SQL. Der interessierte Leser wird auf diese Bücher verwiesen.

SQL – Der Standard

SQL/92 mit den Erweiterungen CLI und PSM

Deutsche Ausgabe des amerikanischen Klassikers

Chris J. Date Hugh Darwen

Addison Wesley Longman GmbH, 1998

A Guide to the SQL standard

C. J. Date with H.Darwen

Addison Wesley Longman, Inc.

4. Edition 1997 Covers CLI and PSM

11.2 Datentyp DATE und Datetime Constructor CURRENT_DATE

```
TABLE Tautor
autornr     autor               geburtsdatum
----------- ------------------- ------------
1           Boell               1917-12-21
2           Grass               1927-10-16
3           Eco                 1927-10-16
6           Scheifele           1932-02-15
10          Emil Hack           NULL
11          Frieda Holz         NULL
20          C. J. Date          1954-12-21
21          Colin J. White      1954-10-19
100         BUSCH               NULL
200         BUSCH               NULL
```

Die Table präsentiert nicht die richtigen Geburtstage der Autoren, die präsentierten Daten sind nicht korrekt.

- Welche Autoren sind im Oktober geboren?
- Welche Autoren sind 1927 geboren?
- Welche Autoren haben heute Geburtstag?
- Welche Autoren feiern in diesem Jahr ihren 51. Geburtstag?
- Wie viel Jahre, Monate, Tage sind bis heute seit dem Geburtstag vergangen?

```
DROP TABLE Tautor
;
CREATE TABLE Tautor
  (
   Autornr          INTEGER      NOT NULL
  ,Autor            VARCHAR(240) NOT NULL
  ,Geburtsdatum     DATE
  ,PRIMARY KEY (Autornr)
  )
 ;
DELETE FROM Tautor;

INSERT INTO tautor(autornr,autor) VALUES
(1, 'Boell');
INSERT INTO tautor(autornr,autor) VALUES
(2, 'Grass');
INSERT INTO tautor(autornr,autor) VALUES
(3, 'Eco');
INSERT INTO tautor(autornr,autor) VALUES
(6, 'Scheifele');
INSERT INTO tautor(autornr,autor) VALUES
(10,'Emil Hack');
INSERT INTO tautor(autornr,autor) VALUES
(11, 'Frieda Holz');
INSERT INTO tautor(autornr,autor) VALUES
(20,  'C. J. Date');
INSERT INTO tautor(autornr,autor) VALUES
(21, 'Colin J. White');
insert into tautor (autornr , autor) values
( 100 , 'BUSCH');
insert into tautor (autornr , autor) values
( 200 , 'BUSCH');

update tautor set geburtsdatum =
CAST('1917-12-21' AS DATE) where autornr = 1;
update tautor set geburtsdatum =
CAST('1927-10-16'AS DATE)  where autornr = 2;
update tautor set geburtsdatum =
CAST ('1927-10-16' AS DATE) where autornr = 3;
update tautor set geburtsdatum =
CAST('1932-02-15'AS DATE) where autornr = 6;
update tautor set geburtsdatum =
CAST('1954-12-21'AS DATE) where autornr = 20;
update tautor set geburtsdatum =
CAST('1954-10-19'AS DATE) where autornr = 21;

select autornr               as autornr
     , autor                 as autor
     , geburtsdatum          as geburtsdatum
from tautor
ORDER BY autornr
;
```

11.2.1 DATE ist nicht gleich DATE

SQL Standard	DB2	ORACLE	SQL Server
Types			
DATE	DATE	***DATE*** **ignoring the hour, minute, and second fields**	DATE SQL Server 2008 ***DATETIME*** ***ignoring the hour, minute, and second fields*** **SMALLDATETIME** **ignoring the hour, minute, and second fields**

11.2.2 CURRENT_DATE ist nicht gleich CURRENT_DATE

SQL Standard	DB2	ORACLE	SQL Server
Datetime Constructor			
CURRENT_DATE	CURRENT_DATE	TRUNC(CURRENT_DATE) TRUNC(CURRENT_DATE,'DD') TRUNC(CURRENT_TIMESTAMP)	CAST(GETDATE() AS DATE) CAST(CURRENT_TIMESTAMP AS DATE)

Oracle CURRENT_DATE und **SQL Server GETDATE()** liefern etwas anderes als YYYY-MM-DD.

DB2

```
SELECT CURRENT_DATE as heute
FROM Sysibm.sysdummy1
;
heute
----------
2012-04-30
```

Oracle

```
alter session set nls_date_format = 'YYYY-MM-DD HH24:MI:SS';
SELECT TRUNC(CURRENT_DATE, 'DD') as heute
FROM Dual
;
HEUTE
-------------------------
2012-04-30 00:00:00
```

SQL Server

```
SELECT CAST(GETDATE() AS DATE)  as heute
;
heute
----------
2012-04-30
```

11.2.3 DATE Oracle Eigenarten im Detail

INTEGER wird zu NUMBER(38) umgesetzt.

VARCHAR wird zu VARCHAR2 umgesetzt.

Der Datentyp DATE präsentiert yyyy-mm-dd hh:mi:ss.

```
DROP TABLE Tautor;
CREATE TABLE Tautor
  (
   Autornr          NUMBER(38)     NOT NULL
  ,Autor            VARCHAR2(240) NOT NULL
  ,Geburtsdatum     DATE
  ,PRIMARY KEY (Autornr)
  )
 ;
```

Sofern hh:mi:ss beim Datentyp DATE immer als 00:00:00 im System hinterlegt werden sollen, muss das über Constraints abgesichert werden.

```
ALTER TABLE Tautor ADD CONSTRAINT con_hh CHECK
(EXTRACT (HOUR FROM cast (geburtsdatum as TIMESTAMP(0)))= 0) ;
ALTER TABLE Tautor ADD CONSTRAINT con_mi CHECK
(EXTRACT (MINUTE FROM cast (geburtsdatum as TIMESTAMP(0)))= 0) ;
ALTER TABLE Tautor ADD CONSTRAINT con_ss CHECK
(EXTRACT (SECOND FROM cast (geburtsdatum as TIMESTAMP(0)))= 0) ;
```

Wenn diese Constraints nicht implementiert sind und hh:mi:ss immer ignoriert werden sollen, muss das in den UPDATE-, INSERT- und SELECT-Anweisungen, speziell bei CURRENT_DATE und in der WHERE-Klausel, besonders berücksichtigt werden.

```
TRUNC(CURRENT_DATE, 'DD')
TRUNC(geburtsdatum, 'DD')
```

Über einen Session-Parameter kann gesteuert werden, wie und welche Teile des DATE-Datentyps angelistet werden.

```
ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY-MM-DD';
ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY-MM-DD HH24:MI:SS';
```

Oracle präsentiert das Jahr 0, -1, …, -4711 und das Jahr -4712

Oracle Database can store dates in the Julian era, ranging from January 1, 4712 BCE through December 31, 9999 CE (Common Era, or 'AD'). Unless BCE ('BC' in the format mask) is specifically used, CE date entries are the default.

```
ALTER SESSION SET NLS_DATE_FORMAT = 'SYYYY-mm-dd'
;
select
 (DATE '-4712-01-01')
--,(DATE '-4712-01-01') -1  --Fehlermeldung mit -4713
,(DATE '-0000-01-01')
,(DATE '+0000-01-01')
,(DATE '+0001-01-01')
,(DATE '0001-01-01')
,(DATE '9999-01-01')
from dual
;
```

11.3 Welche Autoren haben heute Geburtstag?

11.3.1 DB2

```
update tautor set geburtsdatum = NULL
;
update tautor set geburtsdatum = cast ('21.12.1917' as date)
where autornr = 1;
update tautor set geburtsdatum =  '16.10.1927'
where autornr = 2;
update tautor set geburtsdatum = '1927-10-16'
where autornr = 3;
update tautor set geburtsdatum = DATE ('15.02.1932')
where autornr = 6;
update tautor set geburtsdatum = DATE ('21.12.1954')
where autornr = 20;
update tautor set geburtsdatum = DATE ('19.10.1954')
where autornr = 21;
ALTER Table tautor
      ADD zeitstempel TIMESTAMP(3)
      NOT NULL WITH DEFAULT CURRENT TIMESTAMP
;
```

--Zeilen, die beim ALTER TABLE schon vorhanden sind, erhalten den aktuellen Timestamp.

```
select
  autornr
, geburtsdatum
, zeitstempel
from tautor
order by autornr
;
AUTORNR     GEBURTSDATUM ZEITSTEMPEL
----------- ------------ --------------------------
          1 1917-12-21   2011-08-06-13.06.32.671
          2 1927-10-16   2011-08-06-13.06.32.671
          3 1927-10-16   2011-08-06-13.06.32.671
          6 1932-02-15   2011-08-06-13.06.32.671
         10 -            2011-08-06-13.06.32.671
         11 -            2011-08-06-13.06.32.671
         20 1954-12-21   2011-08-06-13.06.32.671
         21 1954-10-19   2011-08-06-13.06.32.671
        100 -            2011-08-06-13.06.32.671
        200 -            2011-08-06-13.06.32.671
```

Welche Autoren sind im Oktober geboren?

```
select
 autornr
,geburtsdatum
from tautor
where month(geburtsdatum) = 10
;
```

Welche Autoren sind 1927 geboren?

```
select autornr, geburtsdatum from tautor
where year(geburtsdatum) = 1927
;
select autornr, geburtsdatum
from tautor where
geburtsdatum BETWEEN DATE'1927-01-01' AND DATE'1927-12-31'
;
```

Welche Autoren haben heute Geburtstag?

```
select
 autornr
,geburtsdatum
from tautor
where month(geburtsdatum) = month(current date)
and day(geburtsdatum)     = day(current date)
;
```

Welche Autoren feiern in diesem Jahr ihren 51. Geburtstag?

```
select
 autornr
,year(current date) as aktuellesjahr
,year(geburtsdatum) as geburtsjahr
,year(current_date ) - year(geburtsdatum) as alter
from tautor
where year(current date) - year(geburtsdatum) = 51
;
```

Anlisten des Datums in verschiedenen Formaten

```
select
 autornr
,geburtsdatum
,char(geburtsdatum, eur) as datum1
,char(geburtsdatum, iso) as datum2
from tautor
ORDER BY geburtsdatum, autornr
;
```

11.3.2 Oracle

```
set NULL null;
ALTER SESSION SET NLS_DATE_FORMAT
= 'YYYY-MM-DD';
UPDATE tautor SET geburtsdatum = NULL
;
update tautor set geburtsdatum =
'1917-12-21' where autornr = 1;
update tautor set geburtsdatum =
DATE '1927-10-16' where autornr = 2;
update tautor set geburtsdatum =
CAST ('1927-10-16' AS DATE) where autornr = 3;
update tautor set geburtsdatum =
TO_DATE ('1932-02-15' , 'yyyy-mm-dd') where autornr = 6;
update tautor set geburtsdatum =
to_date ('21.12.1954' , 'dd.mm.yyyy') where autornr = 20;
update tautor set geburtsdatum =
to_date ('1954-19-10' , 'yyyy-dd-mm') where autornr = 21;

--alter table tautor drop column zeitstempel;
ALTER TABLE tautor
      ADD zeitstempel TIMESTAMP(3)
      DEFAULT CURRENT_TIMESTAMP
      CONSTRAINT con_zeitstempel NOT NULL
;
```

--Zeilen, die beim ALTER TABLE schon vorhanden sind, erhalten den aktuellen Timestamp.

```
ALTER SESSION SET NLS_DATE_FORMAT
= 'YYYY-MM-DD HH24:MI:SS';
ALTER SESSION SET NLS_TIMESTAMP_FORMAT
= 'YYYY-MM-DD HH24:MI:SS.FF';
select
 autornr                  as autornr
,geburtsdatum             as geburtdatum
,zeitstempel              as zeitstempel
from tautor
order by autornr
;
```

AUTORNR	GEBURTDATUM	ZEITSTEMPEL
1	1917-12-21 00:00:00	2011-08-08 13:15:03.109
2	1927-10-16 00:00:00	2011-08-08 13:15:03.109
3	1927-10-16 00:00:00	2011-08-08 13:15:03.109
6	1932-02-15 00:00:00	2011-08-08 13:15:03.109
10		2011-08-08 13:15:03.109
11		2011-08-08 13:15:03.109
20	1954-12-21 00:00:00	2011-08-08 13:15:03.109
21	1954-10-19 00:00:00	2011-08-08 13:15:03.109
100		2011-08-08 13:15:03.109
200		2011-08-08 13:15:03.109

Achtung: die Spalte Geburtsdatum ist vom Datentyp DATE. Das Format yyyy-mm-dd ist, auch wenn explizit mit CAST konvertiert wird, abhängig von `NLS_DATE_FORMAT`.

Welche Autoren sind im Oktober geboren?

```
select autornr, geburtsdatum
from tautor where
substr
( to_char(geburtsdatum, 'yyyy-mm-dd'), 6,2)
= '10'
;
select autornr, geburtsdatum
from tautor where
EXTRACT(MONTH FROM geburtsdatum) = 10
;
```

Welche Autoren sind 1927 geboren?

```
select autornr, geburtsdatum
from tautor where
substr
( to_char(geburtsdatum, 'yyyy-mm-dd'), 1,4)
= '1927'
;
select autornr, geburtsdatum
from tautor where
EXTRACT(YEAR FROM geburtsdatum) = 1927
;
select autornr, geburtsdatum
from tautor where
TRUNC(geburtsdatum,'DD')
     BETWEEN DATE'1927-01-01' AND DATE'1927-12-31'
;
select autornr, geburtsdatum
from tautor where
DATE'1927-01-01' <= geburtsdatum
AND geburtsdatum < DATE'1928-01-01'
;
```

Welche Autoren haben heute Geburtstag?

```
select autornr, geburtsdatum
from tautor where
   extract  (month from geburtsdatum)
  =extract  (month from CURRENT_DATE)
and
   extract  (day from geburtsdatum)
  =extract  (day from CURRENT_DATE)
;
```

Welche Autoren feiern in diesem Jahr ihren 51. Geburtstag?

```
Select autornr, geburtsdatum from tautor where
EXTRACT( YEAR FROM geburtsdatum) + 51
=
EXTRACT (YEAR FROM CURRENT_DATE)
;
```

Anlisten des Datums in verschiedenen Formaten

```
select
 autornr
,geburtsdatum
,to_char(geburtsdatum, 'yyyy-mm-dd')  as datum1
,to_char(geburtsdatum, 'dd.mm.yyyy')  as datum2
  from tautor
  ORDER BY geburtsdatum, autornr
;
```

11.3.3 SQL Server

```
UPDATE tautor SET geburtsdatum = NULL
;
SET DATEFORMAT ymd
update tautor set geburtsdatum =  ('1917-12-12') --???
where autornr = 1;
update tautor set geburtsdatum =    '1927-10-16' --???
where autornr = 2;
update tautor set geburtsdatum =  CAST('1927-10-16' AS DATE)
where autornr = 3;
set dateformat dmy
update tautor set geburtsdatum =  ('15.02.1932')
where autornr = 6;
update tautor set geburtsdatum =  ('19541221')
where autornr = 20;
set dateformat dmy
update tautor set geburtsdatum =  ('19/10/1954')
where autornr = 21;
ALTER TABLE tautor
      ADD zeitstempel DATETIME2(3)
      NOT NULL DEFAULT CURRENT_TIMESTAMP
;
```

--Zeilen, die beim ALTER TABLE schon vorhanden sind, erhalten den aktuellen Timestamp.

```
select
 autornr                  as autornr
,geburtsdatum             as geburtsdatum
,zeitstempel              as zeitstempel
from tautor
order by autornr
;
autornr     geburtsdatum zeitstempel
----------- ------------ ----------------------
1           1917-12-21   2011-08-04 14:07:09.170
2           1927-10-16   2011-08-04 14:07:09.170
3           1927-10-16   2011-08-04 14:07:09.170
6           1932-02-15   2011-08-04 14:07:09.170
10          NULL         2011-08-04 14:07:09.170
11          NULL         2011-08-04 14:07:09.170
20          1954-12-21   2011-08-04 14:07:09.170
21          1954-10-19   2011-08-04 14:07:09.170
100         NULL         2011-08-04 14:07:09.170
200         NULL         2011-08-04 14:07:09.170
```

Achtung: die Spalte Geburtsdatum ist vom Datentyp DATE, die Spalte Zeitstempel ist vom Datentyp DATETIME2(3)

Das Format yyyy-mm-dd ist beim Datentyp **DATE nicht language dependent** und **nicht abhängig** von SET DATEFORMAT.

Das Format yyyy-mm-dd ist beim **Datentyp SMALLDATETIME** und **DATETIME language dependent** und **abhängig** von SET DATEFORMAT.

--Welche Autoren sind im Oktober geboren?

```
select autornr, geburtsdatum from tautor
where datepart( mm, geburtsdatum) =10
;
select autornr,geburtsdatum from tautor
where month(geburtsdatum) = 10
;
```

--Welche Autoren sind 1927 geboren?

```
select autornr, geburtsdatum from tautor
where datepart( yy, geburtsdatum) = 1927
;
select autornr, geburtsdatum from tautor
where year(geburtsdatum) = 1927
;
select autornr, geburtsdatum from tautor
where geburtsdatum
BETWEEN cast('1927-01-01' as date)
    AND cast('1927-12-31' as date)
;
```

--Welche Autoren haben heute Geburtstag?

```
select autornr, geburtsdatum from tautor
where datepart( mm, geburtsdatum)
    = datepart( mm, getdate())
and   datepart( dd, geburtsdatum)
    = datepart( dd, getdate())
;
```

--Welche Autoren feiern in diesem Jahr ihren 51. Geburtstag?

```
select
    autornr
   ,geburtsdatum
from tautor
where datepart( yy, getdate())
    - datepart( yy, geburtsdatum) = 51
;
```

Anlisten des Datums in verschiedenen Formaten

```
select
 autornr
,geburtsdatum
,CONVERT(NVARCHAR(10), geburtsdatum, 120  )  as datum1
,CONVERT(NVARCHAR(10), geburtsdatum, 104  )  as datum2
from tautor
ORDER BY geburtsdatum, autornr
;
```

11.4 Präsentation [closed,open), Präsentation [closed,closed]

11.4.1 Ein Vertrag hat eine Laufzeit, eine Geltungsdauer.

Base Table fuer [closed,open) mit Integritätsbedingung von < bisopen mit „letztem Tag“ 30.12.9999 präsentiert wird `bisopen = 9999-12-31`

Tvertragcloope

schl	von	bisopen
1	0001-01-01	9999-12-31
2	0001-01-01	NULL
3	0001-01-01	2016-01-01
4	2000-01-01	2000-01-02
5	2013-01-17	**2013-01-18**

View fuer [closed,closed] mit „letztem Tag“ 30.12.9999 präsentiert wird `bisclosed = 9999-12-30`

vvertragcloclo

schl	von	bisclosed
1	0001-01-01	9999-12-30
2	0001-01-01	NULL
3	0001-01-01	2015-12-31
4	2000-01-01	2000-01-01
5	2013-01-17	**2013-01-17**

Welcher Vertrag ist am 31.12.2015 gültig?

Welcher Vertrag ist am 01.01.2016 gültig?

Angenommen, heute ist der 17.1.2013. Welcher Vertrag ist heute gültig?

Welcher Vertrag ist am 31.12.9999 gültig?

Diese Frage kann nicht beantwortet werden!

```
drop table tvertragcloope
;
create table tvertragcloope
(
  schl    INTEGER    not null
 ,von     DATE       not null
 ,bisopen DATE
 ,PRIMARY KEY(schl)
 ,CONSTRAINT con1_tvertragcloope CHECK (von < bisopen)
 ,CONSTRAINT con2_tvertragcloope CHECK
   (
       CAST('0001-01-01' AS DATE)<=von
       AND bisopen <= CAST('9999-12-31' AS DATE)
   )
)
;
--Oracle
--ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY-MM-DD';

--SQL Server bei DATETIME, SMALLDATETIME
--SET DATEFORMAT ymd

DELETE FROM tvertragcloope;
INSERT INTO tvertragcloope
(schl, von, bisopen)
VALUES  (1 , '0001-01-01', '9999-12-31');
INSERT INTO tvertragcloope
(schl, von, bisopen)
VALUES  (2 , '0001-01-01',         NULL);
INSERT INTO tvertragcloope
(schl, von, bisopen)
VALUES  (3 , '0001-01-01', '2016-01-01');
INSERT INTO tvertragcloope
(schl, von, bisopen)
VALUES  (4 , '2000-01-01', '2000-01-02');
Select schl, von, bisopen from tvertragcloope
;
--go SQL Server
drop view vvertragcloclo
;
--go SQL Server
CREATE VIEW vvertragcloclo AS
SELECT
    schl
   ,von
   --,bisopen - 1 DAY           as bisclosed    --DB2
   ,DATEADD(DAY, -1, bisopen) as bisclosed     --SQL Server
   --,bisopen - 1                 as bisclosed  --Oracle
FROM tvertragcloope
;
--Go
Select schl, von, bisclosed from vvertragcloclo
;
--Einfügen einer Testzeile mit [heute, morgen)
Insert into tvertragcloope
(schl, von, bisopen) values
--( 5, current_date, current_date + 1 DAY)          --DB2
--( 5, TRUNC(current_date), TRUNC(current_date)+ 1) --Oracle
  ( 5, CAST(GETDATE() AS DATE)                      --SQL Server
     , DATEADD(DAY,+1,CAST(GETDATE() AS DATE)))
;
```

11.4.2 Welcher Vertrag ist heute gültig? [closed,open]

Tvertragcloope

schl	von	bisopen
1	0001-01-01	9999-12-31
2	0001-01-01	NULL
3	0001-01-01	2016-01-01
4	2000-01-01	2000-01-02
5	2013-01-17	2013-01-18

Angenommen heute ist der 17.1.2013. Welcher Vertrag ist heute gültig?

schl	von	bisopen
1	0001-01-01	9999-12-31
2	0001-01-01	NULL
3	0001-01-01	2016-01-01
5	2013-01-17	2013-01-18

```
--DB2 [closed,open)
select CURRENT_DATE from sysibm.sysdummy1
;
select * from tvertragcloope
;
select * from tvertragcloope
where
(von <= CURRENT_DATE AND CURRENT_DATE < bisopen)
OR
(von <= CURRENT_DATE AND bisopen IS NULL)
;

--Oracle [closed,open)
select CURRENT_DATE from DUAL
;
select * from tvertragcloope
;
select * from tvertragcloope
where
(TRUNC(von,'DD') <= CURRENT_DATE AND
                    CURRENT_DATE < TRUNC(bisopen,'DD'))
OR
(TRUNC(von,'DD') <= CURRENT_DATE AND bisopen IS NULL)
;

--SQL Server [closed,open)
select getdate()
;
select CAST (getdate() as DATE)
;
select * from tvertragcloope
;
select * from tvertragcloope
where
(von <= GETDATE() AND GETDATE() < bisopen)
OR
(von <= GETDATE() AND bisopen IS NULL)
;
```

11.4.3 Welcher Vertrag ist heute gültig? [closed,closed]

vvertragcloclo

```
schl        von        bisclosed
----------- ---------- ----------
1           0001-01-01 9999-12-30
2           0001-01-01 NULL
3           0001-01-01 2015-12-31
4           2000-01-01 2000-01-01
5           2013-01-17 2013-01-17
```

Angenommen heute ist der 17.1.2013. Welcher Vertrag ist heute gültig?

```
schl        von        bisclosed
----------- ---------- ----------
1           0001-01-01 9999-12-30
2           0001-01-01 NULL
3           0001-01-01 2015-12-31
5           2013-01-17 2013-01-17
```

```
--DB2 [closed,closed]
select CURRENT_DATE from sysibm.sysdummy1
;
--update tvertragcloope
--set bisopen= CURRENT_DATE + 1 DAY
--WHERE schl = 4
--;
select * from vvertragcloclo
;
select * from vvertragcloclo
where
  (CURRENT_DATE BETWEEN von AND bisclosed)
OR
  (von <= CURRENT_DATE AND bisclosed IS NULL)
;

--Oracle [closed,closed]
--??alter session set NLS_DATE_FORMAT
--??= 'yyyy-mm-dd';
--??select CURRENT_DATE from DUAL;

alter session set NLS_DATE_FORMAT
= 'yyyy-mm-dd hh24:mi:ss';
select CURRENT_DATE from DUAL;
select trunc(current_date, 'DD') from dual
;
--update tvertragcloope
--set bisopen = TRUNC(CURRENT_DATE) +1
--WHERE schl = 4
;
select * from vvertragcloclo
;
select * from vvertragcloclo
where
 ( TRUNC(CURRENT_DATE)
     BETWEEN TRUNC(von) AND TRUNC(bisclosed) )
OR
 ( TRUNC(von) <= TRUNC(CURRENT_DATE)
                AND bisclosed IS NULL)
;

--SQL Server [closed,closed]
select getdate()
;
select CAST (getdate() as DATE)
;
--update tvertragcloope
--set bisopen =  DATEADD( DAY, +1, CAST(GETDATE() AS DATE))
--WHERE schl = 4
--;
select * from vvertragcloclo
;
select * from vvertragcloclo
where
 (CAST(GETDATE()AS DATE) BETWEEN von AND bisclosed)
OR
 (von<=CAST(GETDATE()AS DATE) AND bisclosed IS NULL)
;
```

11.5 Datumsarithmetik mit Tagen

11.5.1 Soviel Tage läuft ein Vertrag!

```
--Präsentation [closed,open)
SELECT
  schl, von, bisopen
  --DB2
  --,days(bisopen) - days(von)
  --Oracle
  --,(bisopen - von)
--SQL Server
  ,datediff(DAY,von,bisopen)
as tageinsgesamt
FROM tvertragcloope
;
Tvertragcloope
schl        von        bisopen    tageinsgesamt
----------- ---------- ---------- -------------
1           0001-01-01 9999-12-31 3652058 bzw. 3652060 Oracle
2           0001-01-01 NULL       NULL
3           0001-01-01 2016-01-01 735963
4           2000-01-01 2000-01-02 1
5           2013-01-17 2013-01-18 1

--Präsentation [closed,closed]
SELECT
  schl, von, bisclosed
  --DB2
  --,days(bisclosed) - days(von)  + 1
  --Oracle
  --,(bisclosed - von) + 1
  --SQL Server
  ,datediff (day,von,bisclosed) + 1
as tageinsgesamt
FROM vvertragcloclo
;
Vvertragcloclo

schl        von        bisclosed  tageinsgesamt
----------- ---------- ---------- -------------
1           0001-01-01 9999-12-30 3652058 bzw. 3652060 Oracle
2           0001-01-01 NULL       NULL
3           0001-01-01 2015-12-31 735963
4           2000-01-01 2000-01-01 1
5           2013-01-17 2013-01-17 1
```

Bei [closed,open) kann die Anzahl der Tage einfach berechnet werden.

Bei [closed,closed] muss zusätzlich ein Tag addiert werden.

Achtung: Es gibt in den Datenbanksystemen keinen Tag nach dem 31.12.9999.

die Variante „bisclosed+1Tag“

```
DB2        --,days(bisclosed + 1 day) - days(von)
Oracle     --,(bisclosed +1 -von)
SQL Server --,datediff(day, von, DATEADD(DAY,+1,bisclosed))
```

DB2 Achtung:

Sofern wir den '31.12.9999' als „letzten Tag“ definieren und präsentieren wollen ist nur die Präsentation [closed,closed] möglich und die *„korrekte SELECT-Anweisung“*

`days(bisclosed + 1 day) - days(von)`

liefert einen Laufzeitfehler wegen '31.12.9999' + 1 tag,

`days(bisclosed) - days(von) +1` aber liefert keinen Laufzeitfehler.

Oracle Achtung:

Sofern wir den '31.12.9999' als „letzten Tag“ definieren und präsentieren wollen ist nur die Präsentation [closed,closed] möglich und die *„korrekte SELECT-Anweisung“*

`bisclosed +1 -von`

liefert einen Laufzeitfehler wegen '31.12.9999' + 1 tag,

`(bisclosed -von) + 1` aber liefert keinen Laufzeitfehler.

SQL Server Achtung:

Sofern wir den '31.12.9999' als „letzten Tag“ definieren und präsentieren wollen ist nur die Präsentation [closed,closed] möglich und die *„korrekte SELECT-Anweisung“*

`datediff(day, von, DATEADD(DAY,+1,bisclosed))`

liefert einen Laufzeitfehler wegen '31.12.9999' + 1 tag,

`datediff(day, von, bisclosed) +1` aber liefert keinen Laufzeitfehler.

11.5.2 Der Julianische Tag und die Datenbanksysteme

Der erste Tag sei der 01.01.0001, der „letzte Tag“ sei der 30.12.9999.

[von,bisopen)	Wie viele Tage sind insgesamt vergangen?		
	DB2	**Oracle**	**SQL Server**
['01.01.0001' ,'31.12.9999')	3652058	**3652060**	3652058
['01.01.1900' ,'31.12.9999')	2958463	2958463	2958463
['01.01.0001' ,'31.12.1000')	365241	**365249** **8 Tage mehr**	365241
['05.10.1582' ,'15.10.1582')	10	**0** **Der Gregoriani-sche Kalender** **10 Tage fehlen**	10
['01.01.0001' ,'05.10.1582')	577725	**577737** **12 Tage mehr**	577725
['15.10. 1582' ,'31.12.9999')	3074323	3074323	3074323
['01.01.2011' , '31.01.2001')	30	30	30

1752 wurde in Großbritannien der Gregorianische Kalender eingeführt. In diesem Jahr folgte auf den 2. September 1752 der 14. September 1752.

Der Sybase SQL Server und der Microsoft SQL Server haben deshalb für den Datentyp DATETIME den ersten Tag als 1753-01-01 definiert.

11.5.2.1 DB2

Der erste Tag sei der 01.01.0001, der „letzte Tag“ sei der 30.12.9999.

```
--Präsentation [closed,open)
with
tabcloope as
( select
 cast ( '01.01.0001' as date)  as von
,cast ( '31.12.9999' as date)  as bisopen
from sysibm.sysdummy1
)
select
 von                             as von
,bisopen                         as bisopen
,days(bisopen) - days(von)       as tageinsgesamt
From tabcloope
;
VON        BISOPEN    TAGEINSGESAMT
---------- ---------- -------------
0001-01-01 9999-12-31       3652058
```

```
--Präsentation [closed,closed]
with
tabcloclo as
( select
 cast ( '01.01.0001' as date)  as von
,cast ( '30.12.9999' as date)  as bisclosed
from sysibm.sysdummy1
)
select
 von                             as von
,bisclosed                       as bisclosed
,days(bisclosed) - days(von)  + 1  as tageinsgesamt
--,days(bisclosed + 1 day) - days(von) as tageinsgesamt
From tabcloclo
;
VON        BISCLOSED  TAGEINSGESAMT
---------- ---------- -------------
0001-01-01 9999-12-30       3652058
```

11.5.2.2 Oracle

Der erste Tag sei der 01.01.0001, der "letzte" Tag sei der 30.12.9999.

```
--Präsentation [closed,open)
alter session set nls_date_format = 'dd.mm.yyyy'
;
with
tabcloope as
(
 select
  cast ('01.01.0001' as date) as von
 ,cast ('31.12.9999' as date) as bisopen
from dual
)
Select
 von
,bisopen
,(bisopen - von) as tageinsgesamt
From tabcloope
;
VON                  BISOPEN              TAGEINSGESAMT
-------------------- -------------------- --------------------
01.01.0001           31.12.9999           3652060
```

```
--Präsentation [closed,closed]
alter session set nls_date_format = 'dd.mm.yyyy'
;
with
tabcloclo as
(
 select
  cast ('01.01.0001' as date) as von
 ,cast ('30.12.9999' as date) as bisclosed
from dual
)
Select
 von
,bisclosed
,(bisclosed - von) + 1 as tageinsgesamt
--,(bisclosed +1 -von)   as tageinsgesamt
From tabcloclo
;
VON                  BISCLOSED            TAGEINSGESAMT
-------------------- -------------------- --------------------
01.01.0001           30.12.9999           3652060
```

11.5.2.3 SQL Server

Der erste Tag sei der 01.01.0001, der "letzte" Tag sei der 30.12.9999.

```
--Präsentation [closed,open)
with
tabcloope as
( select
 cast( '01.01.0001' as date ) as von
,cast( '31.12.9999' as date ) as bisopen
)
select
 von                          as von
,bisopen                      as bisopen
,datediff(day, von, bisopen) as tageinsgesamt
From tabcloope
;
von        bisopen    tageinsgesamt
---------- ---------- -------------
0001-01-01 9999-12-31 3652058
```

```
--Präsentation [closed,closed]
with
tabcloclo as
( select
 cast( '01.01.0001' as date)    as von
,cast( '30.12.9999' as date)    as bisclosed
)
select
 von                             as von
,bisclosed                       as bisclosed
,datediff (day,von,bisclosed) + 1  as tageinsgesamt
--,datediff(day, von, DATEADD(DAY,+1,bisclosed)) as tageinsgesamt
From tabcloclo
;
von        bisclosed  tageinsgesamt
---------- ---------- -------------
0001-01-01 9999-12-30 3652058
```

11.5.2.4 der Julianische Tag

Achtung:

`DB2 julian_day(...)` ist etwas anderes als `Oracle TO_CHAR(...,'J')`

DB2

```
select
julian_day(cast ( '0001-01-01' as date) )
from sysibm.sysdummy1
;
--1721426
select
sysibm.to_char(cast ( '0001-01-01' as date), 'J')
from sysibm.sysdummy1
;
--1721424
select
sysibm.to_date(1721424,'J')
from sysibm.sysdummy1
;
--0001-01-01-00:00:00.000000
```

Oracle

```
alter session set nls_date_format = 'YYYY-MM-DD'
;
select
to_char(cast ( '0001-01-01' as date), 'J')
from dual
;
--1721424
select
to_date(1721424,'J')
from dual
;
--0001-01-01
```

DB2

The JULIAN_DAY function returns an integer value that represents a number of days from January 1, 4713 B.C. (the start of the Julian date calendar) to the date that is specified in the argument.

Oracle

Using Julian Days

A Julian day number is the number of days since January 1, 4712 BC. Julian days allow continuous dating from a common reference. You can use the date format model "J" with date functions **`TO_DATE`** and **`TO_CHAR`** to convert between Oracle **`DATE`** values and their Julian equivalents.

Note:

Oracle Database uses the astronomical system of calculating Julian days, in which the year 4713 BC is specified as -4712. The historical system of calculating Julian days, in contrast, specifies 4713 BC as -4713. If you are comparing Oracle Julian days with values calculated using the historical system, then take care to allow for the 365-day difference in BC dates. For more information, see **`http://aa.usno.navy.mil/faq/docs/millennium.php`**.

SQL Server

```
select cast(cast((0) as datetime)as date);
----------
1900-01-01
```

```
--DB2
with zwi as
(
 select
 julian_day(cast ( '0001-01-01' as date) )    as anf
,julian_day(cast ( '2014-09-17' as date) )    as wert
,julian_day(cast ( '9999-12-31' as date) )    as end
from sysibm.sysdummy1
)
select
 anf
,wert
,end
,CAST ('0001-01-01' as DATE) + (wert-anf) days datumwert
,CAST ('0001-01-01' as DATE) + (end -anf) days  datumend
from zwi
;
--DB2 mit Oracle-Funktion TO_CHAR( datum,'J')
with zwi as
(
 select
 TO_CHAR(cast ( '0001-01-01' as date),'J' )    as anf
,TO_CHAR(cast ( '2014-09-17' as date),'J' )    as wert
,TO_CHAR(cast ( '9999-12-31' as date),'J' )    as end
from sysibm.dual
)
select
 anf
,wert
,end
,CAST ('0001-01-01' as DATE) + ((wert-anf)-2) days   datumwert
,CAST ('0001-01-01' as DATE) + ((end -anf)-2) days   datumend
from zwi
;
--Oracle
with zwi as
(
 select
 TO_CHAR(cast ( '0001-01-01' as date),'J' )    as anf
,TO_CHAR(cast ( '2014-09-17' as date),'J' )    as wert
,TO_CHAR(cast ( '9999-12-31' as date),'J' )    as end
from dual
)
select
 anf
,wert
,end
,CAST ('0001-01-01' as DATE) + ((wert-anf))    as datumwert
,CAST ('0001-01-01' as DATE) + ((end -anf))    as datumend
from zwi
;
```

11.5.3 der Anker 1.1.1900, der Julianische Tag 2415021

Wir wählen als Anker den 1.1.1900.

```
schl        von        bisopen
----------- ---------- ----------
1           1900-01-01 9999-12-31
2           1900-01-01 NULL
3           1900-01-01 2016-01-01
4           2000-01-01 2000-01-02
```

Wir konvertieren die Daten in den Julianischen Tag.

```
schl        von         bisopen
----------- ----------- -----------
1           2415021     5373484
2           2415021     NULL
3           2415021     2457389
4           2451545     2451546
```

Wir präsentieren die Daten wieder im Format yyyy-mm-dd.

```
schl        von        bisopen
----------- ---------- ----------
1           1900-01-01 9999-12-31
2           1900-01-01 NULL
3           1900-01-01 2016-01-01
4           2000-01-01 2000-01-02
```

--DB2

```
UPDATE tvertragcloope
set   von = cast( '1900-01-01'  as date )
where von = cast( '0001-01-01'  as date )
;
select
 schl
,von
,bisopen
from tvertragcloope
;
select
 schl
,JULIAN_DAY(von)     as von
,JULIAN_DAY(bisopen) as bisopen
from tvertragcloope
;
--------------
with zwi as
(
select
 schl
,JULIAN_DAY(von)     as von
,JULIAN_DAY(bisopen) as bisopen
from tvertragcloope
)
select
schl
,CAST('1900-01-01' as date)+(von    -2415021)days  as von
,CAST('1900-01-01' as date)+(bisopen-2415021)days as bisopen
from zwi
;
```

--ORACLE

```
ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY-MM_DD';
UPDATE tvertragcloope
set   von = cast( '1900-01-01'  as date )
where von = cast( '0001-01-01'  as date )
;
select
 schl
,von
,bisopen
from tvertragcloope
;
select
 schl
,TO_CHAR(von,'J')     as von
,TO_CHAR(bisopen,'J') as bisopen
from tvertragcloope
;
--------------
with zwi as
(
select
 schl
,TO_CHAR(von,'J')     as von
,TO_CHAR(bisopen,'J') as bisopen
from tvertragcloope
)
select
schl
,CAST('1900-01-01' as date)+(von    -2415021)  as von
,CAST('1900-01-01' as date)+(bisopen-2415021)  as bisopen
from zwi
;
```

--SQL Server

```
UPDATE tvertragcloope
set   von = cast( '1900-01-01'  as date )
where von = cast( '0001-01-01'  as date )
;
select
 schl
,von
,bisopen
from tvertragcloope
;
select
 schl
,2415021+DATEDIFF(DAY, CAST('1900-01-01' AS DATE), von) as von
,2415021+DATEDIFF(DAY, CAST('1900-01-01' AS DATE), bisopen) as bisopen
from tvertragcloope
;
--------------
with zwi as
(
select
 schl
,2415021+DATEDIFF(DAY, CAST('1900-01-01' AS DATE), von) as von
,2415021+DATEDIFF(DAY, CAST('1900-01-01' AS DATE), bisopen) as bisopen
from tvertragcloope
)
select
schl
,DATEADD(DAY, von-2415021,     CAST('1900-01-01' as DATE)) as von
,DATEADD(DAY, bisopen-2415021, CAST('1900-01-01' as DATE)) as bisopen
from zwi
;
```

11.5.4 Datumsarithmetik mit Tagen, Variationen

Arithmetik mit Zahlen

bisopen = von + x
bisopen – x = von
bisopen-von = x

Datumsarithmetik mit Tagen

bisopen = von „tageplus“ x tage
bisopen „tageminus“ x tage = von
bisopen „tagediff“ von = x tage

„tageplus“ Wir benötigen eine Funktion/Methode für die Addition von Tagen zu einem Wert des Datentyps DATE.

„tageminus“ Wir benötigen eine Funktion/Methode für die Subtraktion von Tagen von einem Wert des Datentyps DATE.

„tagediff“ Wir benötigen eine Funktion/Methode zur Berechnung der Tage *„zwischen zwei Werten“* des Datentyps DATE.

Tvertragcloope

```
schl        von        bisopen
----------- ---------- ----------
1           0001-01-01 9999-12-31
2           0001-01-01 NULL
3           0001-01-01 2016-01-01
4           2000-01-01 2000-01-02
```

x=„tagediff“ [von, bisopen)

eine Zeitdauer von x tagen

„tageplus“

von + x Tage

„tageminus“

bisopen -x Tage

--Demonstration
für den Vertrag schl=4

„tagediff“ [von, bisopen) eine Zeitdauer von Tag

„tageplus“ von + 3 Tage

„tageminus“ bisopen -3 Tage

```
schl        von        bisopen    anztage     vonplusx   bisopenminusy
----------- ---------- ---------- ----------- ---------- -------------
4           2000-01-01 2000-01-02 1           2000-01-04 1999-12-30
```

--Demonstration mit DB2

```
select
 schl
,von
,bisopen
,(DAYS(bisopen) - DAYS(von))          as anztage
,von     + 3 DAYS                     as vonplusx
,bisopen - 3 DAYS                     as bisopenminusy
from tvertragcloope
where schl=4
;
```

Achtung DB2

- bisopen-von gibt eine DB2 DATE Duration yyyymmdd. zurück
- eine DB2 DATE Duration yyyymmdd. ist eine interpretierte Dezimalzahl
- die Arithmetik mit einer DATE Duration hat viele Tücken

Demonstration mit Oracle

```
ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY-MM-DD';
select
 schl
,von
,bisopen
,(bisopen - von)                  as anztage
,von     + 3                      as vonplusx
,bisopen - 3                      as bisopenminusy
from tvertragcloope
where schl=4
;
```

Achtung Oracle

Was wird bei dieser Rechnerei angelistet? Das Jahr 0 oder -1 oder -2?

```
ALTER SESSION SET NLS_DATE_FORMAT = 'SYYYY-MM-DD';
select CAST ('0001-01-01'as date) -1   from dual;
select CAST ('0001-01-01'as date) -367 from dual;
```

--Demonstration mit SQL Server

```
select
 schl
,von
,bisopen
,DATEDIFF(DAY, von, bisopen) as anztage
,DATEADD(DAY, +3, von)      as vonplusx
,DATEADD(DAY, -3, bisopen)  as bisopenminusy
from tvertragcloope
where schl=4
;
```

11.6 Intervalle, Perioden und Allen's Operatoren

11.6.1 ALLEN's Operatoren [closed,closed] equals, meets, includes, overlaps, …

Eine Menge von Operatoren kann definiert werden um zu testen, ob zwei Intervalle gleich sind, ob sie sich treffen, ob sie sich überlappen, ob ein Intervall das andere enthält, und so weiter. Diese Operatoren sind als Allen's Operatoren bekannt.

Definition der Operatoren bei [closed,closed].

```
int1=[i1.von,i1.bisclosed]
int2=[i2.von,i2.bisclosed]

int1 equals int2
genau dann wenn
(i1.von=i2.von AND i1.bisclosedclosed=i2.bisclosed)

int1 includes int2
genau dann wenn
(i1.von<=i2.von AND i2.bisclosed<=i1.bisclosed)

int1 before int2
genau dann wenn
(i1.bisclosed<i2.von)

int1 meets int2
genau dann wenn
(i2.von=i1.bisclosed+1 OR i1.von=i2.bisclosed+1)

int1 overlaps int2
genau dann wenn
(i1.von<=i2.bisclosed and i2.von<=i1.bisclosed)

int1 merges int2
genau dann wenn
(int1 overlaps int2 OR int1 meets int2)

int1 begins int2
genau dann wenn
(i1.von=i2.von AND i1.bisclosed<=i2.bisclosed)

int1 ends int2
genau dann wenn
(i1.bisclosed=i2.bisclosed AND i1.von>=i2.von)
```

Interessant ist die Definition von overlaps!

Bei meets muss gerechnet werden.

Beachten Sie bitte, dass wir bei der Definition der Operatoren das mögliche Auftreten von NULL nicht berücksichtigt haben.

11.6.2 ALLEN's Operatoren [closed,open) equals, meets, includes, overlaps, …

Eine Menge von Operatoren kann definiert werden um zu testen, ob zwei Intervalle gleich sind, ob sie sich treffen, ob sie sich überlappen, ob ein Intervall das andere enthält, und so weiter. Diese Operatoren sind als Allen's Operatoren bekannt.

Definition der Operatoren bei [closed,open).

```
int1=[i1.von,i1.bisopen]
int2=[i2.von,i2.bisopen]

int1 equals int2
genau dann wenn
(i1.von=i2.von AND i1.bisopen=i2.bisopen)

int1 includes int2
genau dann wenn
(i1.von<=i2.von AND i2.bisopen<=i1.bisopen)

int1 before int2
genau dann wenn
(i1.bisopen-1<i2.von)

int1 meets int2
genau dann wenn
(i2.von=i1.bisopen OR i1.von=i2.bisopen)

int1 overlaps int2
genau dann wenn
(i1.von<=i2.bisopen-1 and i2.von<=i1.bisopen-1)

int1 merges int2
genau dann wenn
(int1 overlaps int2 OR int1 meets int2)

int1 begins int2
genau dann wenn
(i1.von=i2.von AND i1.bisopen<=i2.bisopen)

int1 ends int2
genau dann wenn
(i1.bisopen=i2.bisopen AND i1.von>=i2.von)
```

Interessant ist die Definition von overlaps!

Bei before und overlaps muss gerechnet werden.

Beachten Sie bitte, dass wir bei der Definition der Operatoren das mögliche Auftreten von NULL nicht berücksichtigt haben.

11.6.3 includes [closed,closed] Welches Intervall enthält ein anderes?

```
vvertragcloclo
schl        von        bisclosed
----------- ---------- ----------
1           0001-01-01 9999-12-30
2           0001-01-01 NULL
3           0001-01-01 2015-12-31
4           2000-01-01 2000-01-01
```

Anlisten von Verträgen, wobei ein Vertrag den anderen enthält (includes).

```
schl1_aussen i1von      i1bisclosed schl2_innen i2von      i2bisclosed
------------ ---------- ----------- ----------- ---------- -----------
1             0001-01-01 9999-12-30  3           0001-01-01 2015-12-31
1             0001-01-01 9999-12-30  4           2000-01-01 2000-01-01
3             0001-01-01 2015-12-31  4           2000-01-01 2000-01-01
```

```
select schl, von, bisclosed from vvertragcloclo order by schl
;
select
       i1.schl       as schl1_aussen
      ,i1.von        as i1von
      ,i1.bisclosed  as i1bisclosed
      ,i2.schl       as schl2_innen
      ,i2.von        as i2von
      ,i2.bisclosed  as i2bisclosed
from
    vvertragcloclo i1
  , vvertragcloclo i2
where 1=1
--nicht der gleiche Vertrag
AND (NOT (i1.schl=i2.schl))
--i1 includes i2
AND (i1.von<=i2.von AND i2.bisclosed<=i1.bisclosed)
order by i1.schl, i2.schl
;
```

Beachten Sie bitte, dass wir das mögliche Auftreten von NULL für den letzten Tag nicht berücksichtigt haben.

11.6.4 meets [closed,closed] Welches Intervall trifft ein anderes?

```
--Test mit SQL Server
delete from tvertragcloope where schl in ( 66, 55 );
INSERT INTO tvertragcloope (schl, von, bisopen)
VALUES  (66 , '9999-01-01', '9999-01-02');
INSERT INTO tvertragcloope (schl, von, bisopen)
VALUES  (55 , '9999-01-02', '9999-01-03');
Select schl, von, bisclosed from vvertragcloclo order by schl
;
Vvertragcloclo
schl        von        bisclosed
----------- ---------- ----------
1           0001-01-01 9999-12-30
2           0001-01-01 NULL
3           0001-01-01 2015-12-31
4           2000-01-01 2000-01-01
55          9999-01-02 9999-01-02
66          9999-01-01 9999-01-01
```

Anlisten von Verträgen, wobei ein Vertrag den anderen triff (meets).

```
schlvorher  schlnachher
----------- -----------
66           55

select
 i1.schl  as schlvorher
,i2.schl  as schlnachher
from
    vvertragcloclo i1
  , vvertragcloclo i2
where 1=1
--nicht der gleiche Vertrag
AND (NOT i1.schl=i2.schl)
--i1 meets i2
AND ( i2.von=DATEADD(DAY, 1,i1.bisclosed)
      OR
      i1.von=DATEADD(DAY,1, i2.bisclosed)
    )
AND i1.von < i2.von
order by i1.schl, i2.schl
;
```

Beachten Sie bitte, dass wir das mögliche Auftreten von NULL für den „letzten Tag“ nicht berücksichtigt haben.

```
delete from tvertragcloope where schl in ( 66, 55 );
```

11.6.5 overlaps und [closed,closed] Welche Intervalle überlappen sich?

```
--Test mit SQL Server
delete from tvertragcloope where schl in ( 66, 55 );
delete from tvertragcloope where schl in (77 );
INSERT INTO tvertragcloope (schl, von, bisopen)
VALUES  (77 , '2015-12-31', '2016-01-02');
select schl, von, bisclosed from vvertragcloclo order by schl
;
```

vvertragcloclo

schl	von	bisclosed
1	0001-01-01	9999-12-30
2	0001-01-01	NULL
3	0001-01-01	2015-12-31
4	2000-01-01	2000-01-01
77	2015-12-31	2016-01-01

Anlisten der überlappenden Zeiträume von Verträgen, wobei ein Vertrag den anderen überlappt (overlaps).

schl1	schl2	vonmin	bisclosedmax
1	3	0001-01-01	2015-12-31
1	4	2000-01-01	2000-01-01
1	77	2015-12-31	2016-01-01
3	4	2000-01-01	2000-01-01
3	77	2015-12-31	2015-12-31

```
select
       i1.schl               as schl1
      ,i2.schl               as schl2
      ,case
       when i1.von <= i2.von then i2.von
       else                       i1.von
       end                   as vonmin
      ,case
       when i1.bisclosed <= i2.bisclosed then i1.bisclosed
       else                                   i2.bisclosed
       end                  as bisclosedmax
from
    vvertragcloclo i1
  , vvertragcloclo i2
where 1=1
--nicht der gleiche Vertrag
AND (NOT i1.schl=i2.schl)
--i1 overlaps i2
AND (i1.von<=i2.bisclosed and i2.von<=i1.bisclosed)
and i1.schl < i2.schl
order by schl1, schl2
;
```

Beachten Sie bitte, dass wir das mögliche Auftreten von NULL für den „letzten Tag“ nicht berücksichtigt haben.

```
delete from tvertragcloope where schl in (77 );
```

11.7 [closed,open) DATE und TIMESTAMP und CAST

Das erste Jahr sei 0001, das letzte Jahr sei 9998.
Dann ist die Präsentation [closed,closed] und [closed,open) möglich:
[0001, 9998] und [0001, 9999)

Der erste Tag sei der 01.01.0001, der letzte Tag sei der 31.12.9998.
Dann ist die Präsentation [closed,closed] und [closed,open) möglich:
[0001-01-01, 9998-12-31] und [0001-01-01, 9999-01-01)

Die erste Millisekunde sei 0001-01-01 00:00:00.000, die letzte sei 9998-12-31-23.59.59.999.
Dann ist die Präsentation [closed,closed] und [closed,open) möglich.
[0001-01-01 00:00:00.000,9998-12-31-23.59.59.999] **.**

Bei [closed,open) entstehen beim CAST von DATE nach TIMESTAMP(3) keine Lücken.

```
   [0001-01-01, 9999-01-01)
   >>> CAST >>>
   [0001-01-01 00:00:00.000, 9999—01-01 00:00:00.000)
```

```
--DB2
with tab as
( select
    cast( '0001-01-01' as DATE) as von
   ,cast( '9999-01-01' as DATE) as bisopen
 FROM SYSIBM.SYSDUMMY1
)
 select von,bisopen
  ,CAST(von     as timestamp(3))   as vontstp
  ,CAST(bisopen as timestamp(3))   as bisopentstp
 from tab
 ;
 von        bisopen    vontstp                  bisopentstp
---------- ---------- ------------------------ ------------------------
0001-01-01 9999-01-01 0001-01-01-00:00:00.000  9999-01-01-00:00:00.000

--Oracle
ALTER SESSION SET NLS_DATE_FORMAT      = 'YYYY-MM-DD HH24:MI:SS';
ALTER SESSION SET NLS_TIMESTAMP_FORMAT = 'YYYY-MM-DD HH24:MI:SS.FF';
with tab as
( select
    cast( '0001-01-01' as DATE) as von
   ,cast( '9999-01-01' as DATE) as bisopen
 FROM DUAL
)
 select von,bisopen
  ,CAST(von     as timestamp(3))   as vontstp
  ,CAST(bisopen as timestamp(3))   as bisopentstp
 from tab
 ;
--SQL Server
with tab as
( select
    cast( '0001-01-01' as DATE) as von
   ,cast( '9999-01-01' as DATE) as bisopen
)
 select von,bisopen
  ,CAST(von     as DATETIME2(3))   as vontstp
  ,CAST(bisopen as DATETIME2(3))   as bisopentstp
 from tab
 ;
 von        bisopen    vontstp                  bisopentstp
---------- ---------- ------------------------ ------------------------
0001-01-01 9999-01-01 0001-01-01 00:00:00.000  9999-01-01 00:00:00.000
```

11.8 Und jetzt etwas zur Unterhaltung! DB2 und Oracle

DB2 versteht inzwischen den SQL-Dialekt von Oracle (z. B. select * from sysibm.dual;), manchmal kommt es aber zu einem kleinen Missverständnis

--Wie viele Tage sind vergangen?

```
with
tabcloope as
( select
 cast ( '01.01.0001' as date)  as von
,cast ( '31.12.9999' as date)  as bisopen
,cast ( '01.01.2012' as date)  as vonx
,cast ( '31.12.2012' as date)  as bisopenx
from sysibm.sysdummy1
)
select
 von                            as von
,bisopen                        as bisopen
,days(bisopen) - days(von)      as tageinsgesamt
From tabcloope
;
--VON        BISOPEN    TAGEINSGESAMT
----------- ---------- -------------
--0001-01-01 9999-12-31       3652058
```

--days(bisopen) – days(von)
--das ist DB2 Syntax
--das Ergebnis ist eine Ganzahl 3652058

```
----------------------------------
with
tabcloope as
(
 select
 cast ( '01.01.0001' as date)  as von
,cast ( '31.12.9999' as date)  as bisopen
,cast ( '01.01.2012' as date)  as vonx
,cast ( '31.12.2012' as date)  as bisopenx
from sysibm.dual
)
Select
 Von             as von
,bisopen         as bisopen
,(bisopen - von) as tageinsgesamt
From tabcloope
;
```

--(bisopen – von)
--das ist DB2 Syntax
--das Ergebnis ist eine date duration 99981130.

--(bisopen – von)
--das ist aber auch Oracle Syntax
--das Ergebnis auf einem Oracle-System wäre eine Ganzzahl 3652060

11.9 SQL Standard

Die folgenden Seiten sind – was die Darstellung des SQL-Standards betrifft – eine knappe Zusammenfassung von 'Kapitel 17 Datums- und Zeitangaben' des Buches von Chris J. Date und Hugh Darwen, der deutschen Ausgabe des amerikanischen Klassikers über SQL. Der interessierte Leser wird auf diese Bücher verwiesen.

SQL – Der Standard
SQL/92 mit den Erweiterungen CLI und PSM
Deutsche Ausgabe des amerikanischen Klassikers
Chris J. Date Hugh Darwen
Addison Wesley Longman GmbH, 1998

A Guide to the SQL standard
C. J. Date with H.Darwen
Addison Wesley Longman, Inc.
4. Edition 1997 Covers CLI and PSM

11.9.1 Datentypen

```
DATE

TIME

TIMESTAMP


INTERVAL


TIME WITH TIME ZONE

TIMESTAMP WITH TIME ZONE
```

Angaben zu den Datentypen betreffen folgende Komponenten:

```
YEAR MONTH DAY HOUR MINUTE SECOND
```

Bis auf SECOND sind die Werte dieser Felder immer ganzzahlig.

- DATE TIMESTAMP repräsentieren eine absolute Position auf der Zeitachse, TIME einen absoluten Tageszeitpunkt.
- Ein INTERVAL ist ein Zeitdauer wie etwa "3 Jahre" oder "90 Tage" oder "5 Minuten 30 Sekunden".
- Die Subtraktion der Zeit "9:00" von "10.15" ergibt das INTERVAL "1 Stunde 15 Minuten".

TIME ZONE Zeitzone

"WARNUNG: ... die Unterstützung in SQL für Zeitzonen ist nicht nur kompliziert, sondern auch etwas konfus – an manchen Stellen sogar inkorrekt und widersprüchlich zu sich selbst ..."

"Von dieser Warnung abgesehen ist der grundlegende Punkt recht simpel: Natürlich-sprachliche Ausdrücke wie "10:00" sind inhärent mehrdeutig – ihre Interpretation hängt von der Zeitzone ab, in der sie benutzt werden. Beispielsweise kann genau derselbe Zeitpunkt von jemandem in San Francisco, Kalifornien, als "10:00" bezeichnet werden, aber als "18:00" von jemandem in London, England und als "20:00" von jemandem in Helsinki, Finnland. Mit anderen Worten, diese drei Ausdrücke, obwohl ihre externen Repräsentationen alle unterschiedlich sind, könnten alle dieselbe Denotation haben, d. h. sie denotieren alle denselben absoluten Zeitpunkt. Die Datentypen TIME WITH TIME ZONE und TIMESTAMP WITH TIME ZONE sind dazu gedacht, mit solchen Fragen umgehen zu können."

Chris J. Date Hugh Darwen, Seite 296

```
CREATE TABLE T
(..., startzeit TIME WITH TIME ZONE, ...);
```

in San Francisco ist es 10 Uhr beim INSERT:

```
INSERT INTO T ( ..., startzeit, ...)
       VALUES ( ..., TIME '10:00:00-08:00', ...);
```

in London wird 18 Uhr angezeigt! TIME '18:00:00+00:00'

in Helsinki wird 20 Uhr angezeigt! TIME '20:00:00+02:00'

zusammen mit dem Displacement

11.9.2 Definition von Datentypen

Definitionen von DATE TIME TIMESTAMP Datentypen:

```
DATE
TIME
TIME (6)

TIMESTAMP
TIMESTAMP (10)
```

Die Feldwerte sind wie folgt begrenzt:

```
YEAR    0001 bis 9999

MONTH   01 bis 12

DAY     01 bis 31

HOUR    00 bis 23

MINUTE  00 bis 59

SECOND  00.000... bis 61.999...
```

'Die merkwürdige obere Grenze für SECOND-Werte ... wird durch folgenden Auszug aus dem Standard erklärt: "On occasion, UTC is adjusted by the omission [...] or insertion of a leap second in order to maintain synchronization with sidereal time ..." '.

'...UTC entspricht dem, was als Greenwich Mean Time (GMT) bezeichnet wurde (und immer noch wird)...'

Chris J. Date Hugh Darwen, S 293, S 297

Definitionen von INTERVAL-Datentypen:

```
INTERVAL YEAR
INTERVAL YEAR TO MONTH
INTERVAL MONTH

INTERVAL DAY(3)
INTERVAL HOUR(3) TO MINUTE
INTERVAL MINUTE
INTERVAL SECOND (5,3)
INTERVAL DAY(3) TO SECOND(3)
```

Die Feldwerte in nichtführender Position sind begrenzt:

```
MONTH   01 bis 11

HOUR    00 bis 23
MINUTE  00 bis 59
SECOND  00.000... bis 59.999...
```

- Ein INTERVAL ist vom Typ Jahr-Monat bzw. Tag-Zeit.
- Der Typ Monat-Tag ist nicht möglich.
- INTERVAL MONTH TO DAY ist keine gültige Definition, da verschiedene Monate eine unterschiedliche Anzahl von Tagen haben. Ein INTERVAL wie "2 Monate 10 Tage" ist nicht eindeutig, sind es "70 Tage" oder "71 Tage" oder "72" Tage? Was ist mit Februar?
- Werte vom Typ INTERVAL können positiv oder negativ sein.
- INTERVAL SECOND (5,3) bedeutet 5 Ziffern im ganzzahligen Bereich und 3 Ziffern im Bruch. **Beachten Sie bitte den Unterschied zwischen dieser Regel und der Regel für Daten vom Typ NUMBERIC(5,3) oder DECIMAL(5,3).**

11.9.3 Literale

Literal Datum

```
DATE '1941-01-18'
```

Literal Zeit

```
TIME '09:30:00'
TIME '09:30:00.75'
```

Literal Zeitstempel

```
TIMESTAMP '1941-01-18 09:30:00.75'
TIMESTAMP '1941-01-18'
```

Literal INTERVAL Jahr-Monat mit Separator -

```
INTERVAL '1'  YEAR
INTERVAL '+1' YEAR
INTERVAL '-1' YEAR
INTERVAL ' 5' MONTH
INTERVAL '+5' MONTH
INTERVAL '-5' MONTH
INTERVAL ' 2-6' YEAR TO MONTH
INTERVAL '+2-6' YEAR TO MONTH
INTERVAL '-2-6' YEAR TO MONTH
INTERVAL ' + 2 - 6 ' YEAR TO MONTH
INTERVAL ' - 2 - 6 ' YEAR TO MONTH
```

Literal INTERVAL Tag-Zeit

```
INTERVAL ' 2' DAY
INTERVAL '+2' DAY
INTERVAL '-2'  DAY
INTERVAL ' 12' HOUR
INTERVAL '+12' HOUR
INTERVAL '-12' HOUR
INTERVAL ' 29' MINUTE
INTERVAL '+29' MINUTE
INTERVAL '-29' MINUTE
INTERVAL ' 35.99' SECOND
INTERVAL '+35.99' SECOND
INTERVAL '-35.99' SECOND
INTERVAL ' 2 12:29:35.99'  DAY TO SECOND
INTERVAL '+2 12:29:35.99'  DAY TO SECOND
INTERVAL '-2 12:29:35.99'  DAY TO SECOND
INTERVAL ' 2 12:29'        DAY TO MINUTE
INTERVAL '+2 12:29'        DAY TO MINUTE
INTERVAL '-2 12:29'        DAY TO MINUTE
INTERVAL ' 2 12'           DAY TO HOUR
INTERVAL '+2 12'           DAY TO HOUR
INTERVAL '-2 12'           DAY TO HOUR
INTERVAL   ' 12:29'        HOUR TO MINUTE
INTERVAL   '+12:29'        HOUR TO MINUTE
INTERVAL   '-12:29'        HOUR TO MINUTE
INTERVAL   ' 12:29:35.99'  HOUR TO SECOND
INTERVAL   '+12:29:35.99'  HOUR TO SECOND
INTERVAL   '-12:29:35.99'  HOUR TO SECOND
INTERVAL      ' 29:35.99'  MINUTE TO SECOND
INTERVAL      '+29:35.99'  MINUTE TO SECOND
INTERVAL      '-29:35.99'  MINUTE TO SECOND
```

11.9.4 Operatoren und Funktionen

Datenumwandlung CAST (scalar-expression AS (data-type | domain))

Das aktuelle Datum, die aktuelle Zeit, den aktuellen Zeitstempel liefern:

```
CURRENT_DATE
CURRENT_TIME
CURRENT_TIMESTAMP
```

Extraktionsfunktion

```
EXTRACT(field FROM scalar-expression)
```

Aggregatfunktionen

```
COUNT MAX MIN bei DATE TIME TIMESTAMP
COUNT MAX MIN SUM AVG bei INTERVAL
```

arithmetische Operationen (datetime bedeutet: DATE TIME TIMESTAMP)

```
datetime – datetime = interval
datetime + interval = datetime
datetime – interval = datetime
interval + datetime = datetime
interval + interval = interval
interval – interval = interval
interval * number   = interval
interval / number   = interval
number   * interval = interval
```

```
DATE '1998-08-31'+ INTERVAL '1' MONTH
invalid date exception
```

Vergleichsbedingung OVERLAPS für Zeiträume

```
(TIME '08:00:00', TIME '09:00:00')
OVERLAPS
(TIME '09:00:00', TIME '09:30:00')
evaluates to false
```

11.9.5 Function EXTRACT

```
select
 autornr
,geburtsdatum
,EXTRACT( MONTH FROM geburtsdatum) as monat
,EXTRACT( YEAR  FROM geburtsdatum) as jahr
FROM tautor
;
```

Zitat Anfang

Oracle® Database SQL Reference 10g Release 2 (10.2)

```
EXTRACT( { { YEAR
           | MONTH
           | DAY
           | HOUR
           | MINUTE
           | SECOND
           }
         | { TIMEZONE_HOUR
           | TIMEZONE_MINUTE
           }
         | { TIMEZONE_REGION
           | TIMEZONE_ABBR
           }
         }
         FROM { datetime_value_expression
              | interval_value_expression
              }
       )
```

Purpose

EXTRACT extracts and returns the value of a specified datetime field from a datetime or interval value expression.

This function can be very useful for manipulating datetime field values in very large tables.

Zitat Ende

EXTRACT extrahiert aus CURRENT_TIMESTAMP bzw. dem Datentyp TIMESTAMP WITH TIME ZONE die Coordinated Universal Time UTC.

```
SELECT  'treffer'  as abcd
FROM DUAL
WHERE
EXTRACT (HOUR FROM
TIMESTAMP '2008-09-02 15:29:41.828000 +00:00')
=
EXTRACT (HOUR FROM
TIMESTAMP '2008-09-02 17:29:41.828000 +02:00')
;
-- 17 Uhr in Berlin entspricht 15 Uhr in London --
```

11.9.6 Datum und CREATE TABLE

```
--Test mit Oracle
alter session set NLS_DATE_FORMAT
= 'YYYY-MM-DD HH24:MI:SS';
ALTER SESSION SET NLS_TIMESTAMP_FORMAT
= 'YYYY-MM-DD HH24:MI:SS.FF';
alter session set NLS_TIMESTAMP_TZ_FORMAT
= 'YYYY-MM-DD HH24:MI:SS.FF TZH:TZM';
drop table tergebnisse
;
CREATE TABLE tergebnisse
(  schluessel      NUMBER not null
  ,datum           DATE
  ,beginn          TIMESTAMP(3)
  ,beginnwithtz    TIMESTAMP(3) WITH TIME ZONE
  ,dauergrob       INTERVAL YEAR TO MONTH
  ,dauerfein       INTERVAL DAY (6) TO SECOND (5)
  ,primary key (schluessel)
);
delete from tergebnisse
;
insert into tergebnisse
(schluessel, datum, beginn, beginnwithtz, dauergrob, dauerfein)
values
(111
,DATE      '0001-01-01'
,TIMESTAMP '0001-01-01 00:00:00.000000'
,TIMESTAMP '0001-01-01 00:00:00.000000 +02:00'
,INTERVAL  '4-11' YEAR TO MONTH
,INTERVAL  '003 11:22:33.1234567890' DAY TO SECOND
)
;
insert into tergebnisse
(schluessel, datum, beginn, beginnwithtz, dauergrob, dauerfein)
values
(222
,current_date
,current_timestamp
,current_timestamp
,CAST ('4-11' AS INTERVAL YEAR TO MONTH)
,CAST ('003 11:22:33.1234567890' AS INTERVAL DAY TO SECOND)
)
;
SELECT
    schluessel                  AS schluessel
  , datum                       as datum
  , beginn                      as beginn
  , beginnwithtz                as beginnwithtz
  , dauergrob                   as dauergrob
  , dauerfein                   as dauerfein
  , beginn  + dauergrob + dauerfein  as ende
FROM tergebnisse
;
```

--FRAGE

--Warum liefern die folgenden Abfragen verschiedene Listen?

```
alter session set NLS_DATE_FORMAT
= 'YYYY-MM-DD'
;
Select current_date from DUAL
;
update tergebnisse
set datum = current_date
;
select schluessel, datum, CURRENT_DATE
from tergebnisse
;
SCHLUESSEL           DATUM                     CURRENT_DATE
-------------------- ------------------------- --------------
111                  2011-07-25                2011-07-25
222                  2011-07-25                2011-07-25

2 rows selected

select schluessel, datum, current_DATE
from tergebnisse
where datum = CURRENT_DATE
;
SCHLUESSEL           DATUM                     CURRENT_DATE
-------------------- ------------------------- --------------

0 rows selected
```

--Antwort

--DATE von Oracle ist etwas anders als YYYY-MM-DD

--CURRENT_DATE von Oracle ist etwas anderes als YYYY-MM-DD

11.9.7 INTERVAL und Arithmetik, EXTRACT

```
alter session set nls_date_format = 'yyyy-mm-dd'
;
select
 cast('2012-01-31' AS DATE) + interval '2' month as dat1
,cast('2012-01-31' AS DATE) + interval '1-2' year to month as dat2
from dual
;
DAT1                    DAT2
----------------------- -----------------------
2012-03-31              2013-03-31

select
 cast  ('2012-01-31' AS DATE) + interval '1' month          as dat1
,cast  ('2012-01-31' AS DATE) + interval '1-2' year to month as dat2
from dual
;
Error report:
SQL Error: ORA-01839: Datum für angegebenen Monat nicht gültig
01839. 00000 -  "date not valid for month specified"
*Cause:
*Action:
```

Die Beispiel zeigt, dass Datumsaritmethik mit INTERVAL MONTH bzw. INTERVAL YEAR TO MONTH nur mit Einschränkungen verwendet werden kann.

```
alter session set nls_date_format = 'YYYY-MM-DD HH24:MI:SS';

--     1.1. bis 15.2.  >>> 1 Monat
--bzw. 1.1. bis 16.2.  >>> 2 Monate
select
  (
    cast ('2008-02-15 16:50:39' as date)
    -
    cast ('2005-01-01 10:20:39' as date)
  )
    YEAR TO  MONTH
AS jjjjtomm_1
,
  (
    cast ('2008-02-16 16:50:39' as date)
    -
    cast ('2005-01-01 10:20:39' as date)
  )
    YEAR TO  MONTH
AS jjjjtomm_2
from dual
;
JJJJTOMM_1  JJJJTOMM_2
----------- -----------
3-1         3-2

with
tabdauer as
(
select
 (
    cast ('2008-02-16 16:50:39' as date)
    -
    cast ('2005-01-01 10:20:39' as date)
  )
    YEAR TO  MONTH  as dauer
from dual
)
SELECT
tabdauer.dauer                        as dauer
,EXTRACT (YEAR FROM tabdauer.dauer)  AS jahre
,EXTRACT (MONTH FROM tabdauer.dauer) AS monate
from tabdauer
;
DAUER       JAHRE                    MONATE
----------- ------------------------ ----------------------
3-2         3                        2
```

```
alter session set nls_date_format = 'YYYY-MM-DD HH24:MI:SS'
;
Select
  (
    cast ('2008-02-15 16:50:39' as date)
    -
    cast ('2005-01-01 10:20:39' as date)
  )
   DAY(7) TO SECOND(3)
AS tagzusekunden
FROM DUAL
;
TAGZUSEKUNDEN
-------------
+0001140 06:30:00.000

alter session set nls_date_format = 'YYYY-MM-DD HH24:MI:SS'
;
with
tabdauer as
(
select
 (
    cast ('2008-02-15 16:50:39' as date)
    -
    cast ('2005-01-01 10:20:39' as date)
  )
    DAY(7) TO  SECOND(3)  as dauer
from dual
)
SELECT
 tabdauer.dauer                          as dauer
,EXTRACT (DAY FROM tabdauer.dauer)       AS tage
,EXTRACT (HOUR FROM tabdauer.dauer)      AS stunden
,EXTRACT (MINUTE FROM tabdauer.dauer)    AS minuten
--,EXTRACT (SECOND FROM tabdauer.dauer) AS sekunden
from tabdauer
;
DAUER                       TAGE      STUNDEN   MINUTEN
------------------------- --------  --------- --------
+0001140 06:30:00.00.000   1140       6         30
```

```
alter session set nls_date_format = 'YYYY-MM-DD HH24:MI:SS';

--6 stunden = 0,25 tage
--30 minuten = 1/48 tage = 0,020833 tage

select
    cast ('2008-02-15 16:50:39' as date)
    -
    cast ('2005-01-01 10:20:39' as date)
    as interval_in_tagen
from dual
;
INTERVAL_IN_TAGEN
---------------------
1140,27083333333333333


with tabcloope as
(
select
 cast ('0001-01-01 00:00:00' as date) as von
,cast ('9999-12-30 23:59:59' as date) as bisclosed
,cast ('9999-12-31 00:00:00' as date) as bisopen
from dual
)
select
 von
,bisclosed                 as bisclosed1
,bisopen - (1/(24*60*60))  as bisclosed2
,bisopen
,bisopen-von                    as anztage1
--,(bisopen-von) DAY TO SECOND   as anztage2
from tabcloope
;
```

11.9.8 Temporale Datenbanken SQL:2011, Implementierung DB2

CREATE TABLE mit der Semantik [closed,open)
von < bisopen mit Wertebereich [1.1.0001,31.12.9999)
der “letzte Tag” ist 9999-12-30

```
drop table tpreiscloope;
commit;
create table tpreiscloope
(
  buchnr  INTEGER    not null
 ,preis   DECIMAL(7,2)
 ,von     DATE       not null
 ,bisopen DATE       not null
 ,UNIQUE(buchnr, von)
 ,UNIQUE(buchnr, bisopen)
 ,CONSTRAINT con1_tpreiscloope CHECK (von < bisopen)
 ,CONSTRAINT con2_preiscloope CHECK
   (
       CAST('0001-01-01' AS DATE)<=von
       AND bisopen <= CAST('9999-12-31' AS DATE)
   )
)
;
```

Integritätsbedingungen

- Intervalle pro buchnr überlappen sich nicht. Overlaps bei gleicher buchnr ist nicht möglich.
- Intervalle für buchnr, preis sind maximal. Meets bei gleicher buchnr,preis ist nicht möglich.

Mit dieser Datendefinition sind die Integritätsbedingungen für Meets und Overlaps nicht implementiert.

```
Delete from tpreiscloope;
--overlaps bei gleicher Buchnr
INSERT INTO tpreiscloope(buchnr, von, bisopen, preis)
VALUES  (1 , '0001-01-01', '0001-01-05', 11.11);
INSERT INTO tpreiscloope(buchnr, von, bisopen, preis)
VALUES  (1 , '0001-01-03','0001-01-07', 22.22);
--meets mit gleicher Buchnr, gleichem Preis
INSERT INTO tpreiscloope(buchnr, von, bisopen, preis)
VALUES  (1 , '2011-01-01', '2012-01-01', 44.44);
INSERT INTO tpreiscloope(buchnr, von, bisopen, preis)
VALUES  (1 , '2012-01-01', '9999-12-31', 44.44);
--
INSERT INTO tpreiscloope(buchnr, von, bisopen)
VALUES  (2 , '0001-01-01', '9999-12-31');

select buchnr, von, bisopen, preis
from tpreiscloope
order by buchnr, von
;
```

buchnr	von	bisopen	preis
1	**0001-01-01**	**0001-01-05**	**11.11**
1	**0001-01-03**	**0001-01-07**	**22.22**
1	2011-01-01	2012-01-01	44.44
1	2012-01-01	9999-12-31	44.44
2	0001-01-01	9999-12-31	NULL

CREATE TABLE mit **PERIOD BUSINESS_TIME** der Semantik [closed,open)
von < bisopen mit Wertebereich [1.1.0001,31.12.9999)
der “letzte Tag” ist 9999-12-30

```
DROP TABLE   tpreiscloopedb2;
commit;
CREATE TABLE   tpreiscloopedb2
 (
   buchnr              integer NOT NULL
  ,preis               decimal(7,2)
  ,von                 DATE NOT NULL
  ,bisopen             DATE NOT NULL
  ,PERIOD BUSINESS_TIME(von, bisopen)
  ,PRIMARY KEY(buchnr, BUSINESS_TIME WITHOUT OVERLAPS)
  ,CONSTRAINT con2_preiscloope CHECK
   (
       CAST('0001-01-01' AS DATE)<=von
       AND bisopen <= CAST('9999-12-31' AS DATE)
   )
)
;
commit
;
```

--Mit dieser Datendefinition ist die Integritätsbedingungen für Overlaps implementiert.

```
Delete from tpreiscloopedb2;
--overlaps bei gleicher Buchnr
INSERT INTO tpreiscloopedb2(buchnr, von, bisopen, preis)
VALUES  (1 , '0001-01-01', '0001-01-05', 11.11);
INSERT INTO tpreiscloope(buchnr, von, bisopen, preis)
VALUES  (1 , '0001-01-03','0001-01-07', 22.22);
--meets mit gleicher Buchnr, gleichem Preis
INSERT INTO tpreiscloopedb2(buchnr, von, bisopen, preis)
VALUES  (1 , '2011-01-01', '2012-01-01', 44.44);
INSERT INTO tpreiscloopedb2(buchnr, von, bisopen, preis)
VALUES  (1 , '2012-01-01', '9999-12-31', 44.44);
--
INSERT INTO tpreiscloopedb2(buchnr, von, bisopen)
VALUES  (2 , '0001-01-01', '9999-12-31');

select buchnr, von, bisopen, preis
from tpreiscloopedb2
order by buchnr, von
;
```

buchnr	von	bisopen	preis
1	**0001-01-01**	**0001-01-05**	**11.11**
1	2011-01-01	2012-01-01	44.44
1	2012-01-01	9999-12-31	44.44
2	0001-01-01	9999-12-31	NULL

```
--Insert
--Achtung: dieser Insert ist nicht möglich, Verstoss gegen eine
Integritätsbedingung: overlaps nicht möglich
INSERT INTO tpreiscloope(buchnr, von, bisopen, preis)
VALUES  (1 , '0001-01-03','0001-01-07', 22.22);

--Update
--Achtung: nach Update Meets von Intervallen mit gleichem Preis
ist möglich, durch Update kann die Anzahl der Zeilen grösser
werden
select buchnr, von, bisopen, preis
from tpreiscloopedb2
order by buchnr, von
;
UPDATE   tpreiscloopedb2
  FOR PORTION OF BUSINESS_TIME
  FROM '0001-01-03' TO '0001-01-04'
SET preis =  11.11
WHERE buchnr = 1
;
select buchnr, von, bisopen, preis
from tpreiscloopedb2
order by buchnr, von
;
--Update
--Achtung: dieser Update liefert „keine Zeile gefunden"
UPDATE   tpreiscloopedb2
  FOR PORTION OF BUSINESS_TIME
  FROM '0001-01-01' TO '0001-01-01'
SET preis =  99.99
WHERE buchnr = 1
;
--Delete
--Achtung: durch Delete kann die Anzahl der Zeilen grösser wer-
den
select buchnr, von, bisopen, preis
from tpreiscloopedb2
order by buchnr, von
;
DELETE FROM   tpreiscloopedb2
  FOR PORTION OF BUSINESS_TIME
  FROM '0002-01-01' TO '0003-01-01'
WHERE buchnr = 2
;
select buchnr, von, bisopen, preis
from tpreiscloopedb2
order by buchnr, von
;
```

```
--Zurücksetzen der Testdaten
Delete from tpreiscloopedb2;
--overlaps bei gleicher Buchnr
INSERT INTO tpreiscloopedb2(buchnr, von, bisopen, preis)
VALUES  (1 , '0001-01-01', '0001-01-05', 11.11);
INSERT INTO tpreiscloope(buchnr, von, bisopen, preis)
VALUES  (1 , '0001-01-03','0001-01-07', 22.22);
--meets mit gleicher Buchnr, gleichem Preis
INSERT INTO tpreiscloopedb2(buchnr, von, bisopen, preis)
VALUES  (1 , '2011-01-01', '2012-01-01', 44.44);
INSERT INTO tpreiscloopedb2(buchnr, von, bisopen, preis)
VALUES  (1 , '2012-01-01', '9999-12-31', 44.44);
--
INSERT INTO tpreiscloopedb2(buchnr, von, bisopen)
VALUES  (2 , '0001-01-01', '9999-12-31');

select buchnr, von, bisopen, preis
from tpreiscloopedb2
order by buchnr, von
;
SELECT COUNT(*) as countstern
FROM   tpreiscloopedb2
;
--Select
--der Preis eines Buches an einem bestimmten Tag
SELECT buchnr, von, bisopen, preis
FROM   tpreiscloopedb2
FOR BUSINESS_TIME AS OF '2012-01-01'
;
--Achtung: dieser select liefert auf Grund der [closed,open)
Semantik nie eine Zeile
SELECT buchnr, von, bisopen, preis
FROM   tpreiscloopedb2
FOR BUSINESS_TIME AS OF '9999-12-31'
;
--Achtung: dieser select liefert aber Zeilen, das Ergebnis ist
nicht korrekt, da die Datenbank für diesen Tag keinen Preis
präsentiert, es ist ein Fehler des Anwenders
SELECT buchnr, von, bisopen, preis
FROM   tpreiscloopedb2
WHERE bisopen = '9999-12-31'
;
--der Preis eines Buches in überlappenden Intervallen
SELECT buchnr, von, bisopen, preis
FROM tpreiscloopedb2
FOR BUSINESS_TIME FROM '2011-07-01' TO '2012-07-01'
;
```

11.10 Datums- und Zeitangaben, ein Vergleich SQL Standard – DB2 – ORACLE – SQL Server

Beachten Sie bitte:

Bei diesem Vergleich werden syntaktische Möglichkeiten aufgelistet und gegenübergestellt.
Die unterschiedliche Semantik der Operatoren und Funktionen der Datumsarithmetik wird hier nicht im Detail beschrieben.

Beispiele für semantische Unterschiede:

- ***Oracle ADD_MONTHS** ist etwas anders als DB2 +…MONTHS bzw. SQL Server DATEADD(MONTH…)*
- ***SQL Server DATEDIFF(MONTH,d1,d2)** ist etwas anderes als DB2 TIMESTAMPDIFF bzw. Oracle MONTHS_BETWEEN*

Oracle ADD_MONTHS(date, integer)
returns the date date plus integer months. The date argument can be a datetime value or any value that can be implicitly converted to DATE. The integer argument can be an integer or any value that can be implicitly converted to an integer. The return type is always DATE, regardless of the datatype of date. **If date is the last day of the month or if the resulting month has fewer days than the day component of date, then the result is the last day of the resulting month.** Otherwise, the result has the same day component as date.

SQL Server DATEDIFF (datepart ,startdate ,enddate)
gibt die Anzahl der Datums- oder Zeitbegrenzungen von datepart zurück, die zwischen zwei angegebenen Daten überschritten wurden.

```
with zwi as
(select  cast('1999-12-31' as date) as von
        ,cast('2000-01-01' as date) as bisopen
)
select
 von
,bisopen
,DATEDIFF(YYYY, von,bisopen) as jahre  --präsentiert 1 Jahr
,DATEDIFF(MM  ,von,bisopen)  as Monate --präsentiert 1 Monat
,DATEDIFF(DD  ,von,bisopen)  as Tage   --präsentiert 1 Tag
from zwi
```

In der SQL Standard-Spalte:
x.y ist ein numerischer Wert (mit oder ohne Dezimalstellen)
d ist ein Wert vom Typ DATE, TIME bzw. TIMESTAMP
i ist ein Wert vom Typ INTERVAL

In der DB2-Spalte:
x.y ist ein numerischer Wert (mit oder ohne Dezimalstellen)
d ist ein Wert vom Typ DATE, TIME, TIMESTAMP
i bedeutet eine Date Duration, Time Duration bzw. Timestamp Duration

In der ORACLE-Spalte:
x.y ist ein numerischer Wert (mit oder ohne Dezimalstellen)
d ist ein Wert vom Typ ***DATE***, TIMESTAMP
i ist ein Wert vom Typ INTERVAL
jd ist vom Typ NUMBER, und repräsentiert einen Julianischen Tag

In der SQL Server-Spalte:
x.y ist ein numerischer Wert (mit oder ohne Dezimalstellen)
d ist ein Wert vom Typ DATE, TIME**, *DATETIME2, DATETIME***
is ist numerischer Wert (mit oder ohne Dezimalstellen) und repräsentiert eine Anzahl von Jahren bzw. Monaten bzw. Tagen etc.

11.10.1 Types

SQL Standard	DB2	ORACLE	SQL Server
Types			
DATE	DATE	***DATE*** ignoring the hour, minute, and second fields	DATE SQL Server 2008 bzw. ***DATETIME*** bzw. **SMALLDATETIME** ignoring the hour, minute, and second fields
TIME	TIME	***DATE*** ignoring the century, year, month, and day fields	TIME SQL Server 2008 bzw. ***DATETIME*** bzw. **SMALLDATETIME** ignoring the century, year, month, and day fields
TIMESTAMP	TIMESTAMP	TIMESTAMP ***DATE*** (to second granularity)	**DATETIME2** SQL Server 2008 bzw. ***DATETIME*** bzw. **SMALLDATETIME** (to second granularity)
TIME WITH TIME ZONE	no equivalent	no equivalent	no equivalent
TIMESTAMP WITH TIME ZONE	TIMESTAMP WITH TIME ZONE DB2 for z/OS Version 10	TIMESTAMP WITH TIME ZONE	**DATETIMEOFFSET** SQL Server 2008
INTERVAL YEAR TO MONTH	a date duration (DECIMAL(8,0)) with a 0 DAY field	INTERVAL YEAR TO MONTH	int (integral number of months)
INTERVAL DAY TO SECOND	a timestamp duration with 0 YEAR, MONTH, and MICROSECOND fields	INTERVAL DAY TO SECOND	int (integral number of seconds)
	Date Duration DECIMAL(8,0) **YYYYMMDD.** Interpretierte Dezimalzahl		
	Time Duration DECIMAL(6,0) **hhmmss.** Interpretierte Dezimalzahl		
	Timestamp Duration DECIMAL(20,6) **YYYYMMD-Dhhmmss.xxxxxx** Interpretierte Dezimalzahl DB2 for z/OS Version 10 DECIMAL(14+s,s) wobei 0<= s<=12		

11.10.2 Literals

SQL Standard	DB2	ORACLE	SQL Server
Literals			
DATE ‘1997-01-31’	CAST ('1997-01-31' AS DATE) bzw. DATE (‘1997-01-31’)	CAST ('1997-01-31' AS DATE) bzw. TO_DATE (‘1997-01-31’, 'YYYY-MM-DD')	CAST ('1997-01-31' AS DATE) SQL Server 2008 bzw. CONVERT (DATE, '1997-01-31') bzw. '1997-01-31'
TIME '12:34:56'	CAST ('12:34:56' AS TIME) bzw. TIME ('12:34:56')	TO_DATE ('12:34:56', 'HH24:MI:SS') ignoring the century, year, month, and day fields	CAST ('12:34:56' AS TIME) SQL Server 2008 bzw. CONVERT (TIME, '12:34:56') bzw. '12:34:56'
TIMESTAMP '1997-01-31 12:34:56'	CAST('1997-01-31 12:34:56' AS TIMESTAMP) bzw. TIMESTAMP ('1997-01-31 12:34:56')	CAST('1997-01-31 12:34:56' AS TIMESTAMP) bzw. TO_DATE ('1997-01-31 12:34:56', 'YYYY-MM-DD HH24:MI:SS')	CAST('1997-01-31 12:34:56' AS **DATETIME2**) SQL Server 2008 bzw. set dateformat ymd CONVERT (DATETIME,'1997-01-31 12:34:56') bzw. '1997-01-31 12:34:56'
INTERVAL '3-4' YEAR TO MONTH	40 MONTHS bzw. 00030400. (only in an expression)	INTERVAL '3-4' YEAR TO MONTH	40 (months)
INTERVAL '1 23:45:12' DAY TO SECOND	171912 SECONDS bzw. 00000001234512.000000. (only in an expression)	INTERVAL '1 23:45:12' DAY TO SECOND xxxx)	171912 (seconds)
TIME WITH TIME ZONE			
TIMESTAMP WITH TIME ZONE TIMESTAMP '0001-01-01 **02**:00:00.000000 +**02**:00'		TIMESTAMP '0001-01-01 **02**:00:00.000000 +**02**:00' TIMESTAMP '0001-01-01 **00**:00:00.000000 +**02**:00'	TIMESTAMP '0001-01-01 **02**:00:00.000000 +**02**:00'

SQL Standard	DB2	ORACLE	SQL Server
Literals			
	Date Duration DECIMAL(8,0) **00020433.** 2 years 4 months 33 days Interpretierte Dezimalzahl bzw. 2 YEARS + 4 MONTHS + 33 DAYS		
	Time Duration DECIMAL(6,0) **020433.** 2 hours 4 minutes 33 seconds Interpretierte Dezimalzahl bzw. 2 HOURS + 4 MINUTES + 33 SECONDS		
	Timestamp Duration DECIMAL(20,6) **00020433020433.993456** 2 years 4 months 33 days 2 hours 4 minutes 33.993456 seconds Interpretierte Dezimalzahl bzw. 2 YEARS + 4 MONTHS + 33 DAYS + 2 HOURS + 4 MINUTES + 00000000000033.993456 DB2 for z/OS Version 10 DECIMAL(14+s,s) wobei 0<= s<=12		

xxxx)

```
alter session set nls_date_format = 'YYYY-MM-DD HH24:MI:SS';
alter session set nls_date_format = 'YYYY-MM-DD';
WITH zwi AS
(
 SELECT
 cast('2000-01-01' as date)                    as von
,cast ('1 23:45:12' as interval day to second ) as dtos_interval
 FROM DUAL
)
select
 von                          as von
,dtos_interval                as dtos_interval
,(von + dtos_interval) - von as tage
from zwi
;
```

11.10.3 Predicates

SQL Standard	DB2	ORACLE	SQL Server
Predicates			
d1= d2	d1= d2	d1= d2	d1= d2
d1< d2	d1< d2	d1< d2	d1< d2
d1<> d2	d1<> d2	d1<> d2	d1<> d2
d1 BETWEEN d2 AND d3	d1 BETWEEN d2 AND d3	d1 BETWEEN d2 AND d3	d1 BETWEEN d2 AND d3
i1= i2	i1= i2	i1= i2 jd1=jd2	 is1=is2
i1< i2	***)	i1< i2 jd1<jd2	 is1<is2
i1<> i2	***)	i1<> i2 jd1<>jd2	 is1<>is2
i1 BETWEEN i2 AND i3	***)	i1 BETWEEN i2 AND i3 jd1 BETWEEN jd2 AND jd3	 is1 BETWEEN is2 AND is3
d IS NULL	d IS NULL	d IS NULL	d IS NULL
i IS NULL	i IS NULL	i IS NULL jd IS NULL	 is IS NULL
(d1, d2) OVERLAPS (d3, d4)			
(d1, i) OVERLAPS (d3, d4)			

SQL Predicate OVERLAPS

SQL stellt einen speziellen Vergleichsoperator bereit, um zu testen ob zwei Zeitperioden sich überlappen. Beispiel für die Syntax

```
(TIME '08:00:00', TIME '09:00:00')
OVERLAPS
(TIME '08:30:00', TIME '09:30:00')
evaluates to true

 (TIME '08:00:00', TIME '09:00:00')
OVERLAPS
(TIME '09:00:00', TIME '09:30:00')
evaluates to false

(date '2007-01-01', date '2008-01-01')
OVERLAPS
(date '2005-01-01', date '2007-01-02')
evaluates to true

(date '2007-01-01', date '2008-01-01')
OVERLAPS
(date '2005-01-01', date '2007-01-01')
evaluates to false
```

OVERLAPS ist implementiert für die Semantik [closed, open) und ist für [i1.von, i1.bis) und [i2.von, i2.bis) semantisch äquivalent mit folgendem Bedingungsausdruck:

```
(i1.von > i2.von AND (i1.von < i2.bis OR i1.bis < i2.bis))
OR
(i2.von > i1.von AND (i2.von < i1.bis OR i2.bis < i1.bis))
OR
(i1.von = i2.von AND i1.bis IS NOT NULL and i2.bis IS NOT NULL)
```

Beispiel Oracle i1<i2

```
INTERVAL 11 Monate sind weniger als 14 Monate
select dauer1, dauer2
from
(
select
 interval '0-11' year to month as dauer1
,interval '1-3' year to month as dauer2
from dual
)zwi where dauer1<dauer2
;
```

Beispiel DB2 ***) Duration ist eine interpretierte Dezimalzahl

```
Date Duration 11 Monate sind weniger als 14 Monate
Select dauer1, dauer2
from
(
select
+00001100. as dauer1 ---11 monate
,00010300. as dauer2 ---1 jahr und 3monate
from sysibm.sysdummy1
)zwi where dauer1<dauer2
;
***) Achtung:
Date Duration
in der Dezimalzahl der Date Duration
sind für den Monatsteil auch Werte >=12 möglich,
dann gilt beim folgenden Beispiel
   dauer2=15 monate sind weniger als dauer1=24monate
aber der Select liefert natürlich
select dauer1, dauer2
from
(
select
+00002400. as dauer1 ---24 monate
,00010300. as dauer2 ---1 jahr und 3monate
from sysibm.sysdummy1
)zwi where dauer1<dauer2
;
***) Achtung:
Time Duration
in der Dezimalzahl der Time Duration
sind für den Sekundenanteil auch Werte >=60 möglich,
Seien dur99 = 000099. und dur100= 000100. Dezimalzahlen,
sei Uhrzeit eine Spalte vom Datentyp TIME,
dann gelten folgende Ungleichungen:

          dur99 < dur100
Uhrzeit + dur99 > dur100 + Uhrzeit

da 99 Sekunden mehr ist als eine Minute!

***) Achtung:
Timestamp Duration
```

11.10.4 Datetime Constructors

***) Beachten Sie bitte die besondere Semantik von Oracle ADD_MONTHS.

SQL Standard	DB2	ORACLE	SQL Server
Datetime Constructors			
d + i	d + i d + x.y YEARS **d + x.y MONTHS** … d + x.y SECONDS …	d + i d + jd d + x.y *****)** ~~**ADD_MONTHS(d, x.y)**~~	 DATEADD(YEAR , x.y, d) **DATEADD(MONTH, x.y, d)** … DATEADD(SECOND,x.y, d) …
i + d	i + d x.y YEARS + d **x.y MONTHS +d** … x.y SECONDS + d …	i + d jd + d x.y + d *****)** ~~**ADD_MONTHS(d, x.y)**~~	 DATEADD(YEAR , x.y, d) **DATEADD(MONTH, x.y, d)** … DATEADD(SECOND,x.y, d) …
d – i	d – i d – x.y YEARS **d – x.y MONTH**S … d – x.y SECONDS …	d – i d – jd d – x.y *****)** ~~**ADD_MONTHS(d, -x.y)**~~	 DATEADD(YEAR ,-x.y, d) **DATEADD(MONTH,-x.y, d)** … DATEADD(SECOND,-x.y,d) …
CURRENT_DATE	CURRENT_DATE	TRUNC(CURRENT_DATE) TRUNC (CURRENT_DATE,'DD') TRUNC(CURRENT_TIMESTAMP) TRUNC(CURRENT_TIMESTAMP,'DD')	 CAST(GETDATE() AS DATE) CAST(CURRENT_TIMESTAMP AS DATE)
CURRENT_TIME	CURRENT_TIME	TO_CHAR (CURRENT_DATE, 'HH24:MI:SS') ist eine Zeichen-kette	CAST(CURRENT_TIMESTAMP AS TIME)
	current_time - current_timezone		
CURRENT_TIMESTAMP	CURRENT_TIMESTAMP	CURRENT_TIMESTAMP	CURRENT_TIMESTAMP GETDATE()

Wann werden die Nachkommastellen ignoriert, wann wird gerundet?

```
--DB2
   with abc as(
   select
   CAST( '0001-01-01 00:00:00.000'  AS timestamp(3) ) as von
   from sysibm.sysdummy1
   )
   select
   von as von
   --,von + 1.9999999 years
   --,von + 1.9999999 months
   --,von + 1.9999999 days
   --,von + 1.9999999 hours
   --,von + 1.9999999 minutes
   --,von + 1.9999999 seconds
   ----,von + 1.9999999 milliseconds
   ,von + 1999.9999999 microseconds
   from abc
   ;

--ORACLE
   ALTER SESSION SET
   NLS_TIMESTAMP_FORMAT = 'YYYY-MM-DD HH24:MI:SS.FF';
   with abc as(
   select
   CAST( '0001-01-01 00:00:00.000'  AS timestamp(3) ) as von
   from dual
   )
   select
   von as von
   ,von + 7.9999999999             --tage
   ,ADD_MONTHS(von, 2.9999999999)  --monate
   --,von + INTERVAL '01' year
   ----,von + INTERVAL  '1.9999999' year
   --,von + INTERVAL '01' month
   --,von + INTERVAL '01' day
   --,von + INTERVAL '01' hour
   --,von + INTERVAL '01' minute
   --,von + INTERVAL '01' second
   --achtung
   --,von + INTERVAL '00.9999' second(3)
   --
   --achtung es wird aufgerundet 1 Sekunde
   --,von + INTERVAL '00.9999999' second(3)
   --,von + INTERVAL '01.999999' second(6)
   --achtung es wird aufgerundet
   --,von + INTERVAL '01.99999999' second(6)
   --,von + INTERVAL '00.000001' second(6)
   from abc
   ;
--SQL Server
   with abc as(
   select
   CAST( '0001-01-01 00:00:00.000' as DATETIME2(7)) as von
   )
   select
    von as von
   ,DATEADD (YEAR,  1.999999 , von)
   ,DATEADD (MONTH, 1.999999 , von)
   ,DATEADD (DAY,   1.999999 , von)
   ,DATEADD (HOUR,  1.999999 , von)
   ,DATEADD (MINUTE, 1.999999 , von)
   ,DATEADD (SECOND, 1.999999 , von)
   ,DATEADD (MILLISECOND, 1.999999 , von)
   ,DATEADD (MICROSECOND, 1.999999 , von)
   --achtung es wird aufgerundet
   ,DATEADD (NANOSECOND,  51.1111 , von)
   from abc
   ;
```

11.10.5 Interval Constructors

***) Beachten Sie bitte die besondere Semantik von SQL Server DATEDIFF(MONTH,d1,d2).

SQL Standard	**DB2**	**ORACLE**	**SQL Server**
Interval Constructors			
i1 + i2	not possible	i1 + i2 jd1 + jd2	is1 + is2
i1 – i2	not possible	i1 – i2 jd1 – jd2	is1 – is2
i * n	not possible	i * n jd * n	is * n
n * i	not possible	n * i n * jd	n * is
		jd1 / jd2	is1 / is2
i / n	not possible	i / n jd / n	is / n
+ i	+ i	+ i + jd	+ is
- i	- i	**-1* i** - jd	- is

(d2 – d1) DAY TO SECOND	DAYS(d2) – DAYS(d1)	d2 – d1 Datentyp DATE (result is a fractional number of days) TRUNC(d2 – d1 , 0) tstp2 – tstp1 Datentyp TIMESTAMP(0) (result is …)	DATEDIFF(**dd**, d1, d2)
~~(d2 – d1) YEAR TO MONTH~~	TIMESTAMPDIFF (**64**, CHAR(d2 – d1))	MONTHS_BETWEEN (d2, d1) (result is a (fractional) number of months) TRUNC(MONTHS_BETWEEN (d2, d1) , 0)	***) ~~**DATEDIFF(MONTH, d1, d2)**~~

SQL Standard *)**		**ORACLE ***)**	
(d2-d1) YEAR TO MONTH (d2-d1) DAY TO SECOND		(d2-d1) YEAR(9) TO MONTH es wird gerundet am 16.des Monats am 16.12.des Jahres (d2-d1) DAY(9) TO SECOND(9) es wird gerundet bei SECOND(x) Wenn x<9	

	DB2 *)**		
	d2 – d1 DATE Duration TIME Duration TIMESTAMP Duration		

SQL Standard *)**	**DB2 ***)**	**ORACLE ***)**	**SQL Server ***)**
(d2 – d1) qual mögliche Werte (Oracle): year to month day to second	TIMESTAMPDIFF (**itype**, CHAR(d2 – d1)) **itype** ist eine Ganzzahl und bedeutet einen Intervaltyp mögliche Werte: 1 Bruchteile von Sekunden, 2 Sekunden, 4 Minuten, 8 Stunden, 16 Tage, 32 Wochen, 64 Monate, 128 Vierteljahre, 256 Jahre Returns an estimated number of intervals of the type defined by the first argument, based on the difference between two timestamps.		DATEDIFF(qual, d1, d2) (result is integral number at the indicated granularity) mögliche Werte: year, yy, yyyy quarter, qq, q month, mm, m dayofyear, dy, y day, dd, d week, wk, ww hour, hh minute, mi, n second, ss, s millisecond, ms

Was ist das Ergebnis folgender SELECT-Anweisungen?

DB2 ?

```
Select current_date - current_date from sysibm.sysdummy1;
Select current_timestamp(3) - current_timestamp(3) from sysibm.sysdummy1 ;
```

Oracle ?

```
select current_date - current_date from dual;
Select current_timestamp(3) - current_timestamp(3) from dual ;
```

SQL Server ?

```
select getdate() - getdate();
Select current_timestamp(3) - current_timestamp(3) ;
Select current_timestamp - current_timestamp ;
```

--DB2 'YYYY-MM_DD' und Date Duration durxx

```
with zwidate
as
(
 select
 cast('0001-01-01' as date)  as von
,CURRENT_DATE                as heute
,cast('9999-12-31' as date)  as bisopen
from sysibm.sysdummy1
)
select
 von
,heute
,bisopen
,bisopen-heute           as durxx
,heute + (bisopen-heute) as bisopenxx
from zwidate
;
```

--Oracle 'YYYY-MM_DD' und Interval von Tagen intxx

```
alter session set NLS_DATE_FORMAT = 'YYYY-MM-DD HH24:MI:SS';
with zwidate
as
(
 select
 cast('0001-01-01' as date)  as von
,TRUNC( CURRENT_DATE, 'DD')  as heute
,cast('9999-12-31' as date)  as bisopen
from DUAL
)
select
 von
,heute
,bisopen
,bisopen-heute           as intxx
,heute + (bisopen-heute) as bisopenxx
from zwidate
;
```

--SQL Server 'YYYY-MM_DD' und Interval von Tagen intxx

```
with zwidate
as
(
 select
 cast('0001-01-01' as date)  as von
,CAST ( GETDATE() AS DATE)   as heute
,cast('9999-12-31' as date)  as bisopen
)
select
 von
,heute
,bisopen
,DATEDIFF(DD, heute, bisopen) as intxx
,DATEADD( DD, DATEDIFF(DD, heute, bisopen), heute) as bisopenxx
from zwidate
;
```

--DB2 'YYYY-MM_DD hh:mi:ss.xyz' und Timestamp Duration durxx

```
with zwidate
as
(
 select
 cast('0001-01-01 00:00:00.000' as TIMESTAMP(3))  as von
,CURRENT_TIMESTAMP(3)           as heute
,cast('9999-12-31 23:59:59.999' as TIMESTAMP(3))  as bisopen
from sysibm.sysdummy1
)
select
 von,heute,bisopen
,bisopen-heute           as durxx
,heute + (bisopen-heute) as bisopenxx
from zwidate
;
```

--Oracle 'YYYY-MM_DD hh:mi:ss.xyz' und Interval von Tagen „ddddddddd hh:mi:ss.xyz" durxx

```
ALTER SESSION SET NLS_TIMESTAMP_FORMAT = 'YYYY-MM-DD HH24:MI:SS.FF';
with zwidate
as
(
 select
 cast('0001-01-01 00:00:00.000' as TIMESTAMP(3))  as von
,CURRENT_TIMESTAMP(3)           as heute
,cast('9999-12-31 23:59:59.999' as TIMESTAMP(3))  as bisopen
from DUAL
)
select
 von,heute,bisopen
,bisopen-heute           as durxx
,heute + (bisopen-heute) as bisopenxx
from zwidate
;
```

--SQL Server 'YYYY-MM_DD hh:mi:ss.xyz' und --Interval von durxx_??

```
with zwidate as
(
 select
 cast('0001-01-01 00:00:00.000' as DATETIME2(3))  as von
,CAST(GETDATE() AS DATETIME2(3))                  as heute
,cast('9999-12-31 23:59:59.999' as DATETIME2(3))  as bisopen
)
select
 von,heute,bisopen
,DATEDIFF(DD, heute, bisopen)          as durxx_dd
--mit heute DATETIME2 funktioniert + nicht
--,heute  + DATEDIFF(DD, heute, bisopen)
--mit getdate() funktioniert +
--,GETDATE()+ DATEDIFF(DD, heute, bisopen)as "bisopenxxx??"
,DATEADD(DD,DATEDIFF(dd, heute, bisopen),heute) as bisopenxx
from zwidate
;
```

11.10.6 CAST

SQL Standard	DB2	ORACLE	SQL Server
CAST	**CAST**	**CAST**	**CAST**
CAST (d AS DATE) d is not a TIME	CAST (d AS DATE)	TRUNC(d) **??????)** CAST (d AS DATE)	CAST (d AS DATE)
CAST (d AS TIME) d is not a DATE	CAST (d AS TIME)	TO_CHAR(d, 'HH24:MI:SS') ist eine Zeichenkette	CAST (d AS TIME)
CAST (d AS TIMESTAMP) d is a DATE	CAST (d AS TIMESTAMP)	CAST (d AS TIMESTAMP) TRUNC(d)	CAST (d AS DATETIME2)
CAST (d AS TIMESTAMP) d is a TIME	CAST (d AS TIMESTAMP)	CAST (d AS TIMESTAMP) TRUNC(CURRENT_DATE) + (d-TRUNC(d))	CAST (d AS TIMESTAMP2) 1900-01-01 hh:mm:ss.xx
CAST (i AS INTERVAL YEAR TO MONTH)	not possible	CAST (i AS INTERVAL YEAR TO MONTH)	not possible
CAST (i AS INTERVAL DAY TO SECOND)	not possible	CAST (i AS INTERVAL DAY TO SECOND) jd	not possible
CAST (d AS CHAR)	CAST (d AS CHAR(x)) CHAR(d)	CAST (d AS CHAR(x)) TO_CHAR(d, 'YYYY-MM-DD' || 'HH24:MI:SS')	CAST (d AS CHAR(x)) CONVERT (CHAR, d)
CAST (i AS CHAR)		CAST (i AS CHAR(x)) TRUNC(jd,0) || ' ' || TO_CHAR(jd + TO_DATE(1,'J'), 'HH24:MI:SS')	CONVERT (CHAR, is)
CAST (i AS INTEGER) i is YEAR TO DAY	JULIAN_DAY (DATE('0001-01-01') + i) – JULIAN_DAY(DATE('0001-01-01'))		
CAST (i AS INTEGER) i is DAY TO HOUR i is DAY TO MINUTE i is DAY TO SECOND	24 * DAY(i) + HOUR(i) 1440 * DAY(i) + 60 * HOUR(i) + MINUTE(i) 86400 * DAY(i) + 3600 * HOUR(i) + 60 * MINUTE(i) + SECOND(i)		
CAST (i AS INTEGER) i is DAY i is HOUR i is MINUTE i is SECOND		TRUNC(jd, 0) TRUNC(jd * 24, 0) TRUNC(jd * 1440, 0) TRUNC(jd * 86400, 0)	

--Oracle

??????)

```
alter session set NLS_DATE_FORMAT = 'yyyy-mm-dd hh24:mi:ss';
select
 cast (CURRENT_DATE as date) as s1
,trunc (CURRENT_DATE)        as s2
,TO_CHAR(CURRENT_DATE, 'HH24:MI:SS' ) as s3
from dual

S1                        S2                        S3
------------------------- ------------------------- --------
2011-08-26 18:05:02       2011-08-26 00:00:00       18:05:02
```

--DB2 Beispiel "year to day"

```
with dauer as
(
select
00010205. as iyd   --yeartoday
from sysibm.sysdummy1
)
select
 iyd
,JULIAN_DAY (DATE('0001-01-01') + iyd) - JULIAN_DAY(DATE('0001-01-01'))
from dauer
```

--Beispiel DB2 "DAY TO SECOND"

```
with dauer as
(
select
 00000005. as idate
,  123018. as itime
from sysibm.sysdummy1
)
select
 idate
,itime
--,year(idate)                          as jahre
--,month(idate)                         as monate
,day(idate)                            as tage
,24 * DAY(idate) + HOUR(itime)
as stunden
,60 *(24 * DAY(idate) + HOUR(itime)) + MINUTE(itime)
as minuten
,60*((24 * DAY(idate) + HOUR(itime)) + MINUTE(itime)) + SECOND(itime)
as sekunden
from dauer
```

11.10.7 EXTRACT

SQL Standard	DB2	ORACLE	SQL Server
EXTRACT		**EXTRACT**	
EXTRACT (DAY FROM d)	Vgl. DB2 z/OS Version 9 DAY(d)	EXTRACT (DAY FROM d) TRUNC(d, 'DD') – TRUNC(d, 'MM') + 1	 DATEPART(DAY,d)
EXTRACT (DAY FROM i)	Vgl. DB2 z/OS Version 9 DAY(i)	TRUNC(jd, 0)	CONVERT (INTEGER, is/86400)
EXTRACT (HOUR FROM i)	Vgl. DB2 z/OS Version 9 HOUR(i)	TRUNC(jd*24, 0) – (TRUNC(jd, 0)*24)	CONVERT (INTEGER, is/3600)

11.10.8 Operators not in SQL Standard

Diese Liste ist nicht vollständig!

	DB2	ORACLE	SQL Server
Operators not in SQL Standard			
convert Julian day n to DATE		n + TO_DATE(1, 'J')	
pick the earliest date		LEAST(d1, ..., dn)	
pick the latest date		GREATEST(d1, ..., dn)	
pick the last day of the month		LAST_DAY(d))	
get the next day of the week		NEXT_DAY(d, 'Monday')	
	DAYS		
		ADD_MONTHS	
			DATEDIFF
	TIMESTAMPDIFF		
	Im Standard YEAR TO MONTH DAY TO SECOND DB2 for Windows, Linux, Unix	Im Standard YEAR TO MONTH DAY TO SECOND	
	Im Standard (d2-d1) YEAR TO MONTH (d2-d1) DAY TO SECOND DB2 for Windows, Linux, Unix	Im Standard (d2-d1) YEAR TO MONTH (d2-d1) DAY TO SECOND	
	DATE Duration ***TIME Duration*** ***TIMESTAMP Duration***		
convert d to Julian day	JULIAN_DAY(d) Semantik nicht wie Oracle		
convert d to Julian day		TO_CHAR(d, 'J') Semantik nicht wie DB2	
		TRUNC(d, 'MONTH') Monatsanfang	
		TRUNC(d, 'DAY') ***Wochenanfang***	
		TRUNC(d, 'DD') Tagesanfang	

12

Anhang 3: Datendefinition TABLE, Datentyp, Datenintegrität

12 Anhang 3: Datendefinition TABLE, Datentyp, Datenintegrität

12.1 CREATE TABLE

```
DROP TABLE Tvautor;
DROP TABLE Tisbn;
DROP TABLE Tbuch;
DROP TABLE Tverlag;
DROP TABLE Tautor;
COMMIT;
-----------------------------------------------
  CREATE TABLE Tverlag
  (
   Verlagnr   INTEGER    NOT NULL
  ,Verlag     CHAR(20)   NOT NULL
  ,PRIMARY KEY (Verlagnr)
  )
  ;
  GRANT SELECT ON Tverlag TO PUBLIC
  ;
-----------------------------------------------
  CREATE TABLE Tbuch
  (
   Buchnr         INTEGER      NOT NULL
  ,Erschj         DECIMAL(4)
  ,Preis          DECIMAL(7,2)
  ,Verlagnr       INTEGER
  ,Titel          VARCHAR(127) NOT NULL
  ,PRIMARY KEY (Buchnr)
  )
  ;
  ALTER TABLE Tbuch ADD CONSTRAINT
                  FK_Tbuch_Tverlag
                  FOREIGN KEY (Verlagnr)
                  REFERENCES Tverlag(Verlagnr)
                  ;
                --ON DELETE NO ACTION

  GRANT SELECT ON Tbuch TO PUBLIC
  ;
-----------------------------------------------
```

12.2 PRIMARY KEY – FOREIGN KEY – CONSTRAINT – UNIQUE

```
CREATE TABLE Tisbn
  (
   Buchnr    INTEGER NOT NULL
  ,Isbn      CHAR(10)  NOT NULL
  ,Lfdnr     DECIMAL(1) NOT NULL
  ,PRIMARY KEY (Isbn)
  ,CONSTRAINT ALTKEY_Tisbn UNIQUE (Buchnr,Lfdnr)
  ,FOREIGN KEY (Buchnr)
              REFERENCES Tbuch(Buchnr)
              ON DELETE CASCADE
  )
  ;
  GRANT SELECT ON Tisbn  TO PUBLIC
  ;
  ------------------------------------------------
  CREATE TABLE Tautor
  (
   Autornr      INTEGER      NOT NULL
  ,Autor        VARCHAR(240) NOT NULL
  ,Geburtsdatum DATE
  ,PRIMARY KEY (Autornr)
  )
  ;
  GRANT SELECT ON Tautor TO PUBLIC
  ;
  ------------------------------------------------
  CREATE TABLE Tvautor
  (
   Buchnr    INTEGER NOT NULL
  ,Autornr   INTEGER NOT NULL
  ,Lfdnr     DECIMAL(1) NOT NULL
  ,Praemie   DECIMAL(9,3)
  ,PRIMARY KEY (Autornr,Buchnr)
  ,CONSTRAINT ALTKEY_Tvautor UNIQUE (Buchnr,lfdnr)
  ,FOREIGN KEY (Buchnr)
              REFERENCES Tbuch (Buchnr)
              ON DELETE CASCADE
  ,FOREIGN KEY (Autornr)
              REFERENCES Tautor (Autornr)
            --ON DELETE NO ACTION
  )
  ;
  GRANT SELECT ON Tvautor TO PUBLIC
  ;
```

- Eine Table ohne Primärschlüssel bzw. UNIQUE-Constraint kann Duplikate präsentieren! SQL erlaubt Base-Tables ohne Primärschlüssel bzw. UNIQUE-Constraint! **Empfehlung für die Praxis: jede Zeile sollte eindeutig identifizierbar sein, definieren Sie für jede Base-Table einen PRIMARY KEY.**
- Jede Spalte eines Primärschlüssels sollte explizit mit NOT NULL definiert werden.
- SQL-Standard: Jede Spalte eines Primärschlüssels ist implizit NOT NULL, selbst wenn NOT NULL nicht explizit spezifiziert wurde.
- Ein Candidate Key wird mit UNIQUE bzw. PRIMARY KEY definiert. Eine TABLE kann mehrere Candidate Keys haben, aber höchstens einen PRIMARY KEY.
- Es wird bei der Definition der Beziehung festgelegt, was mit den Fremdschlüsseln geschehen soll, wenn der Primärschlüssel gelöscht wird:

```
REFERENCES table ON DELETE NO ACTION
REFERENCES table ON DELETE CASCADE
REFERENCES table ON DELETE SET NULL
REFERENCES table ON DELETE SET DEFAULT
```

- Es wird bei der Definition der Beziehung festgelegt, was mit den Fremdschlüssel geschehen soll, wenn der Primärschlüssel geändert wird:

```
REFERENCES table ON UPDATE NO ACTION
REFERENCES table ON UPDATE CASCADE
REFERENCES table ON UPDATE SET NULL
REFERENCES table ON UPDATE SET DEFAULT
```

- Die meisten Produkte am Markt unterstützen ON DELETE. Nur wenige Produkte am Markt unterstützen ON DELETE SET DEFAULT.
- **Von Oracle wird ON DELETE NO ACTION nicht explizit unterstützt.**
- Nur wenige Produkte am Markt unterstützen ON UPDATE.

12.3 TRUNCATE TABLE

12.3.1 Beispiel DB2 for z/OS Version 9

```
TRUNCATE TABLE Tisbn
         IGNORE DELETE TRIGGERS
         IMMEDIATE;
```

Mit der **TRUNCATE-Anweisung** können unter gewissen Voraussetzungen alle Zeilen aus einer Table sehr effizient entfernt werden (zum Beispiel darf die Table von keinem FOREIGN KEY referenziert werden).

The **IMMEDIATE** option causes the TRUNCATE to be immediately executed and **it cannot be undone**.

If IMMEDIATE is not specified you can issue a ROLLBACK to undo the TRUNCATE.

IGNORE DELETE TRIGGERS tells DB2 to not fire any DELETE triggers.

Alternately, you could specify **RESTRICT WHEN DELETE TRIGGERS**, which will return an error if there are any delete triggers defined on the table.

12.3.2 Beispiel Oracle

```
connect sl01/sl01;
select * from tbuch;
truncate table tbuch;
--nicht möglich, da von einem Fremdschlüssel referenziert


select buchnr, preis from tbuch where buchnr = 5;
update tbuch set preis = 222 where buchnr =5;
select buchnr, preis from tbuch where buchnr = 5;
select * from tisbn;

truncate table tisbn;
--truncate ist mit einem commit verbunden
--ein rollback ist nicht mehr möglich
rollback;
select buchnr, preis from tbuch where buchnr = 5;
select * from tisbn;
```

Mit Hilfe von TRUNCATE kann unter gewissen Voraussetzungen eine Table sehr effizient geputzt werden!

“Oracle CAUTION: You cannot roll back a TRUNCATE statement.”

Oracle Achtung: TRUNCATE beendet die aktuelle Transaktion.

Bei TRUNCATE TABLE wird kein Trigger aktiviert!

12.3.3 Beispiel SQL Server

Mit der **TRUNCATE-Anweisung** können unter gewissen Voraussetzungen alle Zeilen aus einer Table sehr effizient entfernt werden (zum Beispiel darf die Table von keinem FOREIGN KEY referenziert werden, die Table darf an keiner indizierten Sicht beteiligt sein.).

Wenn die Tabelle eine Identitätsspalte enthält, wird der Zähler für diese Spalte auf den Ausgangswert zurückgesetzt, der für die Spalte definiert ist. Wenn kein Ausgangswert definiert wurde, wird der Standardwert 1 verwendet. Falls Sie den Wert des Identitätszählers erhalten möchten, verwenden Sie stattdessen DELETE.

Bei TRUNCATE TABLE wird kein Trigger aktiviert!

Nach TRUNCATE TABLE ist bei SQL Server ein ROLLBACK möglich!

12.4 DROP TABLE

```
DROP TABLE base-table {RESTRICT | CASCADE}
```

- RESTRICT: DROP TABLE wird nicht ausgeführt, wenn die Base-Table über eine Integritätsbedingung oder VIEW-Definition referenziert wird.
- CASCADE: Integritätsbedingungen oder VIEWs, die die Table referenzieren, werden ebenfalls gedropt.
- *Die geschweiften Klammern {...} zeigen an, dass aus den durch einen senkrechten Strich getrennten Elementen genau eins ausgewählt werden muss!*

12.5 Temporary Tables

Zitat Anfang

'In diesem Buch haben wir uns bisher ausschließlich mit dem beschäftigt, was man persistente (permanent or persistent) Basistabellen nennen könnte, wobei wir mit persistent meinen, dass die fragliche Tabelle, einmal erzeugt, solange in der Datenbank verbleibt, bis sie zu einem gewissen Zeitpunkt explizit mit Hilfe einer DROP-Operation gelöscht wird. Manchmal werden aber von Anwendungen Tabellen benötigt, die nur zur Weitergabe von Zwischenergebnissen von einer Anwendung zu einer anderen benutzt werden und somit nur für eine vergleichsweise kurze Zeit (sicher nicht über die Lebenszeit der fraglichen Anwendung hinaus) bestehen bleiben. Weiterhin sind solche Tabellen typischerweise "privat" für die sie benutzende Anwendung – die Frage stellt sich nicht, ob die enthaltenen Daten mit anderen Anwendungen, nebenläufig oder nicht, geteilt werden sollen.

Nun wäre es natürlich möglich, für solche Zwecke persistente Tabellen zu verwenden. Allerdings unterstützt der Standard explizit das Konzept "temporärer" Tabellen (speziell temporäre Basistabellen), die zur Vereinfachung des Prozesses eingesetzt werden können. Kurz gesagt, eine temporäre Tabelle ist eine Basistabelle, die:

- explizit bei ihrer Definition durch TEMPORARY gekennzeichnet wird;
- zum Zeitpunkt des ersten Zugriffs aus dem Anwendungsmodul oder der SQL-Sitzung heraus immer leer ist ...;
- optional bei jeder Freigabe (durch COMMIT) automatisch in einen Leerzustand zurückgesetzt werden kann

und

- automatisch (falls sie nicht bereits explizit gelöscht worden ist) am Ende der SQL-Sitzung gelöscht wird.

Außerdem ist es Transaktionen erlaubt, temporäre Tabellen zu ändern, selbst wenn diese Transaktionen als READ ONLY spezifiziert worden sind.

Im Standard fallen temporäre Tabellen in drei Typen, die wir der Einfachheit halber als Typ 1, Typ 2 und Typ 3 bezeichnen wollen. (Der Standard benutzt die Begriffe *declared local temporary table, created local temporary table* bzw. *global temporary table*; diese Begriffe sind allerdings umständlich …'

Zitat Ende

SQL – Der Standard SQL/92 mit den Erweiterungen CLI und PSM Deutsche Ausgabe des amerikanischen Klassikers Ausblick auf SQL3 Chris J. Date Hugh Darwen Addison Wesley Longman GmbH, 1998, Seite 311

12.5.1 **Beispiel** DB2 Version 9 for Linux, UNIX, and Windows **DECLARE GLOBAL TEMPORARY TABLE**

Zitat Anfang

The DECLARE GLOBAL TEMPORARY TABLE statement defines a temporary table for the current session. The declared temporary table description does not appear in the system catalog. It is not persistent and cannot be shared with other sessions. Each session that defines a declared global temporary table of the same name has its own unique description of the temporary table. When the session terminates, the rows of the table are deleted, and the description of the temporary table is dropped.

Zitat Ende

12.5.2 Beispiel Oracle GLOBAL TEMPORARY TABLE

```
connect sl01/sl01
;
drop table my_temp_tbuch_tisbn
;
CREATE GLOBAL TEMPORARY TABLE my_temp_tbuch_tisbn (
   buchnr  integer   NOT NULL
 , isbn    CHAR (10) NOT NULL
 , lfdnr   decimal (1)
 , PRIMARY  KEY (buchnr, lfdnr)
)
ON COMMIT DELETE ROWS
;
insert into my_temp_tbuch_tisbn
 (buchnr, isbn, lfdnr)
  select tbuch.buchnr, tisbn.isbn, tisbn.lfdnr
  FROM Tbuch inner join tisbn
  ON tbuch.buchnr = tisbn.buchnr
;
select * from my_temp_tbuch_tisbn
;
delete from my_temp_tbuch_tisbn t1
where t1.buchnr = 8
;
select * from my_temp_tbuch_tisbn
;
rollback
;
select * from my_temp_tbuch_tisbn
;
```

Zitat Anfang (Quelle www.oracle_base.com)

Applications often use some form of temporary data store for processes that are to complicated to complete in a single pass. Often, these temporary stores are defined as database tables or PL/SQL tables. In Oracle 8i, the maintenance and management of temporary tables can be delegated to the server by using Global Temporary Tables.

Creation of Global Temporary Tables

The data in a global temporary table is private, such that data inserted by a session can only be accessed by that session. The session-specific rows in a global temporary table can be preserved for the whole session, or just for the current transaction. The `ON COMMIT DELETE ROWS` clause indicates that the data should be deleted at the end of the transaction.

```
CREATE GLOBAL TEMPORARY TABLE my_temp_table (
  column1  NUMBER,
  column2  NUMBER
) ON COMMIT DELETE ROWS;
```

In contrast, the `ON COMMIT PRESERVE ROWS` clause indicates that rows should be preserved until the end of the session.

```
CREATE GLOBAL TEMPORARY TABLE my_temp_table (
  column1  NUMBER,
  column2  NUMBER
) ON COMMIT PRESERVE ROWS;
```

Miscellaneous Features

- If the TRUNCATE statement is issued against a temporary table, only the session specific data is trucated. There is no affect on the data of other sessions.
- Data in temporary tables is automatically delete at the end of the database session, even if it ends abnormally.
- Indexes can be created on temporary tables. The content of the index and the scope of the index is that same as the database session.
- Views can be created against temporary tables and combinations of temporary and permanent tables.
- Temporary tables can have triggers associated with them.
- Export and Import utilities can be used to transfer the table definitions, but no data rows are processed.
- There are a number of restrictions related to temporary tables but these are version specific.

Zitat Ende

12.5.3 Beispiel SQL Server lokale temporäre TABLE und globale temporäre TABLE

„Sie können sowohl lokale als auch globale temporäre Tabellen erstellen. Lokale temporäre Tabellen sind nur während der aktuellen Sitzung sichtbar; globale temporäre Tabellen sind von allen Sitzungen aus sichtbar.“

```
--lokale temporäre table #tablename
use dbbuecher06
GO
drop table #tbuch
create table #tbuch
(buchnr integer not null
,titel  varchar(127)
,primary key (buchnr)
)
--globale temporäre table ##tablename
use dbbuecher06
go
drop table ##tbuch
create table ##tbuch
(buchnr integer not null
,titel  varchar(127)
,primary key (buchnr)
)
insert into #tbuch (buchnr, titel)
  select tbuch.buchnr , tbuch.titel
  from dbbuecher03.dbo.tbuch as tbuch
insert into ##tbuch (buchnr, titel)
  select tbuch.buchnr , tbuch.titel
  from dbbuecher03.dbo.tbuch as tbuch
select * from #tbuch
select * from ##tbuch
go
```

Eine temporäre Tabelle wird automatisch aus dem System entfernt („drop table“), es sei denn sie wurde bereits explizit mit Hilfe von DROP TABLE zerstört.

Beachten Sie bitte, dass im folgenden Auszug aus der SQL Server-Onlinedokumentation mit dem Begriff „löschen“ nicht „delete“ gemeint ist sondern „drop“.

SQL Server-Onlinedokumentation

„Temporäre Tabellen werden automatisch gelöscht, wenn sie nicht mehr benötigt werden, es sei denn, sie wurden bereits explizit unter Verwendung von DROP TABLE gelöscht:

Eine **lokale temporäre** Tabelle, die in einer gespeicherten Prozedur erstellt wurde, wird bei Beendigung der gespeicherten Prozedur automatisch gelöscht. Auf die Tabelle kann durch geschachtelte gespeicherte Prozeduren verwiesen werden, die von der gespeicherten Prozedur ausgeführt werden, die die Tabelle erstellt hat. Auf die Tabelle kann nicht durch den Vorgang verwiesen werden, der die gespeicherte Prozedur, die die Tabelle erstellt hat, aufgerufen hat.

Alle anderen lokalen temporären Tabellen werden am Ende der aktuellen Sitzung automatisch gelöscht.

Eine **globale temporäre Tabelle** wird automatisch gelöscht, wenn die Sitzung, die die betreffende Tabelle erstellt hat, beendet wird und kein Task mehr auf die Tabelle verweist.“

12.6 Datentypen

```
BOOLEAN
BLOB(n)

CHAR(n)
VARCHAR(n)
CLOB(n)

NUMERIC(p,q)
DECIMAL(p,q)

BIGINT
INTEGER
SMALLINT

FLOAT(p)

DATE
TIME
TIME WITH TIME ZONE
TIMESTAMP
TIMESTAMP WITH TIME ZONE
INTERVAL

XML

type constructors
      REF
      ROW
      ARRAY
      MULTISET

user defined distinct types
user defined structured types
```

- Alternative Schreibweisen sind zum Beispiel CHARACTER für CHAR, CHARACTER LARGE OBJECT für CLOB.
- REAL bzw. DOUBLE PRECISION sind alternative Sprechweisen für FLOAT.
- NUMERIC (n,m) ist eine Dezimalzahl mit genau n Stellen, davon m hinter dem Dezimalpunkt.
- DECIMAL(n,m) ist eine Dezimalzahl mit mindestens n Stellen, davon m hinter dem Dezimalpunkt.
- CLOB und BLOB sind String-Datentypen, sie haben trotz ihres Namens mit den Objekten der „object technologie“ nichts zu tun!
- VARCHAR, BIT und BIT VARYING(n) sind in SQL89 nicht definiert!
- BIT(n) und BIT VARYING(n) sind inzwischen aus dem Standard SQL:2003 wieder entfernt worden!
- Der in SQL:1999 beschriebene Datentyp BOOLEAN mit seinen Operatoren NOT, AND, OR sieht nur zwei Wahrheitswerte vor. Der Wahrheitswert *UNKNOWN/unbekannt* wird – seltsamerweise, fälschlicherweise – durch NULL repräsentiert!
- SQL:1999 unterstützt die type constructors REF, ROW und ARRAY.
- SQL:2003 unterstützt BIGINT, XML und den type constructor MULTISET

Achtung Oracle, der Datentyp VARCHAR2 und der Datentyp NUMBER:

- Eine in der CREATE-Anweisung mit VARCHAR definierte Spalte wird von ORACLE mit dem proprietären Datentyp **VARCHAR2** generiert.
- Eine in der CREATE-Anweisung mit INTEGER definierte Spalte wird von ORACLE mit dem proprietären Datentyp **NUMBER** generiert.

12.6.1 DB2 Built-in Datatypes

The DB2 Built-in Data Types

All data types include the null value. The null value is a special value that is distinct from all non-null values and thereby denotes the absence of a (non-null) value. Although all data types include the null value, columns defined as NOT NULL cannot contain null values.

Table 8. Numeric Limits

Description	Limit
Smallest INTEGER value	-2 147 483 648
Largest INTEGER value	+2 147 483 647
Smallest BIGINT value	-9 223 372 036 854 775 808
Largest BIGINT value	+9 223 372 036 854 775 807
Smallest SMALLINT value	-32 768
Largest SMALLINT value	+32 767
Largest decimal precision	31
Smallest DOUBLE value	-1.79769E+308
Largest DOUBLE value	+1.79769E+308
Smallest positive DOUBLE value	+2.225E-307
Largest negative DOUBLE value	-2.225E-307
Smallest REAL value	-3.402E+38
Largest REAL value	+3.402E+38
Smallest positive REAL value	+1.175E-37
Largest negative REAL value	-1.175E-37

Table 9. String Limits

Description	Limit
Maximum length of CHAR (in bytes)	254
Maximum length of VARCHAR (in bytes)	32 672
Maximum length of LONG VARCHAR (in bytes)	32 700
Maximum length of CLOB (in bytes)	2 147 483 647
Maximum length of GRAPHIC (in characters)	127
Maximum length of VARGRAPHIC (in characters)	16 336
Maximum length of LONG VARGRAPHIC (in characters)	16 350
Maximum length of DBCLOB (in characters)	1 073 741 823
Maximum length of BLOB (in bytes)	2 147 483 647
Maximum length of character constant	32 672
Maximum length of graphic constant	16 336
Maximum length of concatenated character string	2 147 483 647
Maximum length of concatenated graphic string	1 073 741 823
Maximum length of concatenated binary string	2 147 483 647
Maximum number of hex constant digits	16 336
Maximum size of a catalog comment (in bytes)	254
Largest instance of a structured type column object at run time (in gigabytes)	1

Table 10. Datetime Limits

Description	Limit
Smallest DATE value	0001-01-01
Largest DATE value	9999-12-31
Smallest TIME value	00:00:00
Largest TIME value	24:00:00
Smallest TIMESTAMP value	0001-01-01-00.00.00.000000
Largest TIMESTAMP value	9999-12-31-24.00.00.000000

12.6.2 Oracle Built-in Datatypes

The table that follows summarizes Oracle built-in datatypes. Refer to the syntax in the preceding sections for the syntactic elements. The codes listed for the datatypes are used internally by Oracle Database. The datatype code of a column or object attribute is returned by the DUMP function.

Table 2-1 Built-in Datatype Summary

Code	Datatype	Description
1	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.
1	NVARCHAR2(*size*)	Variable-length Unicode character string having maximum length *size* characters. The number of bytes can be up to two times *size* for AL16UTF16 encoding and three times *size* for UTF8 encoding. Maximum *size* is determined by the national character set definition, with an upper limit of 4000 bytes. You must specify *size* for NVARCHAR2.
2	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.
2	FLOAT [(*p*)]	A subtype of the NUMBER datatype having precision *p*. A FLOAT value is represented internally as NUMBER. The precision *p* can range from 1 to 126 binary digits. A FLOAT value requires from 1 to 22 bytes.
8	LONG	Character data of variable length up to 2 gigabytes, or 2^{31} -1 bytes. Provided for backward compatibility.
12	DATE	Valid date range from January 1, 4712 BC, to December 31, 9999 AD. The default format is determined explicitly by the NLS_DATE_FORMAT parameter or implicitly by the NLS_TERRITORY parameter. The size is fixed at 7 bytes. This datatype contains the datetime fields YEAR, MONTH, DAY, HOUR, MINUTE, and SECOND. It does not have fractional seconds or a time zone.
21	BINARY_FLOAT	32-bit floating point number. This datatype requires 5 bytes, including the length byte.
22	BINARY_DOUBLE	64-bit floating point number. This datatype requires 9 bytes, including the length byte.
180	TIMESTAMP [(*fractio-*	Year, month, and day values of date, as well as hour, mi-

Code	Datatype	Description
	nal_seconds_precision)]	nute, and second values of time, where *fraction-al_seconds_precision* is the number of digits in the fractional part of the SECOND datetime field. Accepted values of *fractional_seconds_precision* are 0 to 9. The default is 6. The default format is determined explicitly by the NLS_DATE_FORMAT parameter or implicitly by the NLS_TERRITORY parameter. The sizes varies from 7 to 11 bytes, depending on the precision. This datatype contains the datetime fields YEAR, MONTH, DAY, HOUR, MINUTE, and SECOND. It contains fractional seconds but does not have a time zone.
181	TIMESTAMP [(*fraction-al_seconds*)] WITH TIME ZONE	All values of TIMESTAMP as well as time zone displacement value, where *fractional_seconds_precision* is the number of digits in the fractional part of the SECOND datetime field. Accepted values are 0 to 9. The default is 6. The default format is determined explicitly by the NLS_DATE_FORMAT parameter or implicitly by the NLS_TERRITORY parameter. The size is fixed at 13 bytes. This datatype contains the datetime fields YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, TIMEZONE_HOUR, and TIMEZONE_MINUTE. It has fractional seconds and an explicit time zone.
231	TIMESTAMP [(*fraction-al_seconds*)] WITH LOCAL TIME ZONE	All values of TIMESTAMP WITH TIME ZONE, with the following exceptions: • Data is normalized to the database time zone when it is stored in the database. • When the data is retrieved, users see the data in the session time zone. The default format is determined explicitly by the NLS_DATE_FORMAT parameter or implicitly by the NLS_TERRITORY parameter. The sizes varies from 7 to 11 bytes, depending on the precision.
182	INTERVAL YEAR [(*year_precision*)] TO MONTH	Stores a period of time in years and months, where *year_precision* is the number of digits in the YEAR datetime field. Accepted values are 0 to 9. The default is 2. The size is fixed at 5 bytes.
183	INTERVAL DAY [(*day_precision*)] TO SECOND [(*fraction-al_seconds*)]	Stores a period of time in days, hours, minutes, and seconds, where • *day_precision* is the maximum number of digits in the DAY datetime field. Accepted values are 0 to 9. The default is 2. • *fractional_seconds_precision* is the number of digits in the fractional part of the SECOND field. Accepted values are 0 to 9. The default is 6.

Code	Datatype	Description
		The size is fixed at 11 bytes.
23	RAW(*size*)	Raw binary data of length *size* bytes. Maximum *size* is 2000 bytes. You must specify *size* for a RAW value.
24	LONG RAW	Raw binary data of variable length up to 2 gigabytes.
69	ROWID	Base 64 string representing the unique address of a row in its table. This datatype is primarily for values returned by the ROWID pseudocolumn.
208	UROWID [(*size*)]	Base 64 string representing the logical address of a row of an index-organized table. The optional *size* is the size of a column of type UROWID. The maximum size and default is 4000 bytes.
96	CHAR [(*size* [BYTE | CHAR])]	Fixed-length character data of length *size* bytes or characters. Maximum *size* is 2000 bytes or characters. Default and minimum *size* is 1 byte. BYTE and CHAR have the same semantics as for VARCHAR2.
96	NCHAR[(*size*)]	Fixed-length character data of length *size* characters. The number of bytes can be up to two times *size* for AL16UTF16 encoding and three times *size* for UTF8 encoding. Maximum *size* is determined by the national character set definition, with an upper limit of 2000 bytes. Default and minimum *size* is 1 character.
112	CLOB	A character large object containing single-byte or multi-byte characters. Both fixed-width and variable-width character sets are supported, both using the database character set. Maximum size is (4 gigabytes – 1) * (database block size).
112	NCLOB	A character large object containing Unicode characters. Both fixed-width and variable-width character sets are supported, both using the database national character set. Maximum size is (4 gigabytes – 1) * (database block size). Stores national character set data.
113	BLOB	A binary large object. Maximum size is (4 gigabytes – 1) * (database block size).
114	BFILE	Contains a locator to a large binary file stored outside the database. Enables byte stream I/O access to external LOBs residing on the database server. Maximum size is 4 gigabytes.

12.6.3 Datentypen von SQL Server

Name	Beschreibung
bigint	Ganzzahlige Daten von -2^63 (-9223372036854775808) bis 2^63-1 (9223372036854775807). Die Speichergröße beträgt 8 Byte.
binary [(*n*)]	Binärdaten fester Länge mit einer Länge von *n* Byte, wobei *n* ein Wert zwischen 1 und 8.000 ist. Die Speicherplatzgröße beträgt *n* Byte.
bit	Ein ganzzahliger Datentyp, der den Wert 1, 0 oder NULL annehmen kann.
char [(*n*)]	Nicht-Unicode-Zeichendaten fester Länge mit *n* Byte. *n* muss ein Wert zwischen 1 und 8.000 sein. Die Speichergröße beträgt *n* Byte. Das SQL-2003-Synonym für char lautet character.
cursor	Ein Datentyp für Variablen oder für OUTPUT-Parameter von gespeicherten Prozeduren, die einen Verweis auf einen Cursor enthalten. Alle Variablen, die mit dem cursor-Datentyp erstellt wurden, lassen NULL zu. Der cursor-Datentyp kann nicht für eine Spalte in einer CREATE TABLE-Anweisung verwendet werden.
DATE	**SQL Server 2008**
Datetime	Datums- und Zeitdaten vom 1. Januar 1753 bis zum 31. Dezember 9999 mit einer Genauigkeit von 300stel-Sekunden, also 3,33 Millisekunden.
DATETIME2	**SQL Server 2008**
DATETIMEOFFSET	**SQL Server 2008**
decimal[(*p*[, *s*])] und numeric[(*p*[, *s*])]	Numerische Datentypen mit fester Genauigkeit und fester Anzahl von Dezimalstellen. Zahlen mit fester Genauigkeit und mit fester Anzahl von Dezimalstellen. Bei maximaler Genauigkeit befinden sich die gültigen Werte in einem Bereich von – 10^38 +1 bis 10^38 – 1. Die SQL-92-Synonyme für decimal sind dec und dec(*p*, *s*). numeric entspricht bezüglich der Funktion decimal.
float [(*n*)]	Eine Gleitkommazahl im Bereich von – 1.79E + 308 bis 1.79E + 308. *n* gibt an, wie viele Bits zum Speichern der Mantisse der float-Zahl in der wissenschaftlichen Schreibweise verwendet werden, und legt dadurch die Genauigkeit und Speichergröße fest. Wenn *n* angegeben wird, muss es sich um einen Wert zwischen 1 und 53 handeln. Der Standardwert von *n* ist 53.
Image	Binärdaten variabler Länge von 0 bis 2^{31}-1 (2.147.483.647) Byte.
Int	Ganzzahlige Daten von -2^31 (-2.147.483.648) bis 2^31 – 1 (2.147.483.647).
money	Währungsdatenwerte von -2^63 (-922.337.203.685.477,5808) bis 2^63 – 1 (+922.337.203.685.477,5807) mit der Genauigkeit von einem Zehntausendstel einer Währungseinheit.
nchar [(n)]	Unicode-Zeichendaten fester Länge mit *n* Zeichen. *n* muss ein Wert

Name	Beschreibung
	zwischen 1 und 4.000 sein. Die Speichergröße beträgt das Zweifache von *n* Bytes. Die SQL-2003-Synonyme für nchar sind national char und national character.
ntext	Unicode-Daten variabler Länge mit einer maximalen Länge von 2^30 – 1 (1.073.741.823) Zeichen. Die Speichergröße in Bytes ist doppelt so groß wie die Anzahl von eingegebenen Zeichen. Das SQL-2003-Synonym für ntext ist national text.
numeric	Entspricht funktional dem Datentyp decimal.
nvarchar [(n | max)]	Unicode-Zeichendaten variabler Länge. *n* kann ein Wert zwischen 1 und 4.000 sein. max gibt an, dass die maximale Speichergröße 2^31-1 Bytes beträgt. Die Speichergröße in Bytes ist doppelt so groß wie die Anzahl eingegebener Zeichen + 2 Bytes. Die eingegebenen Daten können 0 Zeichen lang sein. Die SQL-2003-Synonyme für nvarchar sind national char varying und national character varying.
real	Gleitkommazahl im Bereich von -3.40E + 38 bis 3.40E + 38. Die Speichergröße beträgt 4 Byte. Das SQL-92-Synonym für real ist float(24).
smalldatetime	Datums- und Zeitdaten vom 1. Januar 1900 bis zum 6. Juni 2079 mit einer Genauigkeit von einer Minute.
smallint	Ganzzahlige Daten von -2^15 (-32.768) bis 2^15 – 1 (32.767). Die Speichergröße beträgt 2 Byte.
smallmoney	Währungsdatenwerte zwischen -214.748,3648 und +214.748,3647 mit der Genauigkeit eines Zehntausendstels der Währungseinheit. Die Speichergröße beträgt 4 Byte.
sql_variant	Ein Datentyp, der Werte verschiedener SQL Server 2005-gestützter Datentypen speichert, ausgenommen text, ntext, image, timestamp und sql_variant. sql_variant kann in Spalten, Parametern, Variablen und Rückgabewerten von benutzerdefinierten Funktionen verwendet werden. sql_variant lässt zu, dass diese Datenbankobjekte Werte anderer Datentypen unterstützen.
table	Ein besonderer Datentyp, der zum Speichern eines Resultsets für die spätere Verarbeitung verwendet werden kann. table wird primär zum temporären Speichern einer Reihe von Zeilen verwendet, die als Resultset einer Tabellenwertfunktion zurückgegeben werden.
text	Nicht-Unicode-Daten variabler Länge in der Codepage des Servers und mit einer maximalen Länge von 2^31-1 (2.147.483.647) Zeichen. Auch wenn die Servercodepage Doppelbyte-Zeichen verwendet, ist der Speicherplatz 2.147.483.647 Bytes groß. Abhängig von der Zeichenfolge kann die Speichergröße unter 2.147.483.647 Bytes liegen.

TIME	SQL Server 2008
timestamp	Ein Datentyp, der automatisch generierte eindeutige, binäre Zahlen in einer Datenbank verfügbar macht. timestamp wird normalerweise als Mechanismus für die Erstellung einer Versionskennung von Tabellenzeilen verwendet. Die Speichergröße beträgt 8 Bytes.
tinyint	Ganzzahlige Daten von 0 bis 255. Die Speichergröße beträgt 1 Byte.
uniqueidentifier	Die Merge- und die Transaktionsreplikation mit Abonnements mit Aktualisierung verwenden uniqueidentifier-Spalten. Dadurch wird sichergestellt, dass die Zeilen über mehrere Kopien der Tabelle hinweg eindeutig identifiziert werden.
varbinary [(*n* | max)]	Binärdaten variabler Länge. *n* kann ein Wert zwischen 1 und 8.000 sein. max zeigt an, dass die maximale Speicherplatzgröße 2^31-1 Byte beträgt. Die Speicherplatzgröße ist die tatsächliche Länge der eingegebenen Daten + 2 Byte. Die eingegebenen Daten können 0 Byte lang sein. Das SQL-2003-Synonym für varbinary ist binary varying.
varchar [(*n* | max)]	Nicht-Unicode-Zeichendaten variabler Länge. *n* kann ein Wert von 1 bis 8.000 sein. max gibt an, dass die maximale Speichergröße 2^31-1 Byte beträgt. Die Speichergröße ist die tatsächliche Länge der eingegebenen Daten + 2 Byte. Die eingegebenen Daten können 0 Zeichen lang sein. Die SQL-2003-Synonyme für varchar lauten char varying oder character varying.
Xml [(*xml_schema_collection*)]	Der Datentyp, in dem XML-Daten gespeichert sind. xml-Instanzen können in einer Spalte oder in einer Variablen vom Typ xml gespeichert werden. Die gespeicherte Darstellung von Instanzen vom Datentyp xml darf die Größe von 2 Gigabyte (GB) nicht überschreiten.

12.7 Datenintegrität

- **Prüfen des Datentyps**

- **user defined distinct types**
- **DOMAIN Constraint *)**

- **user defined structured types**

- **Primärschlüssel PRIMARY KEY**
- **Candidate Key UNIQUE**

- **Fremdschlüssel FOREIGN KEY**

- **Check-Constraint**
- **ASSERTION General Constraint **)**

- **Trigger**

- **VIEW mit CHECK-Option**

*) DOMAIN wird nur von wenigen Produkten am Markt unterstützt und wird hier nur der Dokumentation wegen erwähnt.

**) ASSERTION wird nur von wenigen Produkten am Markt unterstützt und wird hier nur der Dokumentation wegen erwähnt.

Vergleichen Sie dazu bitte die Diskussion zu benutzerdefinierten Datentypen.

12.7.1 DOMAIN

```
CREATE DOMAIN
  Djahre DECIMAL(4) NOT NULL DEFAULT 1900
    CONSTRAINT Conjahre CHECK
                ( VALUE BETWEEN 1900 AND 2000 );


CREATE TABLE Tbuch
  (
   Buchnr         INTEGER      NOT NULL
  ,Erschj         Djahre
  ,Preis          DECIMAL(7,2)
  ,Verlagnr       INTEGER
  ,Titel          VARCHAR(127) NOT NULL
  ,PRIMARY KEY (Buchnr)
  )
  ;
```

- Ein DOMAIN ist eine Menge erlaubter Werte eines Datentyps. Optional können auch DEFAULT-Werte zugewiesen werden.
- Beim CREATE TABLE kann dann der DOMAIN-Name angegeben werden.
- Bei DROP DOMAIN CASCADE erbt eine referenzierende Spalte den Datentyp, den DEFAULT-Wert und die Integritätsbedingungen des DOMAIN!
- ALTER DOMAIN ist auch möglich.

```
CREATE DOMAIN domain [ AS ] data-type
    [ DEFAULT { literal | niladic-function | NULL } ]
    [ [CONSTRAINT constraint]
       CHECK (conditional-expression) ]
DROP DOMAIN domain RESTRICT;
DROP DOMAIN domain CASCADE;
ALTER DOMAIN domain SET default-definition;
ALTER DOMAIN domain DROP DEFAULT;
ALTER DOMAIN domain ADD domain-constraint-definition;
ALTER DOMAIN domain DROP CONSTRAINT constraint;
```

12.7.2 ASSERTION

Beliebig viele Spalten von beliebig vielen Tables sind an einer allgemeinen Integritätsbedingung beteiligt.

```
... CONSTRAINT con111 CHECK (Tisbn.lfdnr <5)


CREATE ASSERTION ass111 CHECK
      ((SELECT MAX(Tisbn.Lfdnr) FROM Tisbn) < 5)



... CONSTRAINT con222 CHECK
      (TBUCH.Erschj < 1950 AND TBUCH.Preis < 99.99)


CREATE ASSERTION ass222 CHECK
      (NOT EXISTS
                  (SELECT * FROM Tbuch
                  WHERE NOT(TBUCH.Erschj < 1950
                  AND
                  TBUCH.Preis < 99.99)
                  )
      )
```

- Jede Check-Constraint einer Base-Table kann auch als ASSERTION definiert werden!

12.8 NULL und DEFAULT

```
Buchnr  INTEGER NOT NULL
```

die Spalte Buchnr präsentiert immer einen numerischen Wert

```
Erschj  DECIMAL (4)
```

das Erscheinungsjahr muss im System nicht definiert sein

```
Erschj  DECIMAL (4) NOT NULL DEFAULT 9999.

Preis      DECIMAL (7,2) DEFAULT 0.0

Preis      DECIMAL (7,2) DEFAULT NULL

Titel      VARCHAR (127) DEFAULT ´???´
```

- DEFAULT-Wert kann sein: ein Literal, NULL oder eine niladic function (USER, CURRENT_USER, SESSION_USER, SYSTEM_USER, CURRENT_DATE, CURRENT_TIME, CURRENT_TIMESTAMP)
- **Was ist der DEFAULT von VARCHAR? Was ist die Länge des DEFAULTs von VARCHAR?**

  DB2 und SQL Server:

```
alter Table Tautor ADD Vorname VARCHAR(200);
select autornr, len(vorname) lenvorname
from tautor;
--db2 length(vorname) ist jetzt NULL
--sql server len(vorname) ist jetzt NULL
update tautor set vorname='';
select autornr, len(vorname) lenvorname
from tautor;
--db2 length(vorname) ist jetzt 0
--sql server len(vorname) ist jetzt 0
```

  Oracle:

```
alter Table Tautor ADD Vorname VARCHAR(200);
COMMIT;
select * from tautor;
update tautor set vorname='';
select autornr, length(vorname) lenvorname
from tautor;
--oracle die Länge ist jetzt NULL
```

- Folgende Syntax wird von manchen Produkten am Markt unterstützt, aber nicht von SQL:1992!

```
Preis  DECIMAL (7,2) NOT NULL WITH DEFAULT
```

12.9 ALTER TABLE und ADD CONSTRAINT…CHECK

```
ALTER TABLE Tbuch
      DROP CONSTRAINT Tbuchconpreis
;
COMMIT;

ALTER TABLE Tbuch
      ADD CONSTRAINT Tbuchconpreis
      CHECK (Preis > 0.00)
;
COMMIT;

UPDATE Tbuch SET PREIS = -9999.99;
```

- Der Integritätsbedingung kann explizit ein Name gegeben werden.
- Alle Daten in der Table müssen der Bedingung **TRUE oder UNKNOWN** sein.

DB2

DB21034E Der Befehl wurde als SQL-Anweisung verarbeitet, da es sich um keinen gültigen Befehl des Befehlszeilenprozessors handelte. Während der SQL-Verarbeitung wurde folgendes ausgegeben:

SQL0545N Die angeforderte Operation ist nicht zulässig, da eine Zeile gegen die Prüfung auf Integritätsbedingung "DB2ADMIN.TBUCH.TBUCHCONPREIS" verstößt.

SQLSTATE=23513

SQL Server

Server: Nachr.-Nr. 547, Schweregrad 16, Status 1, Zeile 1

Die UPDATE-Anweisung verstieß gegen die COLUMN CHECK-Einschränkung 'Tbuchconpreis'. Der Konflikt trat in der sample-Datenbank, Tabelle 'Tbuch', column 'Preis' auf.

Die Anweisung wurde beendet.

12.10 ALTER TABLE, UNIQUE INDEX und Schlüsselkandidaten

```
DROP TABLE Tisbn;
CREATE TABLE Tisbn
(
 Buchnr         INTEGER     NOT NULL
,Isbn           CHAR (10)   NOT NULL
,Lfdnr          DECIMAL (1) NOT NULL
);
CREATE UNIQUE INDEX Xisbn01 on Tisbn(isbn)
;
CREATE UNIQUE INDEX Xisbn02 on Tisbn(buchnr, lfdnr)
;
--Der unique index auf den Schlüsseln
--soll explizit generiert werden!
--CREATE INDEX ist eine Anweisung
--der physischen Ebene und deshalb
--was die Parameter betrifft
--vom SQL Standard nicht festgelegt!
ALTER TABLE Tisbn
      ADD PRIMARY KEY (Isbn)
;
ALTER TABLE Tisbn
      ADD CONSTRAINT alt_schl
      UNIQUE(buchnr, lfdnr)
;
ALTER TABLE Tisbn
      ADD CONSTRAINT FK_tisbn_tbuch
      FOREIGN KEY (Buchnr)
      REFERENCES Tbuch ON DELETE CASCADE
;
/*
ALTER TABLE Tisbn
      DROP CONSTRAINT FK_tisbn_tbuch
;
*/
ALTER TABLE Tisbn
      ADD CONSTRAINT con_1_bis_5
      CHECK ( 1<= Lfdnr AND Lfdnr <= 5)
;
ALTER TABLE Tautor DROP COLUMN Geburtsort;
ALTER TABLE Tautor ADD         Geburtsort CHAR (20);
```

Achtung Oracle:

```
ALTER TABLE tautor ADD          geburtsort char(20);
ALTER TABLE tautor DROP COLUMN geburtsort;
```

Bei ADD darf COLUMN nicht codiert werden.

Bei DROP muss COLUMN codiert werden.

```
ALTER TABLE base-table
      ADD [COLUMN] column-definition
```

- Eine Spalte kann hinzugefügt werden. Das optionale Schlüsselwort COLUMN ist ein Füllwort und kann ausgelassen werden.
- *Die eckigen Klammern [...] zeigen an, dass das umschlossene Material optional ist. Aus durch einen senkrechten Strich getrennten Elementen kann höchstens eins ausgewählt werden!*

```
ALTER TABLE base-table
      DROP [COLUMN] column {RESTRICT | CASCADE}
```

- Eine Spalte kann entfernt werden. Das optionale Schlüsselwort COLUMN ist ein Füllwort und kann ausgelassen werden.
- RESTRICT: Sofern die SPALTE durch eine VIEW-Definition oder Integritätsbedingung referenziert wird, wird der DROP nicht ausgeführt.
- CASCADE: ...

```
ALTER TABLE base-table
      ALTER [COLUMN] column
      {SET default-definition | DROP DEFAULT}
```

- Eine DEFAULT-Definition kann entfernt oder verändert werden. Das optionale Schlüsselwort COLUMN ist ein Füllwort und kann ausgelassen werden.

```
ALTER TABLE base-table
      ADD base-table-constraint-definition
ALTER TABLE base-table
      DROP CONSTRAINT constraint {RESTRICT | CASCADE}
```

- Integritätsbedingungen können entfernt bzw. aufgebaut werden.
- RESTRICT und CASCADE beziehen sich auf candidate-key-definition PRIMARY KEY bzw. UNIQUE, sofern irgendein FOREIGN KEY den candidate-key referenziert, wird der DROP nur bei CASCADE ausgeführt.

12.11 Abgeleitete Spalten

Beispiel DB2

```
--connect to sample user db2admin using KENNWORT;
drop table tbuchccc
;
create table tbuchccc
(buchnr integer not null
 ,preis decimal (7,2) NOT NULL
 ,preisbrutto   GENERATED ALWAYS AS (preis*1.16)
 ,PRIMARY KEY (buchnr)
);
insert into tbuchccc (buchnr, preis) values (5, 55.55);
insert into tbuchccc (buchnr, preis) values (6, 66.66);
select * from tbuchccc;
```

Beispiel Oracle 11g Virtuelle Spalte

Oracle 11g has introduced a new feature that allows you to create a "virtual column", an empty column that contains a function upon other table columns (the function itself is stored in the data dictionary).

Beispiel SQL Server

```
drop table tbuchccc
;
create table tbuchccc
(buchnr integer not null
 ,preis decimal (7,2) NOT NULL
 ,preisbrutto AS (preis*1.16)
 ,PRIMARY KEY (buchnr)
);
insert into tbuchccc (buchnr, preis) values (5,
55.55);
insert into tbuchccc (buchnr, preis) values (6,
66.66);
select * from tbuchccc;
```

12.12 CREATE TABLE … AS, CREATE TABLE … LIKE

--Achtung: die Constraints werden nicht mitgeneriert!

Beispiel DB2

```
drop table tbuchddd
;
create table tbuchddd as
  (select buchnr , preis from tbuch
  ) WITH NO DATA
;
Select * from tbuchddd
;
drop table tbucheee
;
create table tbucheee LIKE tbuch
;
Select * from tbucheee
;
```

Beispiel Oracle

```
drop table tbuchneu
;
create table tbuchneu (buchnr, preis)
as
  (select buchnr , preis from tbuch
  )
;
select * from tbuchneu
;
```

Beispiel SQL Server

```
drop table tbuchnrtitel
;
select buchnr, titel into tbuchnrtitel
from tbuch
;
Alter table tbuchnrtitel add primary key(buchnr)
;
select * from tbuchnrtitel
;
delete from tbuchnrtitel
;
```

12.13 IDENTITY

IDENTITY ermöglicht im Rahmen einer Table die automatische Vergabe einer laufenden Nummer.

Beispiel DB2

```
connect to sample user db2admin using KENNWORT
;
drop table tbuchbbb
;
create table tbuchbbb
( buchnr integer not null
         generated always as identity
         (
          start with 1000
          , increment by 10
          , minvalue 100
          , no maxvalue
          , no cycle
         )
 , titel varchar(127) not null
 , primary key (buchnr)
);
insert into tbuchbbb (titel) values ('AAAA')
;
insert into tbuchbbb (titel) values ('BBBB')
;
select * from tbuchbbb
;
```

Beispiel SQL Server

```
use sample
go
DROP TABLE tbuchbbb
GO
CREATE TABLE tbuchbbb
( buchnr INTEGER      NOT NULL
         IDENTITY(1000,10)
 ,titel  VARCHAR(127) NOT NULL
 ,primary key (buchnr)
)
GO
insert into tbuchbbb (titel) values ('AAAA');
SET IDENTITY_INSERT tbuchbbb ON;
insert into tbuchbbb (buchnr,titel) values (777, 'BBBB');
SET IDENTITY_INSERT tbuchbbb OFF;
insert into tbuchbbb (titel) values ('CCCC');
insert into tbuchbbb (titel) values ('CCCC');
select * from tbuchbbb;
```

Beispiel Oracle mit SEQUENCE

```
DROP SEQUENCE buchnr_seq
;
CREATE SEQUENCE buchnr_seq
start with 1000
increment by 10
;
drop table tbuchbbb
;
create table tbuchbbb
(buchnr integer     not null
,titel varchar(127) not null
,primary key (buchnr)
)
;
insert into tbuchbbb (buchnr, titel)
values (buchnr_seq.nextval, 'AAAA')
;
insert into tbuchbbb (buchnr, titel)
values (buchnr_seq.nextval, 'BBBB')
;
select * from tbuchbbb
;
```

Die folgende Lösung ist natürlich auch möglich:

```
drop table tbuchbbb
;
create table tbuchbbb
(buchnr integer not null
,titel varchar(127) not null
,primary key (buchnr)
)
;
insert into tbuchbbb (buchnr, titel)
values
(
  COALESCE
  (
     (SELECT MAX(buchnr) + 10  from tbuchbbb)
     , 1000
  )
,'AAAA'
)
;
insert into tbuchbbb (buchnr, titel)
values
(
  COALESCE
  (
     (SELECT MAX(buchnr) + 10  from tbuchbbb)
     , 1000
  )
,'BBBB'
)
;
select * from tbuchbbb
;
```

12.14 SEQUENCE

Eine SEQUENCE ist ein eigenständiges Objekt für die automatische Nummernvergabe.

Beispiel DB2

```
drop sequence buchnr_seq
;
create sequence buchnr_seq
start with 1
increment by 5
no maxvalue
no cycle
;
select 'erster aufruf'                as s1
     , next value for buchnr_seq      as s2
from sysibm.sysdummy1;
select 'zweiter aufruf'               as s1
     , next value for buchnr_seq      as s2
from sysibm.sysdummy1;
```

Beispiel Oracle

```
drop sequence buchnr_seq
;
create sequence buchnr_seq
start with 1
increment by 5
;
select 'erster aufruf'                as s1
     , buchnr_seq.nextval             as s2
from dual;
select 'zweiter aufruf'               as s1
     , buchnr_seq.nextval             as s2
from dual
;
select 'dritter aufruf'               as s1
     , buchnr_seq.nextval             as s2
from dual
;
```

SQL Server 2012

Was liefert auf Ihrem Oracle-Server der 3. und 4. Aufruf nach dem zweiten Create, der 5. und 6. Aufruf nach dem dritten Create?

```
drop sequence buchnr_seq
;
create sequence buchnr_seq
start with 1
increment by 5
CACHE 77
;
select '1.aufruf'                    as s1
     , buchnr_seq.nextval            as s2
from dual;
select '2. aufruf'                   as s1
     , buchnr_seq.nextval            as s2
from dual
;
create sequence buchnr_seq
start with 1
increment by 5
;
select '3.aufruf'                    as s1
     , buchnr_seq.nextval            as s2
from dual;
select '4. aufruf'                   as s1
     , buchnr_seq.nextval            as s2
from dual
;
create sequence buchnr_seq
start with 1
increment by 5
;
select '5.aufruf'                    as s1
     , buchnr_seq.nextval            as s2
from dual;
select '6. aufruf'                   as s1
     , buchnr_seq.nextval            as s2
from dual
;
1, 6, 386, 391, 771, 776
```

Ist das ein Bug oder ein Feature? Natürlich ein Feature (77*5 = 385), wenn der zweite Create schief geht wird der Cache neu gefüllt mit 386, 391, 396 etc., wenn der dritte Create schief geht wird der Cache wieder neu gefüllt mit 771, 776, 781 etc.

12.15 TRIGGER

12.15.1 TRIGGER und DB2, ein Beispiel

Mit Hilfe der Trigger können ereignisabhängige Verarbeitungsschritte angestoßen werden. Die Anwendungslogik kann somit in das RDBMS übertragen werden.

```
drop table tbuchlog
;
CREATE TABLE Tbuchlog
(
schl integer      NOT NULL
GENERATED ALWAYS AS IDENTITY
(START WITH 1, INCREMENT BY 1)
,type   char(1)   NOT NULL
,wann   timestamp NOT NULL
,wer    char(8)   NOT NULL
,anzahl integer   NOT NULL
,PRIMARY KEY (schl)
)
;
```

after insert for each statement trigger

```
--dieser trigger dokumentiert
--fuer ein insert-statement
--die anzahl der eingefuegten zeilen
create trigger tbuch_trig1
after insert on tbuch
referencing new_table as neuedaten
   for each statement mode db2sql
   insert into tbuchlog
          (type, wann, wer, anzahl)
   values ( 'i', current timestamp, user,
            (select count(*) from neuedaten))
;
```

after update for each statement trigger

```
--dieser trigger dokumentiert
--fuer ein update-statement
--die anzahl der veraenderten zeilen
create trigger tbuch_trig2
after update on tbuch
  referencing new_table as neuedaten
  for each statement mode db2sql
  insert into tbuchlog
         (type, wann, wer, anzahl)
  values ( 'u', current timestamp, user,
           (select count(*) from neuedaten))
;
```

after delete for each statement trigger

```
--dieser trigger dokumentiert
--fuer ein delete-statement
--die anzahl der geloeschten zeilen
create trigger tbuch_trig3
  after delete on tbuch
  referencing old_table as altedaten
  for each statement mode db2sql
  insert into tbuchlog
         (type, wann, wer, anzahl)
  values ( 'd', current timestamp, user,
           (select count(*) from altedaten))
;
```

12.15.2 TRIGGER und Oracle, ein Beispiel

Gespeicherte Funktionen, gespeicherte Prozeduren und TRIGGER werden mit Hilfe von PL/SQL generiert.

Mit Hilfe der Trigger können ereignisabhängige Verarbeitungsschritte angestoßen werden. Die Anwendungslogik kann somit in das RDBMS übertragen werden.

```
spool d:\mango.txt;
DROP TABLE tbuchlogzeile;
CREATE TABLE tbuchlogzeile
(
  schl           INTEGER        not null,
  type           char(1)      not null,
  wann           timestamp(9) not null ,
  wer            char(8)      not null,
  buchnrneu     INTEGER,
  buchnralt     INTEGER,
  preisneu       decimal(7,2),
  preisalt       decimal(7,2),
  PRIMARY KEY (schl)
);
```

--der TRIGGER tbuch_trigzeile protokolliert für

--INSERT, UPDATE und DELETE jede geänderte Zeile

```
CREATE OR REPLACE TRIGGER tbuch_trigzeile
AFTER  INSERT OR UPDATE OR DELETE
       ON tbuch
REFERENCING NEW AS neuezeile
            OLD AS altezeile
FOR EACH ROW
BEGIN
   IF INSERTING THEN
   INSERT INTO tbuchlogzeile
   (
      schl, type, wann, wer
     ,buchnrneu ,buchnralt
     ,preisneu  ,preisalt
   ) VALUES
   (
     ( select coalesce(max(schl)+ 1 , 1)
        from tbuchlogzeile )
     , 'I'
     , current_timestamp(9)
     , USER
     , :neuezeile.buchnr
     , NULL
     , :neuezeile.preis
     , NULL
   );
   END IF;
```

```
  IF DELETING THEN
   INSERT INTO tbuchlogzeile
   (  schl, type, wann, wer
     ,buchnrneu, buchnralt
     ,preisneu,  preisalt
    )
    VALUES
   (
     ( select coalesce(max(schl)+ 1 , 1)
        from tbuchlogzeile )
     , 'D'
     , current_timestamp(9)
     , USER
     , NULL
     , :altezeile.buchnr
     , NULL
     , :altezeile.preis
    );
  END IF;
  IF UPDATING THEN
   INSERT INTO tbuchlogzeile
   (  schl ,type ,wann ,wer
     ,buchnrneu ,buchnralt
     ,preisneu ,preisalt
    )
    VALUES
   (
     ( select coalesce(max(schl)+ 1 , 1)
        from tbuchlogzeile )
     , 'U'
     , current_timestamp(9)
     , USER
     , :neuezeile.buchnr
     , :altezeile.buchnr
     , :neuezeile.preis
     , :altezeile.preis
    );
  END IF;
END;
/
SHOW ERRORS;
```

--TEST

```
delete from tbuch;
delete from tbuchlogzeile;
commit;

insert into tbuch
(buchnr, titel, erschj, preis, verlagnr) values
(7777  , 'abcd', 1907  , 12.77, NULL    )
;
insert into tbuch
(buchnr, titel, erschj, preis, verlagnr) values
(8888  , 'abcd', 1908  , 12.88, NULL    )
;
insert into tbuch
(buchnr, titel, erschj, preis, verlagnr) values
(9999  , 'abcd', 1909  , 12.99, NULL    )
;

delete from tbuch where buchnr = 7777
;

Update tbuch set preis = preis + 100.00
;

update tbuch set buchnr = buchnr + 100000
;

set linesize 500;
select schl, type, wer
      , buchnrneu, buchnralt
      , preisneu,  preisalt
      ---- , wann
from tbuchlogzeile
;

spool off;
```

Testprotokoll

```
Tabelle wurde gelöscht.


Tabelle wurde erstellt.


Trigger wurde erstellt.

Keine Fehler.

2 Zeilen wurden gelöscht.


2 Zeilen wurden gelöscht.


Transaktion mit COMMIT abgeschlossen.


1 Zeile wurde erstellt.


1 Zeile wurde erstellt.


1 Zeile wurde erstellt.


1 Zeile wurde gelöscht.


2 Zeilen wurden aktualisiert.


2 Zeilen wurden aktualisiert.
```

SCHL	T	WER	BUCHNRNEU	BUCHNRALT	PREISNEU	PREISALT
1	I	SL01	7777		12,77	
2	I	SL01	8888		12,88	
3	I	SL01	9999		12,99	
4	D	SL01		7777		12,77
5	U	SL01	8888	8888	112,88	12,88
6	U	SL01	9999	9999	112,99	12,99
7	U	SL01	108888	8888	112,88	112,88
8	U	SL01	109999	9999	112,99	112,99

```
8 Zeilen ausgewählt.
```

12.15.3 TRIGGER und SQL Server, ein Beispiel

Mit Hilfe der Trigger können ereignisabhängige Verarbeitungsschritte angestoßen werden. Die Anwendungslogik kann somit in das RDBMS übertragen werden.

```
DROP TABLE Tbuchlog
;
CREATE TABLE Tbuchlog
(
 schl   integer IDENTITY(1,1) PRIMARY KEY
,type   char(1)   not null
,wann   DATETIME2 not null
,wer    varchar(30)   not null
,anzahl integer   not null
)
;
Go
```

FOR INSERT AS

```
--dieser trigger dokumentiert
--fuer ein insert-statement
--die anzahl der eingefuegten zeilen
DROP TRIGGER tbuch_trig1
;
go
CREATE TRIGGER tbuch_trig1
ON tbuch
FOR INSERT AS
BEGIN
DECLARE @anzahlzeilen INTEGER;
DECLARE @type CHAR(1);
SET @type = 'I';
SELECT @anzahlzeilen =
  (SELECT COUNT(*) FROM INSERTED)
INSERT INTO tbuchlog
    (type , wann         ,wer  , anzahl)
VALUES
    (@type, SYSDATETIME(), SYSTEM_USER,
(@anzahlzeilen));
END
;
```

FOR UPDATE AS

```
--dieser trigger dokumentiert
--fuer ein update-statement
--die anzahl der veraenderten zeilen
DROP TRIGGER tbuch_trig2
;
go
CREATE TRIGGER tbuch_trig2
ON tbuch
FOR UPDATE AS
BEGIN
DECLARE @anzahlzeilen INTEGER;
DECLARE @type CHAR(1);
SET @type = 'U';
SELECT @anzahlzeilen =
  (SELECT COUNT(*) FROM INSERTED)
INSERT INTO tbuchlog
    (type , wann         ,wer  , anzahl)
VALUES
    (@type, SYSDATETIME(), SYSTEM_USER,
(@anzahlzeilen));
END
;
```

FOR DELETE AS

```
--dieser trigger dokumentiert
--fuer ein delete-statement
--die anzahl der geloeschten zeilen
DROP TRIGGER tbuch_trig3
;
go
CREATE TRIGGER tbuch_trig3
ON tbuch
FOR DELETE AS
BEGIN
DECLARE @anzahlzeilen INTEGER;
DECLARE @type CHAR(1);
SET @type = 'D';
SELECT @anzahlzeilen =
  (SELECT COUNT(*) FROM DELETED)
INSERT INTO tbuchlog
    (type , wann         ,wer  , anzahl)
VALUES
    (@type, SYSDATETIME(), SYSTEM_USER,
(@anzahlzeilen));
END
;
```

12.16 Benutzerdefinierte Datentypen und Funktionen

distinct type, structured type

Es können eigene Datentypen definiert werden. Nur die implizit und explizit definierten und dem Datentyp zugeordneten Vergleichsoperatoren, Funktionen und Zuordnungsoperatoren werden zugelassen (strong typing).

function, method

Neben den Stored Procedures werden benutzerdefinierte Routinen unterstützt:

- Scalar Functions
- Column Functions
- Table Functions können innerhalb der FROM-Klausel einer SELECT-Anweisung benutzt werden (es wird eine Table zurückgegeben).
- Methods sind nur möglich in Verbindung mit structured types.

Besonders interessant sind Scalar Functions und Column Functions in Verbindung mit distinct types!

sourced, bodied, external

- Eine User-Defined Function kann mit Hilfe einer built-in-function erstellt werden (sourced).
- Eine User-Defined Function kann mit Hilfe einer CREATE-Anweisung erstellt werden (SQL bodied).
- Eine User-Defined Function kann in C, C++ oder JAVA geschrieben werden (external).

Die Beispiele auf den folgenden Seiten wurden mit DB2 getestet.

12.16.1 CREATE DISTINCT TYPE

```
drop distinct type dm;
drop distinct type euro;
create distinct type dm
   as decimal(7,2) with comparisons;
create distinct type euro
   as decimal(7,2) with comparisons;

create table tbuch123
       buchnr integer NOT NULL
      ,preis   decimal(7,2)
      ,preisdm dm
```

```
      ,preiseuro euro
      ,primary key (buchnr)
   );

insert into tbuch123
(buchnr, preis ,preisdm, preiseuro)
values (5, 5.0, 5.0, 2.5);
insert into tbuch123
(buchnr, preis ,preisdm, preiseuro)
values (15, 15.0, 15.0, 7.5);

-- das funktioniert nicht !!!!
-- select * from tbuch123  where preisdm > 50.00;
```

12.16.2 Benutzerdefinierte Datentypen und Konvertierungsfunktion CAST

Beim Arbeiten mit benutzerdefinierten Datentypen wird die Konvertierungsfunktion CAST benötigt.

```
select * from tbuch123
where cast(preisdm as decimal(7,2)) > 10.00;
select * from tbuch123
where preisdm > cast(10.00 as dm);
-- das funktioniert !!!!!
update tbuch123 set preisdm = 50.00;

-- das funktioniert nicht!!!
-- update tbuch123 set preisdm = preisdm  + 111.00;
```

12.16.3 Benutzerdefinierte Datentypen und benutzerdefinierte Funktionen

In Verbindung mit benutzerdefinierten Datentypen sind die benutzerdefinierten Funktionen wichtig.

Beispiel für eine sourced function:

```
create function "+" (  dm , dm )   returns dm
       source   "+" ( decimal() , decimal() );

create function "-" (  dm , dm )   returns dm
       source   "-" ( decimal() , decimal() );



--  das funktioniert immer noch nicht!!!
--  update tbuch123
--  set preisdm = preisdm  + 111.00;

--  jetzt funktioniert das folgende!!!
update tbuch123
set preisdm = preisdm  + cast(111.00 as dm);

select * from tbuch123;
```

12.17 Datendefinition und Transaktion

Achtung:

Eine Anweisung der Datendefinition führt bei Oracle zum COMMIT.

Zitat Anfang

Data definition language (DDL) statements let you to perform these tasks:

- Create, alter, and drop schema objects
- Grant and revoke privileges and roles
- Analyze information on a table, index, or cluster
- Establish auditing options
- Add comments to the data dictionary

The CREATE, ALTER, and DROP commands require exclusive access to the specified object. For example, an ALTER TABLE statement fails if another user has an open transaction on the specified table.

The GRANT, REVOKE, ANALYZE, AUDIT, and COMMENT commands do not require exclusive access to the specified object. For example, you can analyze a table while other users are updating the table.

Oracle Database implicitly commits the current transaction before and after every DDL statement.

Zitat Ende

Mischen Sie nie – mit Ausnahme von temporary tables **– Anweisungen der Datenmanipulation und Anweisungen der Datendefinition.**

12.18 SCHEMA und USER

```
USE dbbuecher01;
go
drop table schemaxx.tabxx01;
drop table schemaxx.tabxx02;
drop table schemaxx.tabxx03;
drop schema schemaxx;
go
CREATE SCHEMA schemaxx
 create table tabxx01
  (schl  integer  not null
  ,daten char(10) not null
  ,primary key (schl)
  )
 create table tabxx02
  (schl  integer  not null
  ,daten char(10) not null
  ,primary key (schl)
  );
go
create table schemaxx.tabxx03
  (schl  integer  not null
  ,daten char(10) not null
  ,primary key (schl)
  );
go
insert into schemaxx.tabxx01
(schl, daten) values (1, 'AAAA');
use dbbuecher01
--select * from tabxx01
select * from schemaxx.tabxx01;
select * from dbbuecher01.schemaxx.tabxx01;
--in MASTER hole den Servernamen SRVNAME
use master
select * from sysservers;
use dbbuecher01
select * from
"S-MGJ-NB\SQL2005".dbbuecher01.schemaxx.tabxx01;
select * from sys.schemas;
select * from information_schema.schemata;
```

SQL Server 2005 trennt die implizite Verbindung zwischen Datenbankbenutzer und Schema.

SQL Server-Onlinedokumentation

Vorteile für Benutzer – Schematrennung

Die Abstraktion von Datenbankbenutzern und Schemas bietet verschiedene Vorteile für Administratoren und Entwickler.

- Über die Mitgliedschaft in Rollen oder in Windows-Gruppen können mehrere Benutzer dasselbe Schema besitzen. Dies erweitert die bekannte Funktionalität dahin gehend, dass Rollen und Gruppen Objekte besitzen können.
- Das Löschen von Datenbankbenutzern ist sehr viel einfacher.
- Für das Löschen eines Datenbankbenutzers ist es nicht mehr notwendig, die im Schema des Benutzers enthaltenen Objekte umzubenennen. Daher müssen auch Anwendungen, die explizit auf im Schema enthaltene Objekte verweisen, nach dem Löschen des die Objekte erstellenden Benutzers nicht mehr überarbeitet und getestet werden.
- Um eine einheitliche Namensauflösung sicherzustellen, können mehrere Benutzer gemeinsam ein einziges Standardschema verwenden.
- Über gemeinsam verwendete Standardschemas können Entwickler gemeinsam genutzte Objekte anstelle im DBO-Schema in einem speziell für eine bestimmte Anwendung erstellten Schema speichern.
- Berechtigungen für Schemas und in Schemas enthaltene Objekte können differenzierter verwaltet werden, als dies in früheren Versionen der Fall war.
- Vollqualifizierte Objektnamen bestehen jetzt aus vier Teilen: `server.database.schema.object`.“

Standardschemas

Neu ist in SQL Server 2005 auch der Begriff "Standardschema", mit dem die Namen von Objekten aufgelöst werden, auf die ohne vollqualifizierte Namen verwiesen wird. In SQL Server 2000 wird zuerst das im Besitz des aufrufenden Datenbankbenutzers befindliche Schema und anschließend das DBO-Schema überprüft. In SQL Server 2005 besitzt jeder Benutzer ein Standardschema, das angibt, welches Schema der Server zuerst sucht, wenn er die Namen von Objekten auflöst. Dieses Standardschema kann mit der Option DEFAULT_SCHEMA von CREATE USER und ALTER USER festgelegt und geändert werden. Wenn DEFAULT_SCHEMA nicht definiert ist, erhält der Datenbankbenutzer das Standardschema DBO.“

Gesamtindex

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

