integrata cegos

SQL - die Sprache: Interaktives Arbeiten mit SQL

Relationale Datenbanken und Datenbankmanagement

Relationale Datenbank

- Tverlag
- Verlagnr Verlag
- ---------- --------------------
- 1111 Forkel

- Tbuch
- buchnr erschj preis verlagnr titel
- ---------- ------ -------- ---------- ----------------------------
- 5 1988 3.50 NULL Ansichten eines Clowns
- 27 NULL 99.99 NULL die Jüdin von Toledo
- 6 1988 20.50 NULL die Blechtrommel
- 7 1989 99.99 NULL der Name der Rose
- 8 1977 0.50 1111 der Butt
- 9 1990 55.00 1111 DB2 fuer Sie
- 11 1990 NULL NULL Elvis in Heidelberg
- 12 1989 NULL NULL a guide to db2
- 18 1989 99.99 NULL Database Systems
- 1 NULL NULL NULL C
- 2 NULL NULL NULL C

- 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
;
```

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
- ;

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

TABLE, VIEW, Daten werden präsentiert!

integrata cegos

- 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!

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

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 Datenbank-managementsystem vorbehalten.

- 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.
- Empfehlung: Vermeiden Sie „Tables ohne Schlüssel“!

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

Relationale Algebra

- Die drei wichtigsten Operationen sind:
  - 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). Das Ergebnis ist wieder eine Table.
  - Die Join-Operation (salopp auch Join bzw. Verbund genannt): Verbinden von zwei Tables auf der Basis gemeinsamer Werte.

- Das Ergebnis einer Operation der relationalen Algebra ist wieder eine Table und präsentiert deshalb nie Duplikate!
- Die Sprache SQL hält sich leider nicht an dieses mathematische Modell und ermöglicht SELECT-Anweisungen, mit deren Hilfe Listen mit Duplikaten generiert werden können!

Was ist eine Projektion?

- 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

- 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.

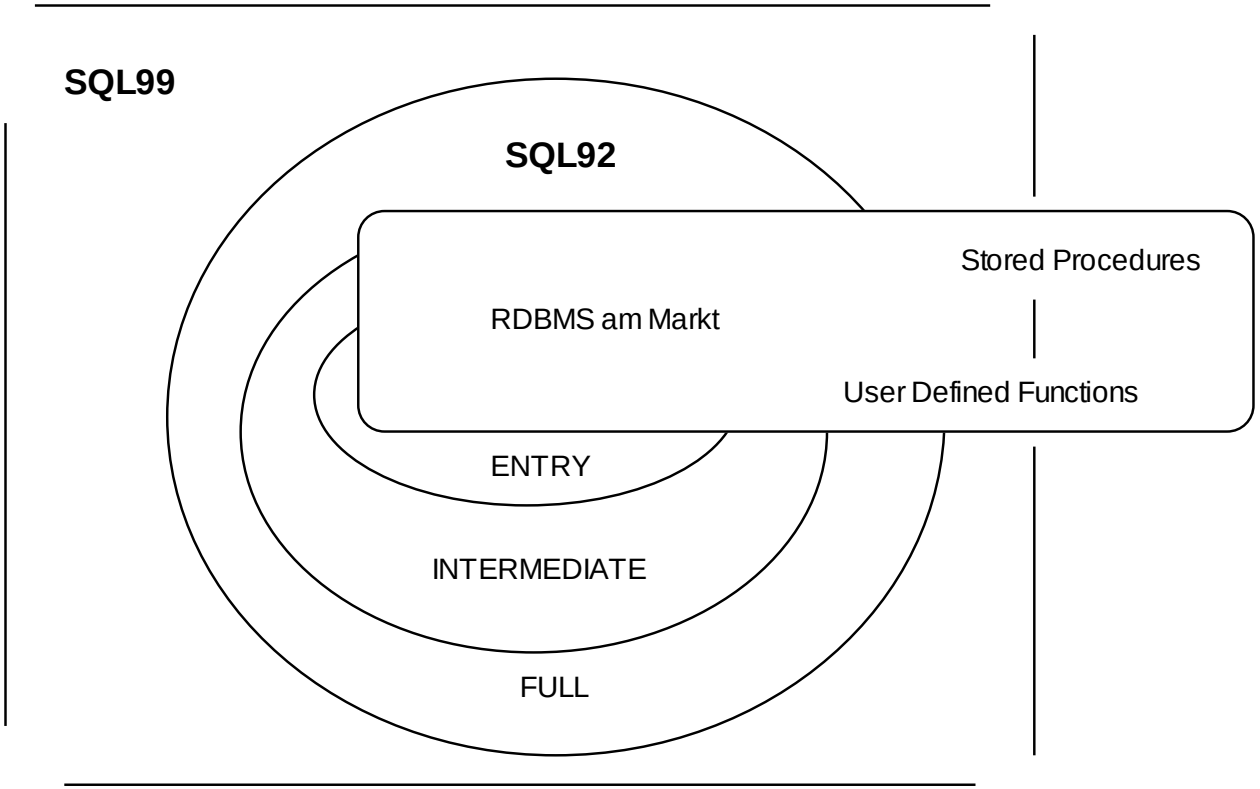
SQL die Sprache

- Datendefinition
- CREATE
- ALTER
- DROP
- Datenmanipulation
- SELECT
- INSERT
- UPDATE
- DELETE
- Datenkontrolle
- GRANT
- REVOKE
- Transaktionsverarbeitung
- COMMIT
- ROLLBACK
- SAVEPOINT
- Der Isolation Level

- Weitere Sprachelemente für die Anwendungsentwicklung

- Proprietäre Sprachelemente

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.

SQL Schnittstellen

- Für die Verwendung von SQL in Verbindung mit Programmiersprachen wie JAVA, COBOL, Assembler usw. gibt es zwei Techniken.
  - Eine herkömmliche Programmierschnitstelle (API Application Programming Interface) erlaubt die direkte Übergabe von SQL-Anweisungen an ein Datenbank-system über Funktionsaufrufe, dabei ist kein Precompiler erforderlich. SQL-An-weisungen, 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.

- 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.

Codd 12 Regeln

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

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

3-Ebenen Architektur für ein Datenbankmanagementsystem

- Konzeptionelle Ebene/Logische Ebene für alle Benutzer
- Interne Ebene/Physische Ebene
- Externe Ebene/Logische Ebene für einen Benutzer
- 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 vorge-nommen werden.
- Logische Datenunabhängigkeit Anwender und Anwendungsprogramme bleiben unbeeinträchtigt, wenn Veränderungen (Erweiterung bzw. Restruktu-rierung) 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-Datenbank-systemen DB2, ORACLE und SQL Server) nur teilweise erfüllt wird.

3-Ebenen-Architektur und relationale Datenbankmanagementsysteme

- Auf der Konzeptionellen Ebene präsentiert die Datenbank z.B. Informationen über Bücher. Das sichtbare Objekt ist eine Base Table Tbuch. In SQL wird eine Base Table mit der CREATE TABLE Anweisung generiert.
- Auf der Internen Ebene können die Zeilen der Base 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 sind die sichtbaren Objekte auch Tables. Diese Tables der externen Ebene werden oft virtuelle Tables (VIEW) genannt. Virtuelle Tables werden aus den Base Tables der konzeptionellen Ebene abgeleitet. In SQL wird eine virtuelle Table mit der CREATE VIEW Anweisung generiert.

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

„Was und wie" – SELECT und der Optimizer

- 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 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.

Client/Server Architektur

- 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 Datenbank-systems – 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).

2

SELECT MIT EINER TABLE

query specification

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

- <search condition> ::=
- (<predicate> OR <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>::=
- = > < <= >= <>

SELECT

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

SELECT

- 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
- ;

SELECT … ORDER BY

- --aufsteigend sortiert nach Erscheinungsjahr und Buchnr

- SELECT Tbuch.Buchnr AS Bnr
- , Tbuch.Erschj AS Erschj
- , Tbuch.Preis AS Preis
- , Tbuch.Verlagnr AS Vnr
- FROM Tbuch
- ORDER BY Erschj, Bnr
- ;
-

- Nur mir Hilfe der Option ORDER BY ist sichergestellt, dass die Daten in der gewünschten Reihenfolge angelistet werden!
- SQL Server und Sybase:
- Die NULL wird bei ASC am Anfang angelistet, bei DESC am Schluss!
- Die Zeichenordnung (collating sequence) einer Zeichenmenge (character set) regelt den Vergleich von Zeichenketten (character strings).

SELECT DISTINCT

- -- die verschiedenen Werte in der Spalte Preis
- SELECT DISTINCT Preis FROM Tbuch
- ;
- -- die verschiedenen Kombinationen in Erschj,Preis
- SELECT DISTINCT Erschj,Preis FROM Tbuch
- ;

SELECT ... WHERE ... IS NULL

- SELECT Tbuch.Buchnr, Tbuch.Erschj
- FROM Tbuch
- WHERE Tbuch.Erschj = 1990
- ;
- Buchnr Erschj
- ---------- ----------
- 9 1990
- 11 1990
- SELECT Tbuch.Buchnr, Tbuch.Erschj
- FROM Tbuch
- WHERE NOT (Erschj = 1990)
- ;
- Buchnr Erschj
- ---------- ----------
- 5 1988
- 6 1988
- 7 1989
- 8 1977
- 12 1989
- 18 1989

- SELECT Tbuch.Buchnr, Tbuch.Erschj
- FROM Tbuch
- WHERE Erschj IS NULL
- ;
- Buchnr Erschj
- ---------- ----------
- 1 NULL
- 2 NULL
- 27 NULL
-
-

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

die Suchbedingung

- 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.

SELECT ... WHERE Suchbedingung

```
-- Buchnr und Titel
-- aller Bücher von 1990

SELECT Buchnr, Titel
FROM Tbuch
WHERE Erschj = 1990
;

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

Buchnr        Titel
-----------   ----------------------------------------
9             DB2 fuer Sie
11            Elvis in Heidelberg
(2 row(s) affected)
```

- Empfohlene Schreibweise mit erstem Qualifier, dem sogenannten Schema:
- SELECT
- tbuch.Buchnr AS Buchnr
- ,tbuch.Titel AS Titel
- FROM schemaxyz.Tbuch Tbuch
- WHERE tbuch.Erschj = 1990
- ;
- SELECT
- abc.Buchnr AS Buchnr
- ,abc.Titel AS Titel
- FROM schemaxyz.Tbuch abc
- WHERE abc.Erschj = 1990
- ;

CHAR oder VARCHAR

- Tverlag.Verlag CHAR(20)
-
- select verlagnr, verlag from tverlag
- where verlag = 'Forkel'
- ;
- select verlagnr, verlag from tverlag
- where verlag = 'Forkel '
- ;
-
- VERLAGNR VERLAG
- ---------- --------------------
- 1111 Forkel

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

- --DB2 bzw. SQL Server mit VARCHAR
- 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
-
-

Dreiwertige Aussagenlogik, TRUE, FALSE, UNKNOWN

integrata cegos

- -- Buchnr, Erschj, Preis 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 .50
- 9 1990 55.00
- 11 1990 NULL
- 12 1989 NULL

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

- AND bindet stärker als OR

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.		

Logische Operatoren AND OR NOT

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
- --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))
- ;

- 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)
- ;

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
- ;

SELECT ... WHERE ... IN

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

SELECT ... WHERE ... LIKE

- -- Titel die mit dem Buchstaben D beginnen
- SELECT Buchnr, Titel FROM Tbuch
- WHERE Titel LIKE 'D%'
- ;
- -- Titel die ab dem zweiten Zeichen
- -- den String ABC enthalten
- SELECT Buchnr, titel FROM Tbuch
- WHERE Titel LIKE '_ABC%';
-
- Das Ergebnis ist natürlich abhängig von der Eigenschaft case-sensitive bzw. case-insensitive der Zeichenordnung.

Nested Table Expression bzw. Derived Table

```
SELECT
 tbuch.buchnr       AS  buchnr
,tbuch.preis        AS  preisnetto
,tbuch.preis*1.6    AS  preisbrutto
FROM tbuch
WHERE (tbuch.preis * 1.6) > 100.00
;

SELECT
 zwi.buchnr         AS  buchnr
,zwi.preisnetto     AS  preisnetto
,zwi.preisbrutto    AS  preisbrutto
FROM
(
SELECT
 tbuch.buchnr       AS  buchnr
,tbuch.preis        AS  preisnetto
,tbuch.preis * 1.6  AS  preisbrutto
FROM tbuch
) zwi
WHERE zwi.preisbrutto > 100.00
;
```

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
-
- -- Wie viel Bücher haben kein Erscheinungsjahr?

- 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!“

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
- (5 row(s) affected)
-
-

- 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
- (9 row(s) affected)

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

integrata cegos

- 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(*) AS Anzbue
- FROM Tbuch GROUP BY Erschj
-) Zwitab
- WHERE Zwitab.Anzbue = 2
- ;
- Erschj Mapr Anzbue
- ------ --------- -----------
- 1988 20.50 2
- 1990 55.00 2
-
- (2 row(s) affected)

- alternative Formulierung:
- SELECT Erschj AS Erschj
- , MAX(Preis) AS Mapr
- , COUNT(*) AS Anzbue
- FROM Tbuch GROUP BY Erschj
- HAVING COUNT(*) = 2
- ;

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
;
```

“NULL-Werte sind gleich - NULL-Werte sind nicht gleich“

- 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

„NULL ist etwas anderes als UNKNOWN!“

	Name	Antwort	
	Otto	wahr	
	Emil	falsch	
	Fritz	unbekannt	
	Franz	NULL	

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

3

SKALARE OPERATOREN UND FUNKTIONEN, EXPRESSION, CASE, CAST

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)
```

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('neu: ' , Titel)
-
- LOWER('der Butt')
- TRIM (' Test ')
- UPPER('der Butt')
- USER

- 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).
- 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 SUBSTRING(Titel, 1, 1) ='C'
- ;
- SELECT * from Tbuch WHERE LOWER(Titel) = 'der butt'
- ;
- SELECT * FROM Tautor WHERE YEAR(Geburtsdatum)=2012
- ;

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.

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
;
```

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
-

- CAST AS CHAR und “eine fette Wanze“
- CAST AS CHAR bzw. CONVERT 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)

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)
```

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: SUBSTRING DATALENGTH
- Character-Semantik: SUBSTRING LEN

Syntax und Semantik

- 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).

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

Skalare Operatoren und Funktionen, der SQL 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

Skalare Operatoren und Funktionen, der SQL Standard und die Dialekte

integrata cegos

SQL Standard	DB2	ORACLE	SQL Server
OVERLAY			
POSITION ***)	X LOCATE POSSTR	INSTR	CHARINDEX PATINDEX
POWER	X	X	X
SESSION_USER			
SQRT	X	X	X
SUBSTRING ***) FROM FOR	SUBSTRING SUBSTRING (Titel, 1, 5, OCTETS) mit Byte-Semantik SUBSTR	SUBSTR SUBSTRB mit Byte-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

Zeichenkettenfunktionen im Detail

- SQL Server LEN bzw. DATALENGTH
- 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.“

Mathematische Funktionen im Detail

- 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

Mathematische Funktionen im Detail

- 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

4

DATENBANKDESIGN

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 Design-fragen, 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 Integritäts-bedingungen auf einfache Art und Weise durchzusetzen.

Konzeptionelle Ebene, Integritätsbedingungen und Redundanzen

- Externe Ebene/Logische Ebene für einen Benutzer – Redundanz erwünscht
-
- Konzeptionelle Ebene/Logische Ebene für alle Benutzer – möglichst wenig Redundanz
-
- Interne Ebene/Physische Ebene – Redundanz notwendig
-

Semantische Datenmodellierung, Entity/Relationship Diagramm

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

  - Entität (entity)
  - Entitätstyp (entity type)
  - Entitätsmenge
  - Eigenschaft (property, Merkmal, Attribut)
  - Identität (identity)
  - Beziehung (relationship)
  - Subtyp (subtype)

Präsentation von Daten in Form von Tables

integrata cegos

```
Verlagnr   Verlag
---------- ----------
      1111  Forkel

buchnr      erschj preis      verlagnr    titel
----------- ------ ---------- ----------- ----------------------
1           NULL   NULL       NULL        C
2           NULL   NULL       NULL        C
5           1988   3.50       NULL        Ansichten eines Clowns
6           1988   20.50      NULL        die Blechtrommel
7           1989   99.99      NULL        der Name der Rose
8           1977   0.50       1111        der Butt
9           1990   55.00      1111        DB2 fuer Sie
11          1990   NULL       NULL        Elvis in Heidelberg
12          1989   NULL       NULL        a guide to db2
18          1989   99.99      NULL        Database Systems
            NULL   99.99      NULL        die Jüdin von Toledo

    BUCHNR ISBN           LFDNR
---------- ---------- ----------
         8 3472864303          1
         8 34728643yx          2
        12 0201501139          1
```

- 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

- 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

Konzeptionelles Strukturdiagramm

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

Integritätsbedingungen

- Primärschlüssel, Alternativschlüssel, Fremdschlüssel

- 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.

- Integritätsbedingungen des Relationalen Modells
- 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.

- Fachliche Integritätsbedingungen

ON DELETE CASCADE

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

ON DELETE NO ACTION

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

Referentielle Aktionen in der Delete-Regel für den Fremdschlüssel

integrata cegos

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.

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.

Erstellen der Tables und der Constraints

- 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)
- ;
- ON DELETE NO ACTION ist der Default.
- 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.
-

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 sollte die Performance einer Anwendung nicht negativ beeinflussen.

- 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.

Selbst-referenzierende Table

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

```
DROP TABLE emp
;
CREATE TABLE emp
   (
    empno         INTEGER      NOT NULL
   ,mgr           INTEGER
   ,PRIMARY KEY (empno)
   )
;
ALTER TABLE emp ADD CONSTRAINT
                  FK_emp_emp
                  FOREIGN KEY (mgr)
                  REFERENCES emp(empno)
                  --ON delete no action
                  ON delete cascade       --nicht möglich
                  --ON delete set null    --nicht möglich
                  --ON delete set default --nicht möglich
;
```

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

Empno	Mgr
100	NULL
200	100
300	200
400	300

DELETE FROM Emp WHERE Empno > 200;

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
- 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

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

- Wir sagen, dass Stadt funktional abhängig ist von Stadtnr.

- Diese funktionale Abhängigkeit (Functional Dependency) ist eine Integritäts-bedingung 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.

Der Normalisierungsprozess, Integritätsbedingungen und Redundanz

integrata cegos

- 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.

Normalformen

- Jede Table hat wenigstens einen Schlüsselkandidaten und damit einen Primärschlüssel!
- 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 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 Primär-schlüssel aus genau einer Spalte besteht, ist immer in 2. Normalform.
- Eine Table befindet sich in 3. Normalform, wenn sie in 2. Normalform ist und zwischen nicht zum Primärschlüssel gehörenden Spalten keine funktionale Abhängigkeiten bestehen.

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“.

„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 Datenbank-managementsystems für die Implementierung „unzureichend“?
- Die Message kann nur lauten: das Mapping zwischen der konzeptionellen und der internen Ebene des Produkts ist unzureichend!

Denormalisierung

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

Boyce/Codd Normalform und Schlüsselkandidaten

integrata cegos

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.

4. Normalform und mehrwertige Abhängigkeit

integrata cegos

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

- 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.
- 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!

Eingebettete mehrwertige Abhängigkeit

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

- Die eingebettete mehrwertige Abhängigkeit kann nicht auf einfache Weise durchgesetzt werden.

5. Normalform und Verbundabhängigkeiten

- 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

Bücher, Autoren und Verlage

integrata cegos

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

Bücher, Autoren und Verlage

integrata cegos

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

Bücher, Autoren und Verlage

integrata cegos

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

query specification

- SELECT [ALL | DISTINCT] ...
  FROM <table reference>
- ,<table reference>
- , ...
  WHERE <search condition>
  GROUP BY ...
  HAVING <search condition>

Kartesisches Produkt

- SELECT * FROM Tbuch, Tisbn
- ;

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
-

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
;
```

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
;
```

JOIN und GROUP BY

- Variante 1:
- SELECT
- Tbuch.Buchnr, Tbuch.Titel, XX.Sumbuch
- FROM
- Tbuch
- INNER JOIN
- (SELECT Tvautor.Buchnr AS Buchnr
- ,SUM(Praemie) AS Sumbuch
- FROM Tvautor GROUP BY Buchnr
-) XX
- ON Tbuch.Buchnr = XX.Buchnr
- ;
- Variante 2:
- SELECT
- Tbuch.Buchnr, Tbuch.Titel, SUM(Praemie) AS Sumbuch
- FROM
- Tbuch INNER JOIN Tvautor
- ON Tbuch.Buchnr = Tvautor.Buchnr
- GROUP BY Tbuch.Buchnr, Tbuch.Titel
- ;

Join einer Table mit sich selbst

- 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
- ;

Der Left Outer Natural-Join

- SELECT Tbuch.Buchnr, Tbuch.Erschj, Tisbn.ISBN
- FROM
- (
- Tbuch LEFT OUTER JOIN Tisbn
- ON Tbuch.Buchnr = Tisbn.Buchnr
-)
- ;

- --Beispiel WHERE links
- SELECT
- Tbuch.Buchnr AS Buchnr
- ,Erschj AS Erschj
- ,Tisbn.Lfdnr AS Lfdnri
- ,Tisbn.Isbn AS Isbn
- FROM
- TBUCH LEFT OUTER JOIN TISBN
- ON TBUCH.Buchnr = TISBN.Buchnr
- WHERE tbuch.erschj = 1977
- ORDER BY Buchnr, Lfdnri
- ;
- Buchnr Erschj Lfdnri Isbn
- ---------- ------ ------ ----------
- 8 1977 1 3472864303
- 8 1977 2 34728643yx

```
--Beispiel AND links
SELECT
 Tbuch.Buchnr                AS Buchnr
,Erschj                      AS Erschj
,Tisbn.Lfdnr                 AS Lfdnri
,Tisbn.Isbn                  AS Isbn
FROM
    TBUCH LEFT OUTER JOIN TISBN
    ON TBUCH.Buchnr = TISBN.Buchnr
AND tbuch.erschj = 1977
ORDER BY Buchnr, Lfdnri
;
Buchnr      Erschj Lfdnri Isbn
----------- ------ ------ ----------
1           NULL   NULL   NULL
2           NULL   NULL   NULL
5           1988   NULL   NULL
6           1988   NULL   NULL
7           1989   NULL   NULL
8           1977   1      3472864303
8           1977   2      34728643yx
9           1990   NULL   NULL
11          1990   NULL   NULL
12          1989   NULL   NULL
18          1989   NULL   NULL
27          NULL   NULL   NULL
```

DER LEFT OUTER NATURAL-JOIN UND MEHR ALS 2 TABLES

integrata cegos

- 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
-)

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.

Der FULL Outer Natural-Join und COALESCE

- 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-/Primärschlüssel-beziehung formuliert ist!

```
--(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
)
;
```

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
- ;

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 Subquery mit IN etwas genauer betrachtet

integrata cegos

```
--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
--es gibt Praemie<>10
--es gibt nicht Praemie<>10
```

IN, EXISTS, JOIN, Varianten des Codings

- 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
        )
;
```

- --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
- ;

Ein guter Optimizer liefert

- 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
-)
- ;

SQL und der gesunde Menschenverstand, >ALL bzw. >MAX

- Bücher, die teurer sind als jedes Buch des Jahres 2000!
- Teurer als alle Bücher von 2000.
- Teurer als das teuerste Buch von 2000.
- Es gibt kein Buch von 2000 das teurer ist.
- Die folgenden Select-Anweisungen liefern 11 Zeilen bzw. 0 Zeilen bzw. 7 Zeilen 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
    )
;
```

UNION, INTERSECT, EXCEPT

- 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
```

- ;

```
        select buchnr from  tvautor
UNION
        select buchnr from  tisbn
;


        select buchnr from tvautor
INTERSECT
        select buchnr from tisbn
;


        select buchnr from tvautor
EXCEPT
        select buchnr from tisbn
,
```

Common Table Expression

```
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
;
```

Division

- Wer kann wenigstens ein Fahrzeug fahren und alle Fahrzeuge des Fuhrparks und auch vielleicht noch andere?
- SQL-Implementierung mit NOT EXISTS
-

```
SELECT DISTINCT AB.Name FROM Tfahrer AB
WHERE NOT EXISTS
      (SELECT * FROM Tfahrzeug B
       WHERE NOT EXISTS
                  (SELECT * FROM Tfahrer abab
                  WHERE abab.Name      =        AB.Name
                  AND   abab.Fahrzeug  =        B.Fahrzeug)
      )
;
```

- SQL-Implementierung mit LEFT OUTER JOIN und GROUP BY
- SELECT Tfahrer.Name
- FROM Tfahrer LEFT OUTER JOIN Tfahrzeug
- ON Tfahrer.Fahrzeug = Tfahrzeug.Fahrzeug
- GROUP BY Name
- HAVING COUNT(DISTINCT Tfahrzeug.fahrzeug)
- = (SELECT COUNT(*) FROM Tfahrzeug)
- ;
- Achtung: Sofern der Fuhrpark leer ist, werden alle Fahrer die wenigstens ein Fahrzeug fahren angelistet.
-

Quota Query „Die zwei teuersten Bücher“

- preis buchnr
- ---------- ----------
- 99.99 7
- 99.99 18
- 99.99 27
- 55.00 9
- 20.50 6
- 3.50 5
- 0.50 8
- select tbuch.preis as preis
- ,tbuch.buchnr as buchnr
- from tbuch
- where preis is not null
- and
- (select count(*) from tbuch aaa
- where aaa.preis > tbuch.preis) < 2
- order by preis desc, buchnr desc
- ;
- preis buchnr
- ---------- ----------
- 99.99 27
- 99.99 18
- 99.99 7

- Vergleichen Sie dazu bitte auch die SQL Server proprietäre Implementierung mit TOP 2 WITH TIES und die dabei nötige Hilfskonstruktion wegen der gewünschten Sortierung preis desc, buchnr desc.
- select zwi.preis as preis, zwi.buchnr as buchnr
- from
- (
- Select TOP 2 WITH TIES
- Tbuch.preis, tbuch.buchnr
- from tbuch
- where tbuch.preis is not null
- order by tbuch.preis DESC
-)zwi
- order by preis desc, buchnr desc
- ;
-

Rekursive Query

- SELECT * FROM Tfeudal
- order by pl, dl, steuer, dependent
- ;
- parent dependent pl dl steuer
- ---------- ---------- ------ ------ ------------
- könig fürst1 1 2 1000
- könig fürst2 1 2 2000
- könig fürst3 1 2 3000
- fürst1 ritter1 2 3 1500
- fürst3 ritter2 2 3 3500
- fürst2 knecht4 2 6 2500
- ritter1 bürger1 3 4 2001
- ritter1 bürger2 3 4 2002
- ritter2 bürger3 3 4 3502
- bürger1 bauer1 4 5 1501
- bürger1 bauer2 4 5 1502
- bürger3 bauer3 4 5 3503
- bürger2 knecht3 4 6 2503
- bauer1 knecht1 5 6 1601
- bauer2 knecht2 5 6 1602

Rekursive Query

- „alle, die direkt oder indirekt an Fürst1 Hirse abliefern?“
- WITH
- tab123x AS
- (
- (SELECT tfeudal.parent as parent
- , tfeudal.dependent as dependent
- , tfeudal.pl as pl
- , tfeudal.dl as dl
- , tfeudal.steuer as steuer
- from Tfeudal where tfeudal.parent = 'fürst1'
-)
- UNION ALL
- (select Tfeudal.parent as parent
- ,Tfeudal.dependent as dependent
- ,Tfeudal.pl as pl
- ,Tfeudal.dl as dl
- ,Tfeudal.steuer as steuer
- from tab123x inner join Tfeudal
- on tab123x.dependent = Tfeudal.parent
-)
-)
- select *
- from tab123x
- order by pl, dl, steuer, dependent
- ;

- Erstellen von Testdaten mit Hilfe einer Rekursiven Query
- Eine Table mit Einheitsintervallen ist nötig, hole alle Einheitsintervalle vom Anfang der Zeit bis zum Ende der Zeit (anfang=1 ende=100) mit Hilfe einer Rekursiven Query (getestet mit SQL Server bzw. DB2).

```
  WITH
  tabxx (von, bis) AS
   (
        SELECT 1,1   -- FROM sysibm.sysdummy1
      UNION ALL
        SELECT  von+1 , bis+1
        FROM tabxx WHERE von < (1000)
    )
SELECT
  tabxx.von as von
 ,tabxx.bis as bis
FROM tabxx
;
```

```
VON         BIS
----------- -----------
          1           1
          2           2
          3           3
        ...         ...
        100         100
```

Transformationen GROUP BY HAVING und Subqueries

- 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 AS 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 AS Rx
- WHERE Rx.A = R.A) comp scalar;

Die Summarize-Operation

- Die Summarize-Operation ist die Verallgemeinerung der „Verdichtungsoperation mit GROUP BY“.
- --Variante-1 mit LEFT OUTER JOIN
- select
- bb.buchnr AS buchnr
- ,count(aa.autornr) AS countau
- ,count(aa.praemie) AS countpr
- ,COALESCE (sum(aa.praemie) , 0.000) AS sumpr
- ,max(aa.praemie) AS maxpr
- ,min(aa.praemie) AS minpr
- ,avg(aa.praemie) AS avgpr
- from Tbuch bb left outer join tvautor aa
- on bb.buchnr = aa.buchnr
- group by bb.buchnr
- ;

- Die Summarize-Operation ist die Verallgemeinerung der „Verdichtungsoperation mit GROUP BY“.
- --Beispiel nur die Buchnr aus Tvautor, mit ausführlichem Coding:
- select
- bb.buchnr AS buchnr
- ,count(aa.autornr) AS countau
- ,count(aa.praemie) AS countpr
- ,COALESCE(sum(aa.praemie) , 0.000) AS sumpr
- from (select distinct xx.buchnr from tvautor xx)
- bb left outer join tvautor aa
- on bb.buchnr = aa.buchnr
- group by bb.buchnr
- ;
- --Beispiel nur die Buchnr aus Tvautor, mit vereinfachtem Coding:
- select
- tvautor.buchnr as buchnr
- ,count(tvautor.autornr) as countau
- ,count(tvautor.praemie) AS countpr
- ,COALESCE(sum(tvautor.praemie) , 0.000) AS sumpr
- from Tvautor
- GROUP by tvautor.buchnr
- ;

UNIQUE …

- 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
  ))

6

INSERT, UPDATE, DELETE, MERGE, TRANSAKTIONSVERARBEITUNG, COMMIT UND ROLLBACK

Was ist eine Transaktion?

- 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.

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
;
```

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
```

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.

- Ein DELETE in einer Primärschlüsseltabelle wird möglicherweise abgewiesen bzw. hat möglicherweise Auswirkungen auf andere Tabellen. Vergleichen Sie bitte dazu die Diskussion über Referentielle Integrität!
-

- DELETE und nichtkorrelierte Subquery mit NOT IN
- 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
-);

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)
- ;
- Die Eigenschaft Atomicity/Unteilbarkeit ist beim MERGE vom System garantiert!

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;

- Im Autocommitmodus
- ist jede einzelne SQL-Anweisung eine Transaktion,
- wird für jede erfolgreich ausgeführte SQL-Anweisung ein Commit durchgeführt.
-
-

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;
-
- --auf003
- update tbuch set buchnr = 55 where buchnr = 5;

- --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;

INSERT, UPDATE, MERGE und Subqueries, UPDATE FROM

- 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

Rückgabe von Daten bei INSERT, UPDATE, DELETE

- SQL Server
- 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)
- ;

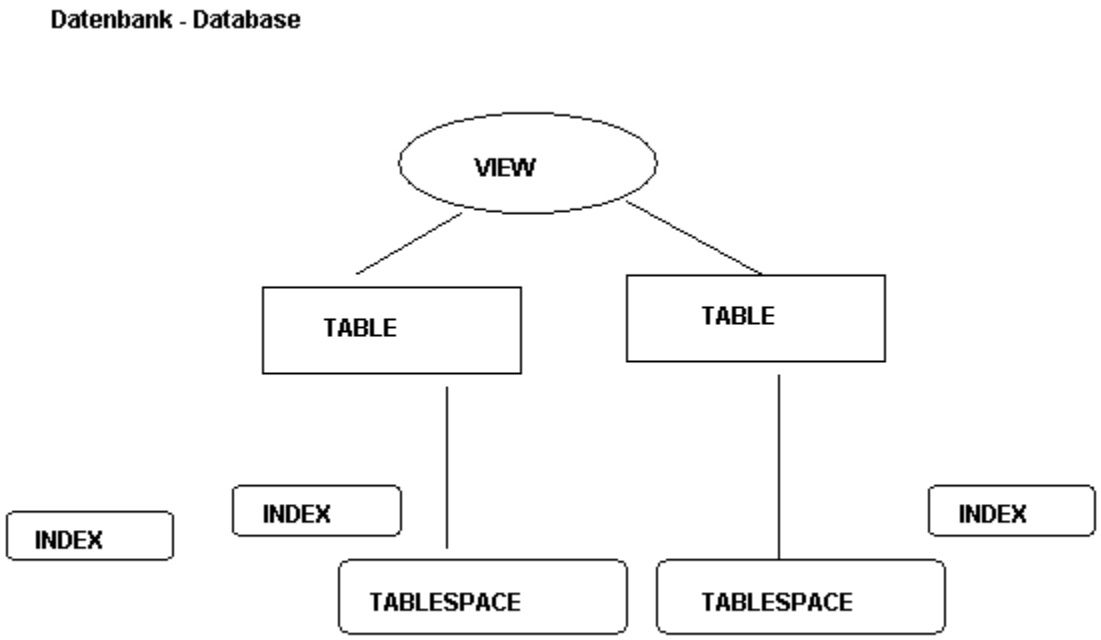
- Oracle

- DB2

7

DATENDEFINITION VIEW, DER KATALOG, USER UND ROLE

Externe Ebene und VIEW



CREATE VIEW

```
CREATE VIEW V9091
AS
        SELECT
          Erschj      AS jahr
        , Titel       AS titel
        , Buchnr      AS buchnr
        FROM   Tbuch
        WHERE  Erschj = 1990 OR  Erschj = 1991
;
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
```

VIEW und INSERT, UPDATE, DELETE

- CREATE VIEW V9091
- AS
- SELECT
- Erschj AS jahr
- , Titel AS titel
- , Buchnr AS buchnr
- FROM Tbuch
- WHERE Erschj = 1990 OR Erschj = 1991
-
- INSERT über eine VIEW ist nicht möglich, wenn die verborgenen Spalten mit NOT NULL
- definiert sind.
- Ein INSERT mit folgenden Daten ist möglich!

- insert into V9091
- (jahr, titel , buchnr)
- values
- (1992, 'das verborgene Buch', 123);

CREATE VIEW … WITH CHECK OPTION

- CREATE VIEW vjatibu
- AS
- SELECT
- Erschj AS jahr
- , Titel AS titel
- , Buchnr AS buchnr
- FROM Tbuch
- WHERE Erschj = 1990 OR 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

VIEW Verwendung und Vorteile

- Datenschutz
- Der Endanwender kann nur die Daten sehen, für die er auf Grund der VIEWdefinition berechtigt ist (Beispiel: alle Personalstammsätze mit Gehältern unter 5000 DM, ... über 5000 DM):
- GRANT SELECT ON Vjatibu TO PUBLIC
- GRANT SELECT ON Vjatibu TO otto
  - 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).

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
     , SUBSTRING(Autor1, 1, 10)  AS Autor
     , Buchnr                 AS Buchnr
     , SUBSTRING(Titel,  1, 6)   AS Titel
FROM buchautor1
WHERE UPPER (Autor1)          LIKE 'BOELL%'
OR    LOWER (Titel)           LIKE '%der%'
;
```

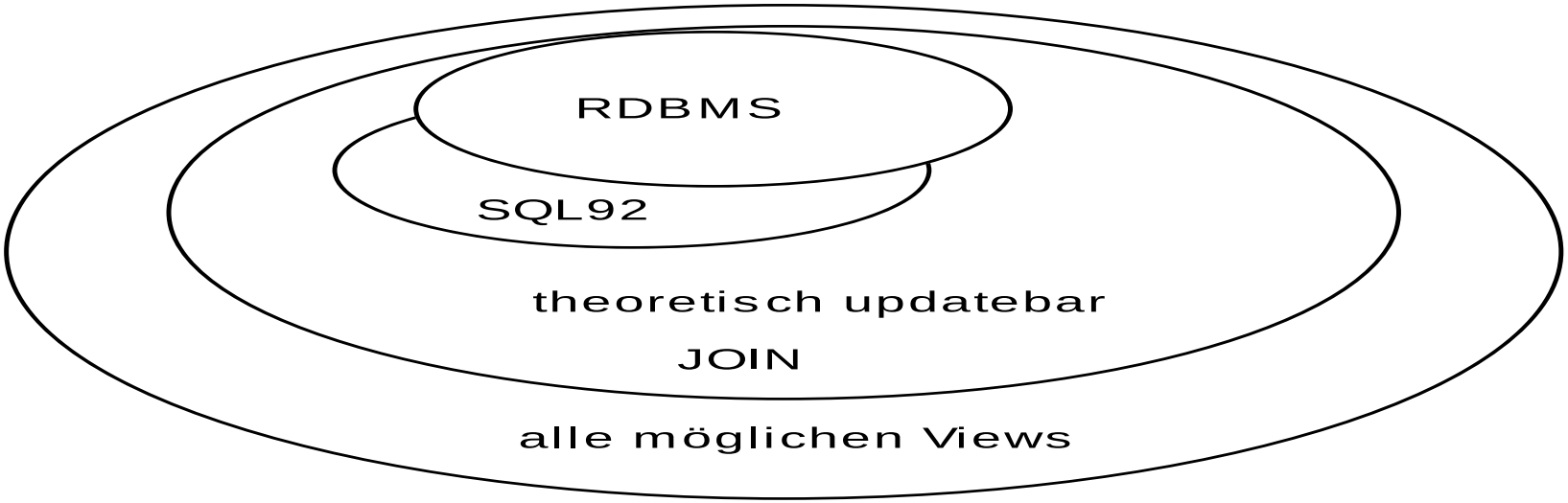
VIEW und logische Datenunabhängigkeit

- CREATE VIEW Vpreistitel
- AS
- SELECT Buchnr, Preis, Titel
- FROM Tbuch
- Redesign der konzeptionellen Ebene
- 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
-
- SQL Server unterstützt diesen UPDATE!

VIEW und INSTEAD OFF-Trigger

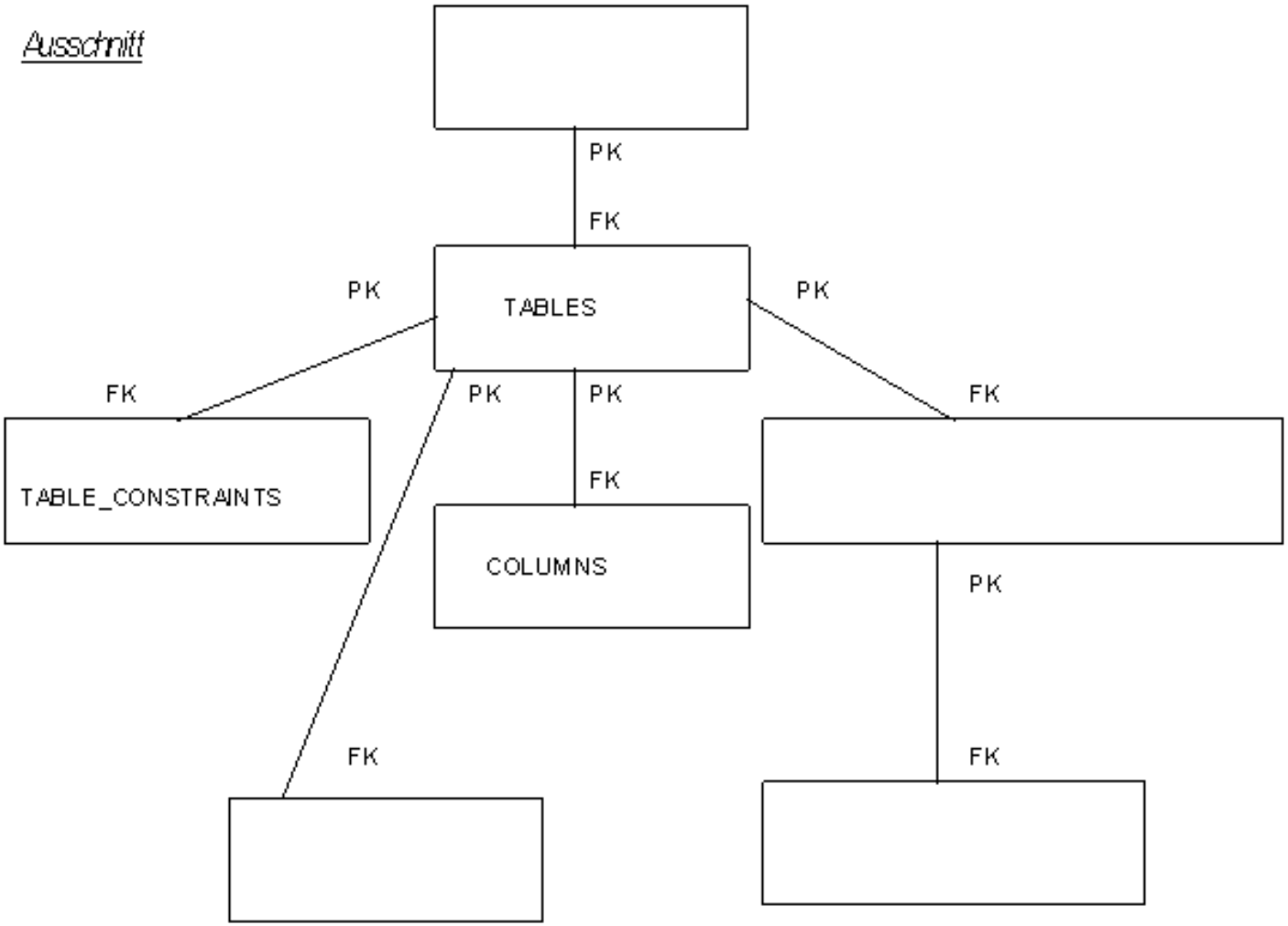
-
- DB2
- Oracle
- SQL Server

Einschränkungen beim Update einer VIEW



Der Katalog

- *Ausschnitt*



- 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 SQL Standard nicht definiert).
- Der Katalog wird zum Beispiel verändert durch
  CREATE, ALTER, DROP von Objekten
  GRANT , REVOKE von Privilegien
- Über SQL-Zugriffe sind Abfragen möglich!
- Der SQL Standard spezifiziert die interne Struktur des Katalogs nicht! Er fordert aber, dass bestimmte VIEWs vorhanden sein müssen:
- DOMAINS TABLES VIEWS COLUMNS DOMAIN_CONSTRAINTS TABLE_CONSTRAINTS REFERENTIAL_CONSTRAINTS CHECK_CONSTRAINTS KEY_COLUMN_USAGE ASSERTIONS ...
- SQL Server 2005 unterstützt diese VIEWs
- select *
- from information_schema.tables
- select *
- from information_schema.columns
- select *
- from information_schema.table_constraints

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.

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

9

LÖSUNGEN INTERAKTIVE SQL

10

DATUMS- UND ZEITANGABEN

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.“

DATENTYP 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)
-)
- ;

- 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;

Welche Autoren haben heute Geburtstag?

- Beispiel DB2
- select
- autornr as autornr
- ,char(geburtsdatum, eur) as geburtsdatum
- from tautor
- where month(geburtsdatum) = month(current date)
- and day(geburtsdatum) = day(current date)
- ;
- Beispiel Oracle
- select autornr, geburtsdatum
- from tautor where
- extract (month from geburtsdatum)
- =extract (month from sysdate)
- and
- extract (day from geburtsdatum)
- =extract (day from sysdate)
- ;
-

- Beispiel SQL Server
- select autornr, geburtsdatum from tautor
- where datepart(mm, geburtsdatum)
- = datepart(mm, getdate())
- and datepart(dd, geburtsdatum)
- = datepart(dd, getdate())
- ;

DATE und CURRENT_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) as heute
- FROM Dual

Datumsarithmetik

- “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 (ganze Zahl mit Vorzeichen) der angegebenen datepart-Begrenzungen zurück, die zwischen den angegebenen Werten für startdate und enddate überschritten wurden.“
-

- DB2 Date Duration
- Mit Werten des Datentyps DATE kann gerechnet werden. Manchmal liefert „das Rechnen“ aber nicht das gewünschte Ergebnis.
- Wie viele Monate und Tage sind zwischen dem 2.1.2012 und dem 1.3.2012 vergangen? 1 Monat und 30 Tage oder 1 Monat und 28 Tage?

Date subtraction: result = date1 – date2

If DAY(DATE2) <= DAY(DATE1) then DAY(RESULT) = DAY(DATE1) – DAY(DATE2)
If DAY(DATE2) > DAY(DATE1) then DAY(RESULT) = N + DAY(DATE1) – DAY(DATE2) where N = the last day of MONTH(DATE2). MONTH(DATE2) is then incremented by 1.
If MONTH(DATE2) <= MONTH(DATE1) then MONTH(RESULT) = MONTH(DATE1) – MONTH(DATE2)
If MONTH(DATE2) > MONTH(DATE1) then MONTH(RESULT) = 12 + MONTH(DATE1) – MONTH(DATE2) and YEAR(DATE2) is incremented by 1.
YEAR(RESULT) = YEAR(DATE1) – YEAR(DATE2)

Wie viele Tage sind vergangen?

integrata cegos

```
--DB2
--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
```

Wie viele Tage sind vergangen?

```
--Oracle
--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            as von
,bisopen        as bisopen
,(bisopen – von) as tageinsgesamt
From tabcloope
;
VON               BISOPEN                TAGEINSGESAMT
------------------- ------------------------ -----------------
01.01.0001         31.12.9999              3652060
```

Wie viele Tage sind vergangen?

```
--SQL Server
--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
```

Wie viele Monate sind vergangen?

- DB2
- Oracle
- SQL Server

Ein Monat später?

- DB2
- Oracle
- SQL Server

Wie viele Jahre, Monate, Tage sind vergangen?

- DB2
- Oracle
- SQL Server

SQL Standard

- Die folgenden Seiten sind – was die Darstellung des SQL-Standards betrifft – eine knappe Zusammenfassung des Kapitels 17 "Datums- und Zeitangaben" aus dem Buch von Chris J. Date und Hugh Darwen, der deutsche Ausgabe des amerikanischen Klassikers. Der interessierte Leser wird auf dieses Buch verwiesen.
- SQL – Der Standard SQL/92 mit den Erweiterungen CLI und PSM
  Chris J. Date Hugh Darwen
  Addison Wesley Longman GmbH, 1998

11

DATENDEFINITION DATABASE, TABLE, DATENTYP, DATENINTEGRITÄT

CREATE TABLE

- 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
                FK_Tbuch_Tverlag
                FOREIGN KEY (Verlagnr)
                REFERENCES Tverlag(Verlagnr)
                ON DELETE NO ACTION
                ON UPDATE NO ACTION
;
```

PRIMARY KEY – FOREIGN KEY – CONSTRAINT – UNIQUE

integrata cegos

```
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
              ON UPDATE NO ACTION
  )
;
```

- CREATE TABLE Tautor
- (
- Autornr INTEGER NOT NULL
- ,Autor VARCHAR(240) NOT NULL
- ,Geburtsdatum DATE
- ,PRIMARY KEY (Autornr)
-)
- ;
-

- 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
- ON UPDATE NO ACTION
- ,FOREIGN KEY (Autornr)
- REFERENCES Tautor (Autornr)
- ON DELETE NO ACTION
- ON UPDATE NO ACTION
-)
- ;
- Eine „Table ohne Schlüssel“ die Duplikate zulässt ist nach unserer Ansicht ein sinnloses Konstrukt und von keinerlei Relevanz für die betriebliche Praxis.
  SQL erlaubt aber Tables ohne Schlüssel!

TRUNCATE TABLE

- Mit der TRUNCATE-Anweisung können unter gewissen Voraussetzungen alle Zeilen aus einer Table sehr effizient entfernt werden

- TRUNCATE TABLE Tbuch
- ;

- DB2

- Oracle
- TRUNCATE beendet die aktuelle Transaktion.

- SQL Server
- Nach TRUNCATE TABLE ist bei SQL Server ein ROLLBACK möglich!

DROP TABLE

- Entfernt eine Table sowie alle Daten, Indizes, Trigger, Einschränkungen und Berechtigungen für diese Table.

- DROP TABLE Tbuch
- ;

- DB2

- Oracle
- DROP beendet die aktuelle Transaktion.

- SQL Server

temporäre TABLE

- DB2
- Oracle
- SQL Server

temporäre TABLE
SQL Server Beispiel

integrata cegos

- 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 dbbuecher
- GO
- drop table #tbuch
- create table #tbuch
- (buchnr integer not null
- ,titel varchar(127)
- ,primary key (buchnr)
-)
- --globale temporäre table ##tablename
- use dbbuecher
- go
- drop table ##tbuch
- create table ##tbuch
- (buchnr integer not null
- ,titel varchar(127)
- ,primary key (buchnr)
-)
- go

- 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

Datentypen

- DB2
- Oracle
- SQL Server

Datenintegrität, ein Überblick

- 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

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 ´???´

ALTER TABLE und ADD CONSTRAINT … CHECK

integrata cegos

- ALTER TABLE Tbuch
- DROP CONSTRAINT Conpreise
- ;
-
- ALTER TABLE Tbuch
- ADD CONSTRAINT Conpreise
- CHECK (TBUCH.Preis >= 0.00)
- ;
-
- UPDATE Tbuch SET PREIS = -9999.9
- 9;

- Der Integritätsbedingung kann explizit ein Name gegeben werden.
- Alle Daten in der Table müssen der Bedingung genügen: für jede Zeile in der Table muss die Bedingung TRUE oder UNKNOWN sein.

ALTER TABLE, UNIQUE INDEX und Schlüsselkandidat

- Ändert eine Tabellendefinition durch Ändern, Hinzufügen oder Löschen von Spalten und Einschränkungen oder durch Deaktivieren oder Aktivieren von Einschränkungen und Triggern.
- drop table tisbn
- ;
- CREATE TABLE Tisbn
- (
- Buchnr INTEGER NOT NULL
- ,Isbn CHAR (10) NOT NULL
- ,Lfdnr DECIMAL (1) NOT NULL
-)
- ;
- CREATE UNIQUE INDEX Xisbn1 on Tisbn(Isbn)
- ;
- CREATE UNIQUE INDEX Xisbn0 on Tisbn(buchnr,lfdnr)
- ;
- Ein Index ist ein Objekt der physischen Ebene. Parameter beim CREATE INDEX steuern die physische Datenorganisation. Vergleichen Sie dazu bitte die Diskussion zur Performance.

```
ALTER TABLE Tisbn
      ADD CONSTRAINT PK_tisbn PRIMARY KEY (Isbn)
;
ALTER TABLE Tisbn
      ADD CONSTRAINT ALTKEY_Tisbn UNIQUE (Buchnr,Lfdnr)
;
ALTER TABLE Tisbn
      ADD CONSTRAINT FK_Tisbn_Tbuch
      FOREIGN KEY (Buchnr) REFERENCES Tbuch(Buchnr)
      ON DELETE CASCADE
;
ALTER TABLE Tisbn ADD CONSTRAINT Conabc
    CHECK ( 1<= Tisbn.Lfdnr AND Tisbn.Lfdnr <= 5)
;
ALTER TABLE Tautor ADD Vorname CHAR (20)
;
ALTER TABLE Tautor DROP COLUMN Vorname
;
```

abgeleitete Spalten

- DB2

- Oracle

- Beispiel SQL Server
- CREATE TABLE Tbuchdemo
- (
- Buchnr INTEGER NOT NULL
- ,Preis DECIMAL(7,2)
- ,Preisbrutto AS preis * 1.3
- ,PRIMARY KEY (Buchnr)
-)
- ;

CREATE TABLE …AS
CREATE TABLE …LIKE

- DB2
- Oracle
- SQL Server

IDENTITY

- IDENTITY ermöglicht im Rahmen einer Table die automatische Vergabe einer laufenden Nummer.

- DB2

- Oracle

- SQL Server

SEQUENCE

- DB2
- Oracle
- SQL Server

- SQL Server 2012
- 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
- ;
- select 'zweiter aufruf' as s1
- , next value for buchnr_seq as s2
- ;

Create Trigger

- Mit Hilfe der Trigger können ereignisabhängige Verarbeitungsschritte angestoßen werden. Die Anwendungslogik kann somit in das RDBMS übertragen werden.

- DB2
- Oracle
- SQL Server

Benutzerdefinierte Datentypen

- DB2
- Oracle
- SQL Server

Datendefinition und Transaktion

- Achtung Oracle:
- Eine Anweisung der Datendefinition führt bei Oracle zum COMMIT.

SCHEMA und USER

- DB2 trennt die implizite Verbindung User und Schema.
- SQL Server 2005 trennt die implizite Verbindung zwischen User und Schema.

12

ANHANG A TRANSAKTIONSVERARBEITUNG, DER ISOLATION LEVEL

Was ist eine 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.

ACID Eigenschaften einer Transaktion

- Atomicity/Unteilbarkeit
- Consistency/Konsistenz
- Isolation
- Durability/Beständigkeit

- Die Eigenschaft Atomicity/Unteilbarkeit ist, was eine einzelne SQL-Anweisung betrifft, vom System garantiert! Die Eigenschaft Atomicity/Unteilbarkeit ist aber, was mehrere SQL-Anweisungen betrifft, ohne besondere Bemühungen des Endanwenders nicht gegeben!
- Die Eigenschaft Consistency/Konsistenz wird vom System garantiert.
- Die Eigenschaft Durability/Beständigkeit wird vom System garantiert.
- Die Eigenschaft Isolation ist ohne besondere Bemühungen des Endanwenders nicht notwendigerweise gegeben, sie ist eine Entscheidung des Anwendungsdesigns.

Transaktion und Atomicity

Trans1	Zeitpunkt
UPDATE Tbuch SET Preis = Preis - 99999.99/2 WHERE Buchnr = 5 CONSTRAINT Tbuchconpreis CHECK (0.00 < Preis) ist nicht erfüllt!	t1
UPDATE Tbuch SET Preis = Preis + 99999.99 WHERE Buchnr = 6 Overflow bei Datentyp DECIMAL(7,2)	t2
	t3
COMMIT	t4

- Was ist das Problem?
- Der COMMIT wird ausgeführt, unabhängig davon ob alle Update-Anweisungen erfolgreich ausgeführt wurden.
- Die Transaktion muss mit ROLLBACK abgebrochen und der Batch beendet werden, wenn eine der Update-Anweisungen nicht erfolgreich ausgeführt werden kann.

Optionen der Transaktionsverarbeitung

- Oracle
- **setzen isolation level
- SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
- SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
- SET TRANSACTION READ ONLY;
- **autocommit auf off setzen
- SET AUTOCOMMIT OFF;
- **Laufzeitfehler
- Benutzen Sie PL/SQL und ROLLBACK in der EXCEPTION-Verarbeitung
- **festlegen timeoutwert
- SELECT … FOR UPDATE OF … WAIT 15

Optionen der Transaktionsverarbeitung

- SQL Server
- **setzen isolation level
- SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
- SET TRANSACTION ISOLATION REPEATABLE READ;
- SET TRANSACTION ISOLATION READ COMMITTED;
- SET TRANSACTION ISOLATION READ UNCOMMITTED;
- SET TRANSACTION ISOLATION SNAPSHOT;
- **autocommit auf off setzen
- SET IMPLICIT_TRANSACTIONS ON
- Vermeiden Sie BEGIN TRANSACTION
- **Laufzeitfehler
- Benutzen Sie TRANSACT SQL und ROLLBACK in der TRY…CATCH-Verarbeitung
- SET XACT_ABORT ON
- Bei Laufzeitfehlern wird der Batch beendet und ein Rollback der Transaktion durchgeführt. Achtung: bei Fehlern wie „DROP TABLE Tbaaach“ -die Table ist nicht vorhanden- wird der Batch nicht beendet, es wir kein Rollback durchgeführt.
- **festlegen timeoutwert
- SET LOCK_TIMEOUT 5000
- Der Timeoutwert legt fest, wie viele Millisekunden eine Anweisung auf die Aufhebung einer Sperre wartet.
  Wenn die Wartezeit auf eine Sperre den Timeoutwert überschreitet, wird ein Fehler zurückgegeben.
- Am Anfang der Verbindung hat diese Einstellung den Wert -1. Der Wert -1 (Standardwert) gibt an, dass keine Wartezeit festgelegt ist (d. h., es wird ewig gewartet). Wird der Wert geändert, so bleibt die neue Einstellung für die restliche Verbindungsdauer bestehen. Der Wert 0 gibt an, dass nicht gewartet und eine Meldung zurückgegeben wird, sobald eine Sperre auftritt.

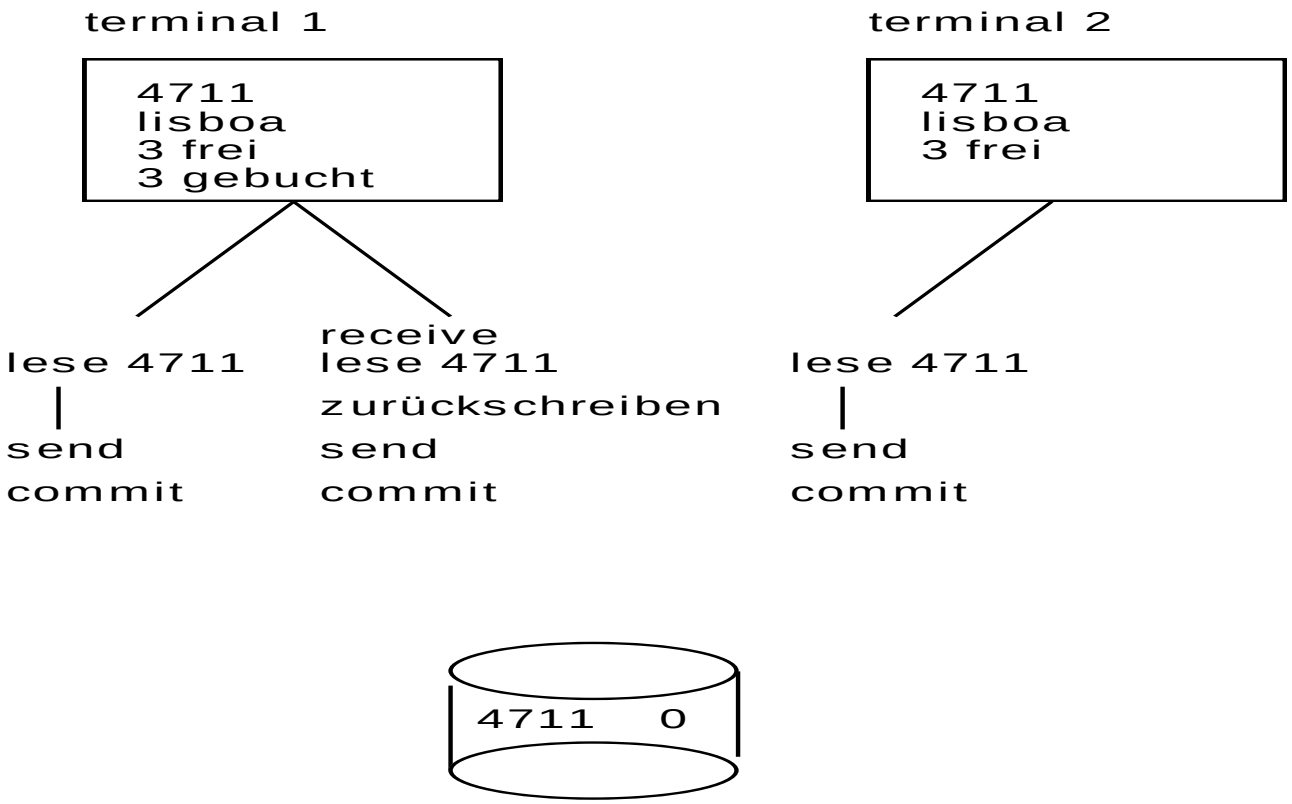
- DB2 for Windows, UNIX, Linux
- **setzen isolation level vor dem Connect
- CHANGE ISOLATION TO RR;
- CHANGE ISOLATION TO RS;
- CHANGE ISOLATION TO CS;
- CHANGE ISOLATION TO UR;
- jetzt CONNECT TO databasename
- **festlegen isolation level auf Ebene der SQL-Anweisung
- SELECT … WITH RR;
- SELECT … WITH RS
- SELECT … WITH CS;
- SELECT … WITH UR;
- **autocommit auf off setzen
- UPDATE COMMAND OPTIONS USING C OFF;
- **Laufzeitfehler
- UPDATE COMMAND OPTIONS USING S ON;
- bei Laufzeitfehlern wird die Skriptverarbeitung abgebrochen ohne Rollback der Transaktion. Diese Option erfordert zwingend einen ROLLBACK nach dem Abbruch eines Skripts!
- **festlegen timeoutwert
- update db cfg for sample using locktimeout 15;
- **festlegen Konfigurationsparameter CUR_COMMIT
- UPDATE DB CFG FOR sample USING cur_commit ON
- UPDATE DB CFG FOR sample USING cur_commit DISABLED

Der Isolation Level

Isolation Level 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

Isolation Level SQL Server	Isolation Level DB2	Isolation Level ORACLE
READ UNCOMMITTED	uncommitted read UR	
READ COMMITTED	cursor stability CS	READ COMMITTED
REPEATABLE READ	read stability RS	
SERIALIZABLE	repeatable read RR	
		“SERIALIZABLE” ????????????
SNAPSHOT		

Transaktion und Concurrency



- Wir können zwei Sperrprobleme identifizieren:
- Das Sperren von Daten über Dialogschritte hinweg, transaktionsübergreifend.
- Das Sperren von Daten während einer Transaktion.

- In einem Datenbankmanagementsystem werden dieselben Daten "gleichzeitig" von verschiedenen Anwendern bearbeitet. Der konkurrierende Zugriff auf die Daten muss überwacht und verwaltet werden. Um die Konsistenz der Daten zu gewährleisten sind folgende drei Konkurrenzprobleme zu lösen:

  - The Lost Update Problem: „Ein UPDATE geht verloren“
  - The Uncommitted Dependency Problem: „Eine Transaktion darf Daten, die von einer anderen Transaktion verändert aber noch nicht committed wurden (die also möglicherweise wieder zurückgesetzt werden), lesen bzw. sogar ändern“
  - The Inconsistent Analysis Problem: „eine Transaktion sieht einen inconsistenten Zustand der Datenbank und kommt dadurch zu einer inconsistenten Analyse“

Der verlorene Update

	Zeitpunkt	
SELECT Preis FROM Tbuch WHERE Buchnr = 5 Es sind 3.50	T1	
erhöhe um 200.00	T2	SELECT Preis FROM Tbuch WHERE Buchnr = 5 Es sind 3.50
UPDATE Tbuch SET Preis = 203.50 WHERE Buchnr = 5 COMMIT	T3	ziehe 50.00 ab
	T4	UPDATE Tbuch SET Preis = – 46.50 WHERE Buchnr = 5

Der schmutzige Read

Trans1	Zeitpunkt	Trans2
	t1	UPDATE (Konto y)
Oracle Ein schmutziges Lesen ist nicht möglich!		
SQL Server SET IMPLICIT_TRANSACTIONS ON; SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED; SELECT (Konto y) DB2 for Windows, UNIX, Linux UPDATE COMMAND OPTIONS USING C OFF; CHANGE ISOLATION TO UR; SELECT (Konto y) SELECT WITH UR (Konto y)	t2	
DB2 for z/OS SPUFI „autocommit off“ “ISOLATION RR im SPUFI Default Panel” SELECT WITH UR (Konto y)	t2	
	t3	ROLLBACK

Die nicht korrekte Analyse

- SELECT SUM(Kontostand) FROM Tkonten;
- KONTO 1 = 40, KONTO 2 = 50, KONTO 3 = 30

Trans1	Zeitpunkt	Trans2
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	

Der Isolation Level einer Transaktion und die Möglichkeiten der Konkurrenten

integrata cegos

- Ein SELECT innerhalb einer Transaction trans1 kann liefern
- die leere Menge
- genau eine Zeile
- eine echte Teilmenge der Base Table
- alle Zeilen der Base Table.
- Während eines SELECTs bzw. nach einem SELECT innerhalb einer Transaktion trans1 hat eine Transaktion trans2 - abhängig vom Isolation Level der Transaktion trans1 - Zugriff auf die Daten oder muss warten bis zum COMMIT der Transaktion trans1.
- Der Isolation Level spezifiziert welche Phänomene bei konkurrierenden Transaktionen möglich bzw. nicht möglich sind.
- Der Isolation Level steuert bei sperrenden Systemen wie lange ein Lock nach einem SELECT gehalten wird! Abhängig vom Zugriffspfad wird aber manchmal mehr gesperrt als logisch nötig ist.

Die fatale Situation, der Isolation Level ist Bestandteil des Anwendungsdesigns!

- Sofern Transaktion Trans2 liest, dabei keine Sperre setzt und anschließend ohne zu Prüfen zurückschreibt, verliert die Transaktion Trans1 ihren UPDATE

Trans1	Zeitpunkt	Trans2 Innerhalb einer Transaktion oder transaktionsübergreifend: Lesen und Schreiben
	t1	**Lesen ohne Sperren**
Lesen mit / ohne Sperren	t2	
Schreiben ohne / mit Prüfen	t3	
COMMIT	t4	
	t5	**Schreiben ohne Prüfen**

- Der verlorene Update ist nicht möglich, sofern alle Transaktionen beim Lesen sperren oder beim Zurückschreiben prüfen!

- Der verlorene Update ist nicht möglich, sofern alle Transaktionen mit dem schärfsten Isolation Level arbeiten!

- Der verlorene Update ist nicht möglich, sofern für jede UPDATE-Anweisung (zum Beispiel unter der Bedingung „autocommit on“) der zu schreibende Wert nicht abhängig ist von einem vorher gelesenen Wert.

- Was ist wichtiger?
- Die Integrität der Daten oder die Performance?

Verpflichtung der Implementierung

- Verpflichtung der Implementierungen: „The execution of concurrent SQL-transactions at isolation level SERIALIZABLE is guaranteed to be serializable…”.
-
- SQL Server SERIALIZABLE garantiert serialisierbar.
-
- DB2 RR repeatable read garantiert serialisierbar.
-
-
- Oracle’s „SERIALIZABLE“ garantiert serialisierbar nicht“.
-
- SQL Server SNAPSHOT garantiert serialisierbar nicht.

Oracle's „SERIALIZABLE" garantiert serialisierbar nicht".

Trans1 SET TRANSACTION ISOLATION LEVEL SERIALIZABLE	Zeitpunkt	Trans2 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

SQL Server SNAPSHOT „garantiert serialisierbar nicht“

integrata cegos

Trans1 **SET TRANSACTION ISOLATION LEVEL SNAPSHOT**	**Zeitpunkt**	**Trans2** **SET TRANSACTION ISOLATION LEVEL SNAPSHOT**
Lese PK = 5 gefunden	t1	
	t2	Lese FK=5 nicht gefunden
INSERT FK=5 **COMMIT**	t3	
	t4	DELETE PK=5

Beispiel DB2

Isolation Level SQL Standard	Isolation Level DB2
READ UNCOMMITTED	uncommitted read UR
READ COMMITTED	cursor stability CS
REPEATABLE READ	read stability RS
SERIALIZABLE	repeatable read RR

- Verträglichkeitsmatrix für Locks

	S share lock	U update lock	X exclusive lock
S share lock			wait
U update lock		wait	wait
X exclusive lock	wait	wait	wait

LOCK TABLE

- ORACLE
- DB2
- SQL Server

Prüfung der Konsistenz beim COMMIT

- Beispiel Oracle

Die ACID-Eigenschaften einer Transaktion sind ohne besondere Bemühungen des Endanwenders nicht gegeben!

- Girokonto + 10 Euro und Sparkonto – 10 Euro

- Ein Buch soll 10 Euro teurer werden, das andere 10 Euro billiger. Beide Bücher müssen samt Preis vorhanden sein und beide Update-Anweisungen müssen erfolgreich sein!

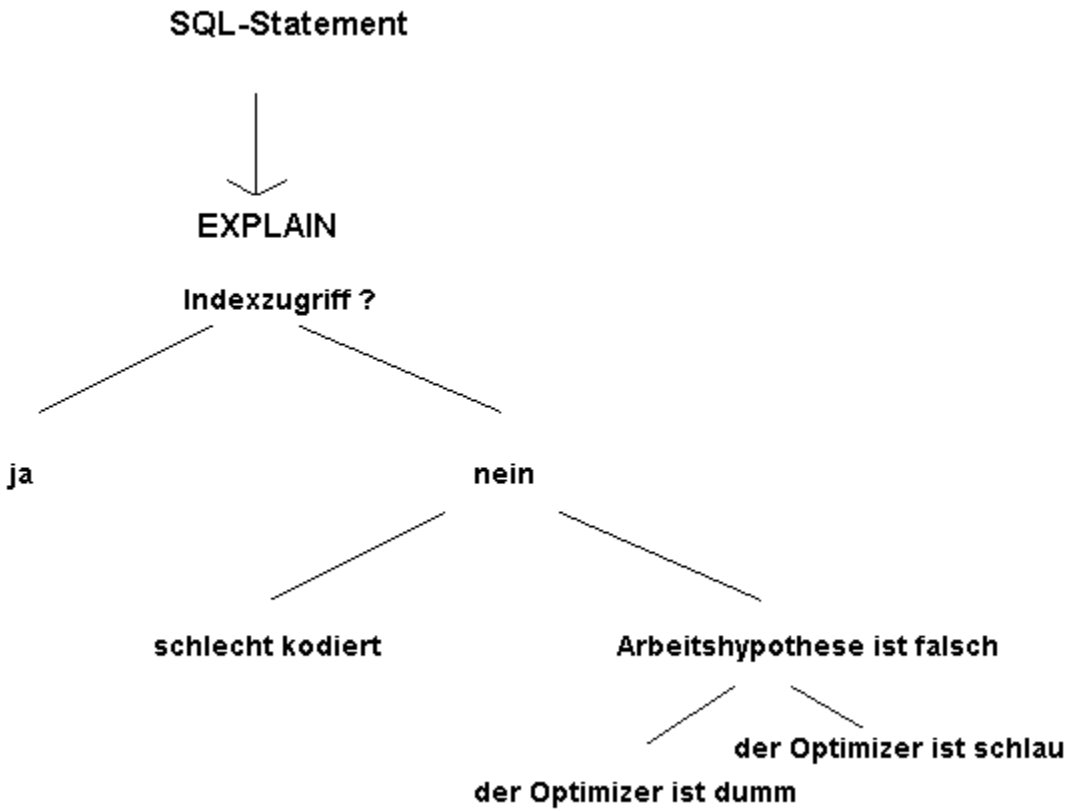
Beide Update-Anweisungen müssen erfolgreich ausgeführt werden oder keine von beiden.

- Fachliche Anforderung
- “Ein Buch soll 10 Euro teurer werden, das andere 10 Euro billiger. Beide Bücher müssen samt Preis aber ohne ISBNs vorhanden sein.
- Die Prämien sollen auch angepasst werden, d.h. jede wohldefinierte Praemie wird um 10 erhöht bzw. um 10 reduziert.

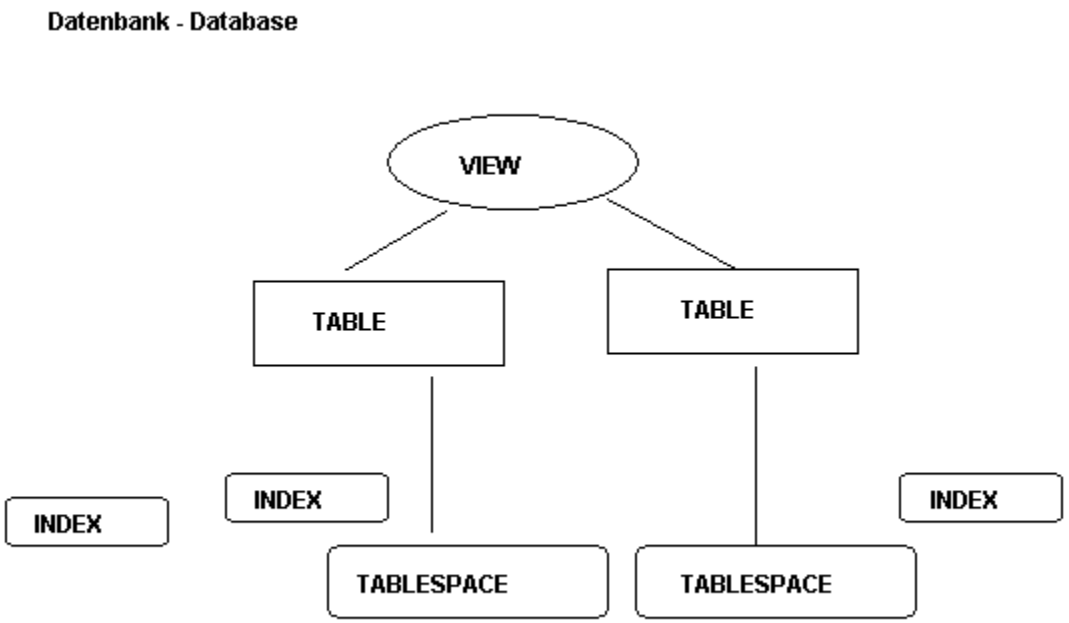
13

ANHANG B PERFORMANCEBETRACHTUNGEN, DER OPTIMIZER

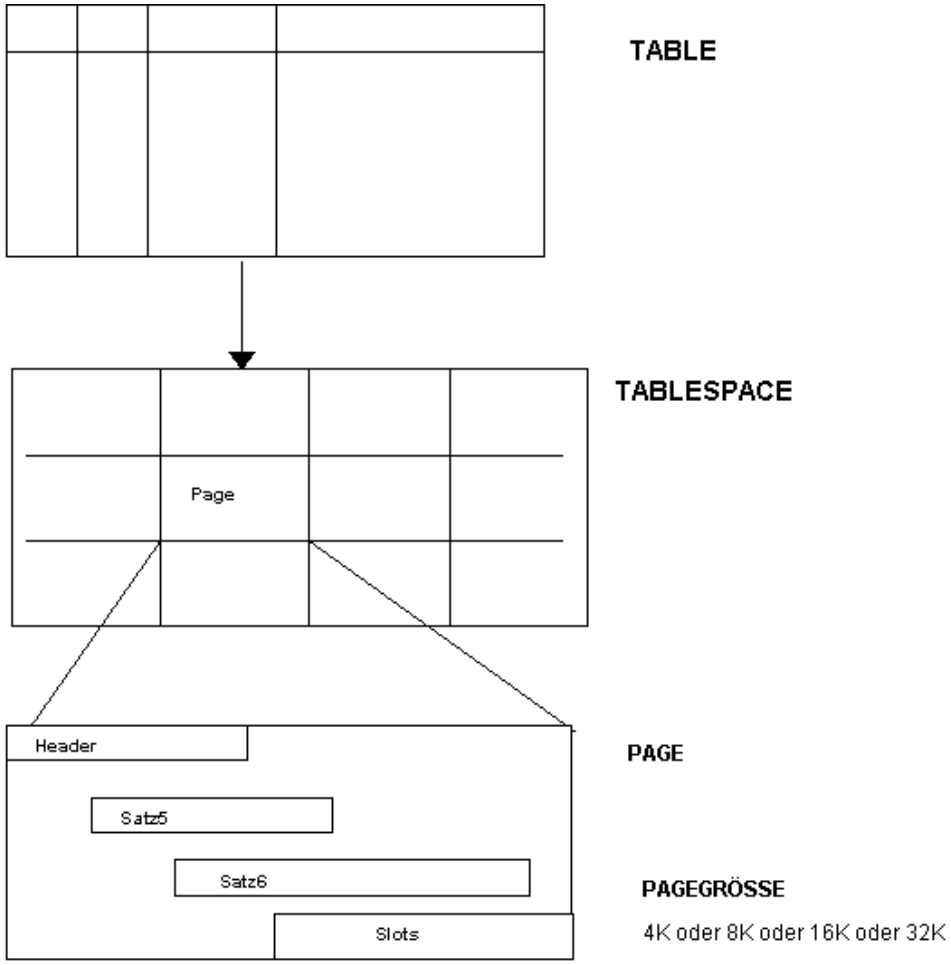
Was und wie


SQL-Statement
EXPLAIN
Indexzugriff ?
ja
nein
schlecht kodiert
Arbeitshypothese ist falsch
der Optimizer ist schlau
der Optimizer ist dumm

Konzeptionelles Datenmodell – Physische Datenstruktur



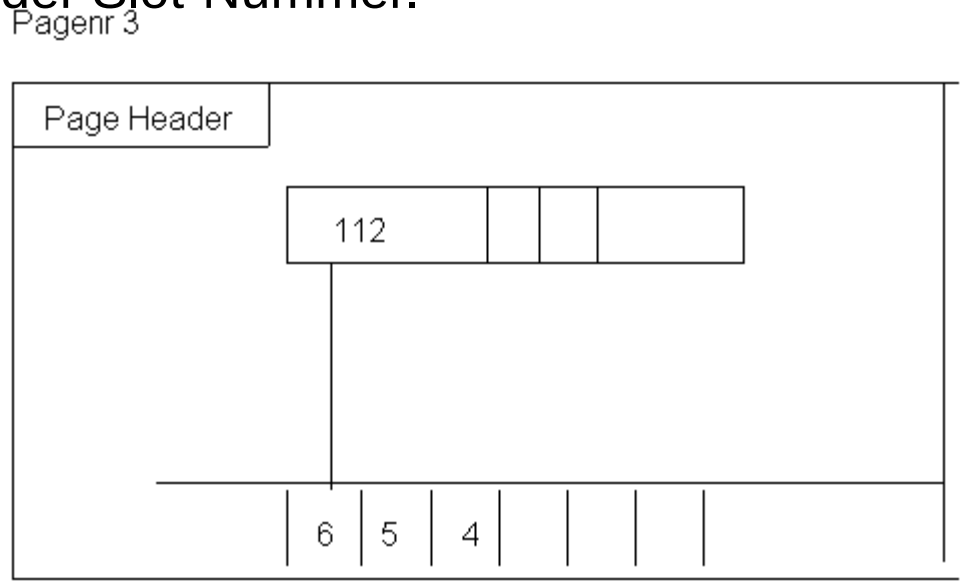
Interne Datenablage



- Auf der Ebene der physischen Speicherung reden wir von Satz-Record, wobei jeder Satz einer Zeile-Row der Basetable entspricht!

Schema einer Daten-PAGE

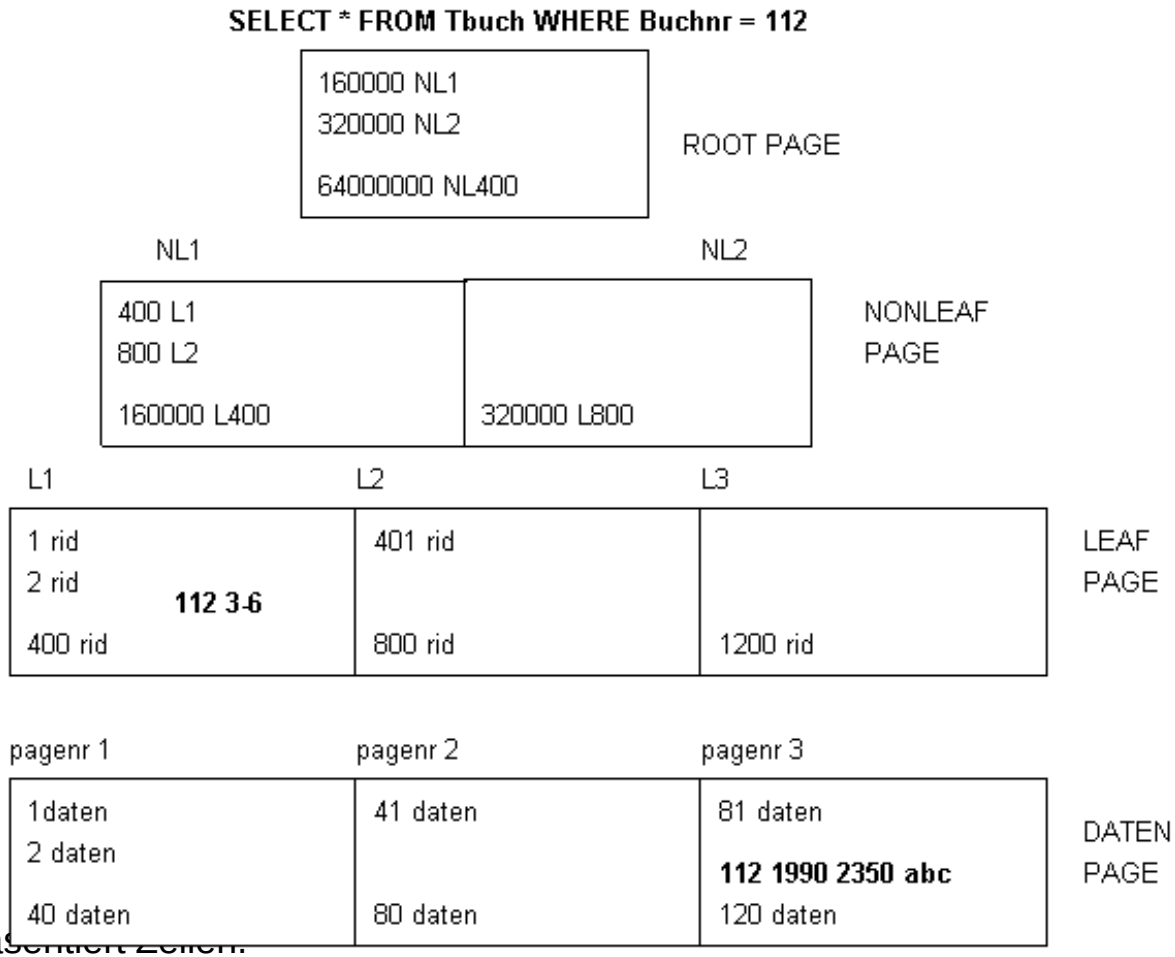
- Der Header beinhaltet unter anderem Informationen über den Zustand der Page, wie beispielsweise die Größe des insgesamt zur Verfügung stehenden Platzes oder die Adresse des ersten freien Platzes.
- Die Adresse eines Satzes (Rid) setzt sich zusammen aus der Page-Nummer und der Slot-Nummer.



Eintrag im Index:

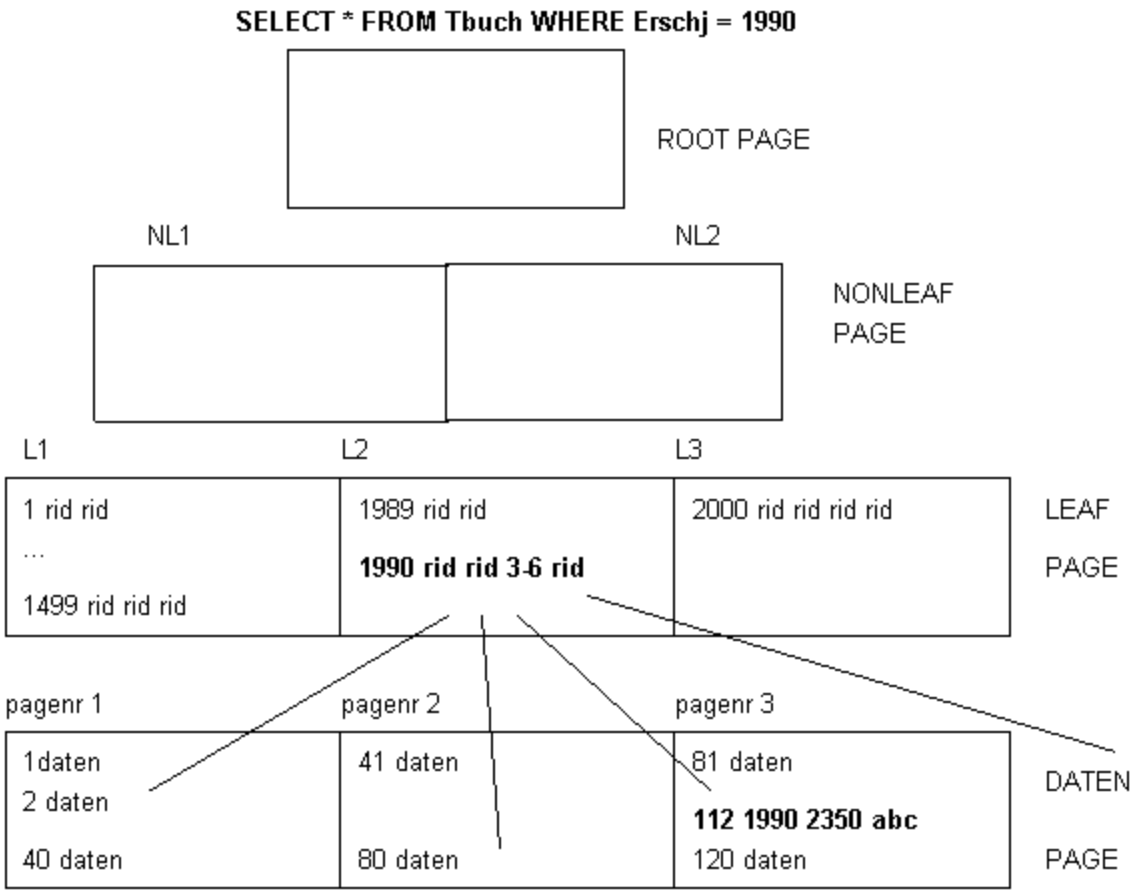
Wert	Rid
112	3/6

INDEX B-Tree – UNIQUE INDEX



- Eine Table präsentiert Zeilen.
- Auf der physischen Ebene werden Sätze/Records gespeichert!

INDEX B-Tree - nonunique



Sequentieller Zugriff oder Indexzugriff?

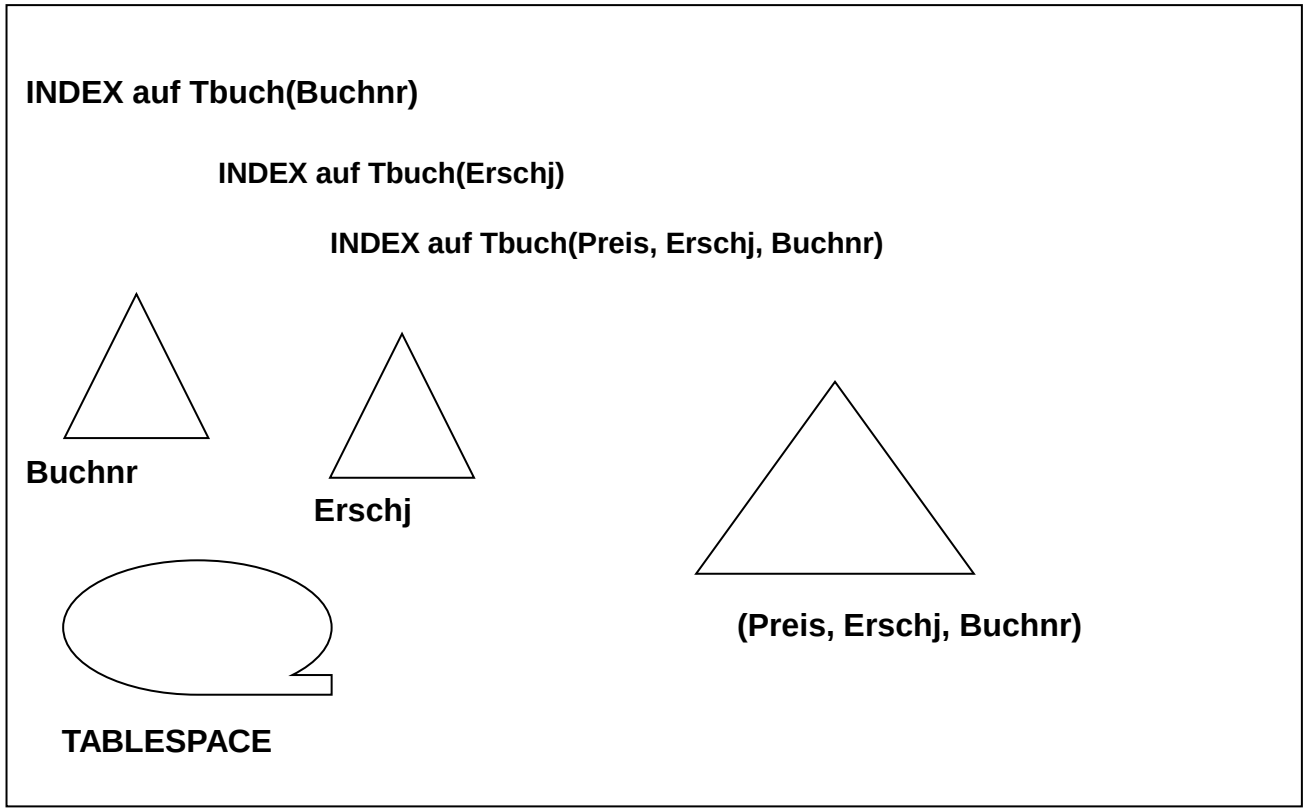
-
- SELECT * FROM Tbuch
- SELECT * FROM Tbuch WHERE Buchnr = 5
- SELECT * FROM Tbuch WHERE Buchnr > 0
- SELECT * FROM Tbuch WHERE Buchnr > 10000
-
- SELECT * FROM Tbuch WHERE Erschj = 1990
- SELECT * FROM Tbuch WHERE Erschj > 0
- SELECT * FROM Tbuch WHERE Erschj > 1900
- SELECT * FROM Tbuch WHERE Erschj > 2000
-

Steuerung der physischen Speicherung

- DB2
- Oracle
- SQL Server

- Stichwort CLUSTER
- Stichwort INDEX

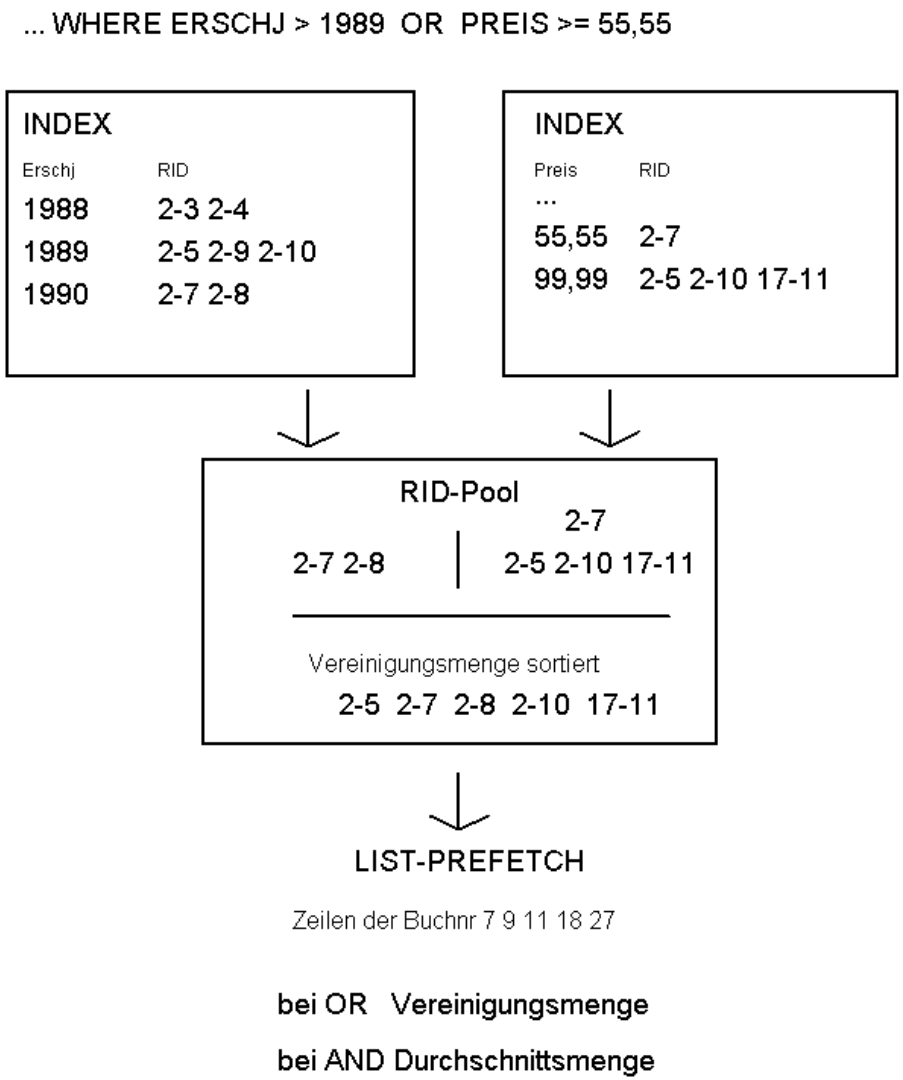
Zusammengesetzter INDEX



- SELECT Buchnr, Preis
- FROM Tbuch WHERE Erschj = 1990;
-
- SELECT Buchnr, Preis, Erschj
- FROM Tbuch WHERE Preis > 5.00;
-
- SELECT Buchnr, Preis, Erschj
- FROM Tbuch
- WHERE Preis > 5.00 AND Erschj > 1999;
-
- SELECT Erschj, Preis, Buchnr
- FROM Tbuch
- WHERE Erschj = 1990 AND Preis = 5.00;
-
- SELECT Erschj, Preis, Buchnr
- FROM Tbuch
- WHERE Erschj = 1990 OR Preis = 5.00;
-

- CREATE UNIQUE INDEX Xpreersbuc
- ON Tbuch(Preis, Erschj, Buchnr);
- Ein zusammengesetzter INDEX ermöglicht bei AND einen Table Access by Index Rowid / Matching Index Scan / Index Seek, bei OR aber nur einen Index Full Scan / Nonmatching Index Scan / Index Scan!
- Ein zusammengesetzter INDEX ermöglicht Zugriffe mit Index Only.
- Beim Generieren eines zusammengesetzten Indices ist auf die Reihenfolge der Spalten zu achten. Sie steht in engem Zusammenhang mit den SELECT-Anforderungen.
-

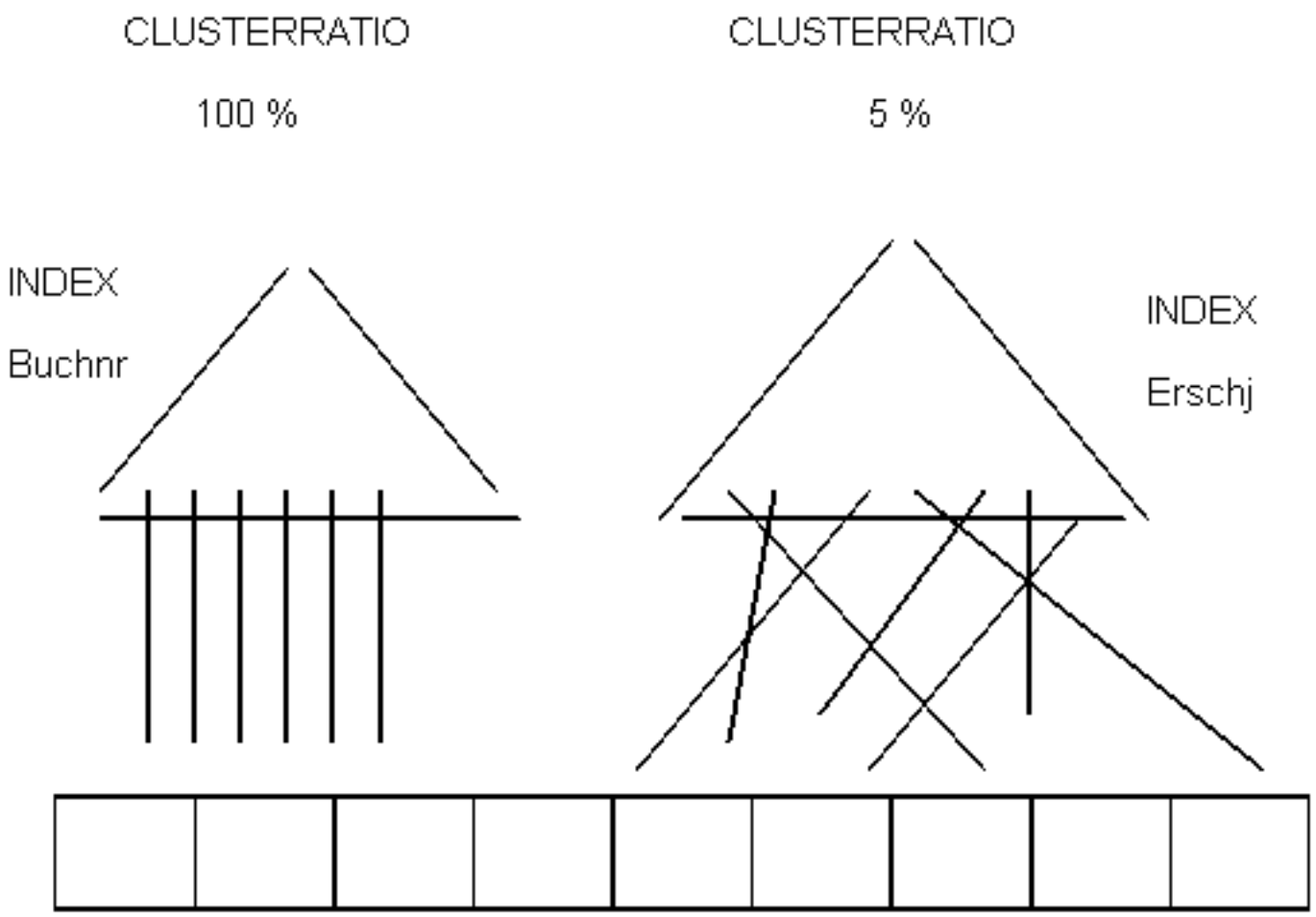
Multiple Index Access



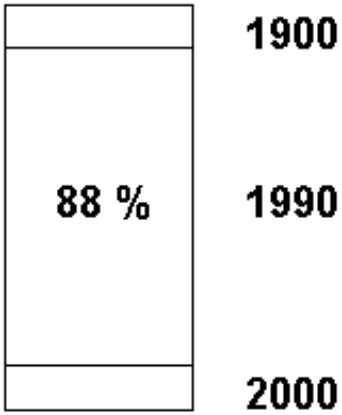
bei OR Vereinigungsmenge

bei AND Durchschnittsmenge

ORDER BY und INDEX – „Einen Sort vermeiden!“



Welchen Zugriffspfad erwarten Sie bei folgenden Abfragen?



- SELECT * FROM Tbuch WHERE Erschj = 1900;
- SELECT * FROM Tbuch WHERE Erschj = 1990;
- SELECT * FROM Tbuch
- WHERE Erschj + 10. = 2010;
-

EXPLAIN „schlecht kodiert?“

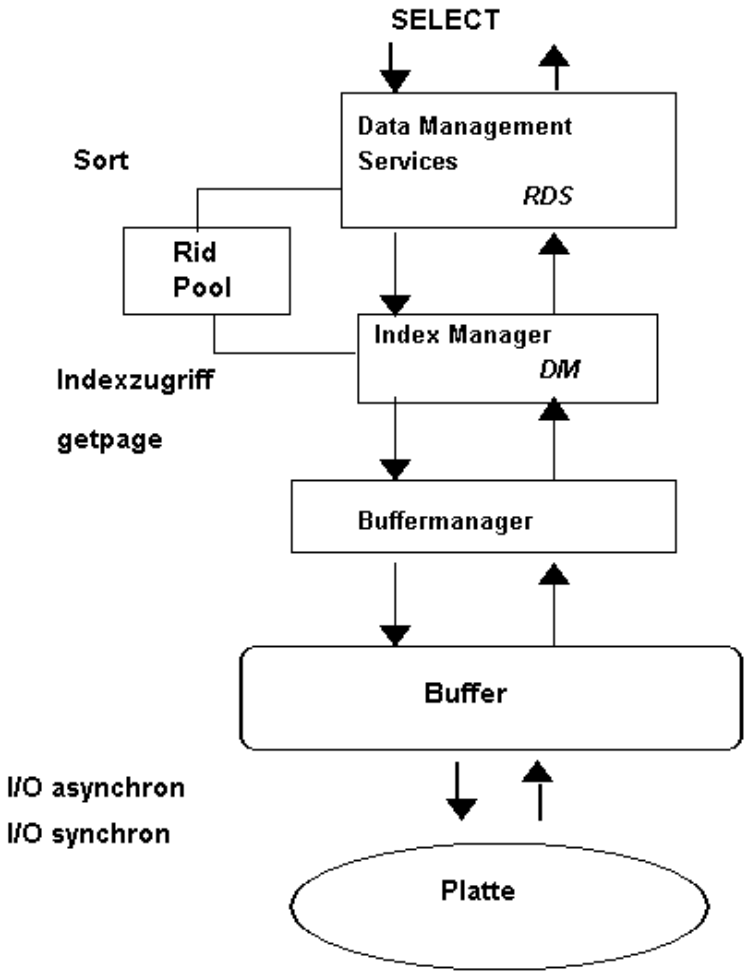
- Achtung: Expressions in der Where-Klausel führen bei den meisten Produkten am Markt zu einem sequentiellen Zugriff, ein Index Seek (Matching Index Scan) ist nicht möglich!
- SELECT * FROM Tbuch WHERE Buchnr + 0 = 8;
- SELECT * FROM Tbuch WHERE Preis/2 = 5.20;
- SELECT * from Tbuch WHERE SUBSTRING(Titel, 1, 1) ='C';
- SELECT * from Tbuch WHERE LOWER(Titel) = 'der butt';
- SELECT * FROM Tautor WHERE YEAR(Geburtsdatum)=2012 ;
- Wahl effizienter SELECT-Anweisungen
- Ein wichtiger Aspekt beim Design einer Anwendung ist die Wahl effizienter Anweisungen. Ein Anforderung (WAS) kann häufig durch verschiedene Anweisungen formuliert werden. Der vom Optimizer gewählte Zugriffspfad (WIE) hängt oft noch von der Formulierung der Anweisung ab.
  Die Anwendungsentwicklung und Datenbankadministration kann also durch eine geeignete Formulierung der Anweisung die Performance beeinflussen.
-

Der Optimizer und die Statistikdaten

- SQL Server CREATE/UPDATE STATISTICS
- DB2 RUNSTATS
- Oracle ANALYZE

- SQL SERVER automatische Statistikaktualisierung
-

Komponenten des Datenbankmanagement-systems



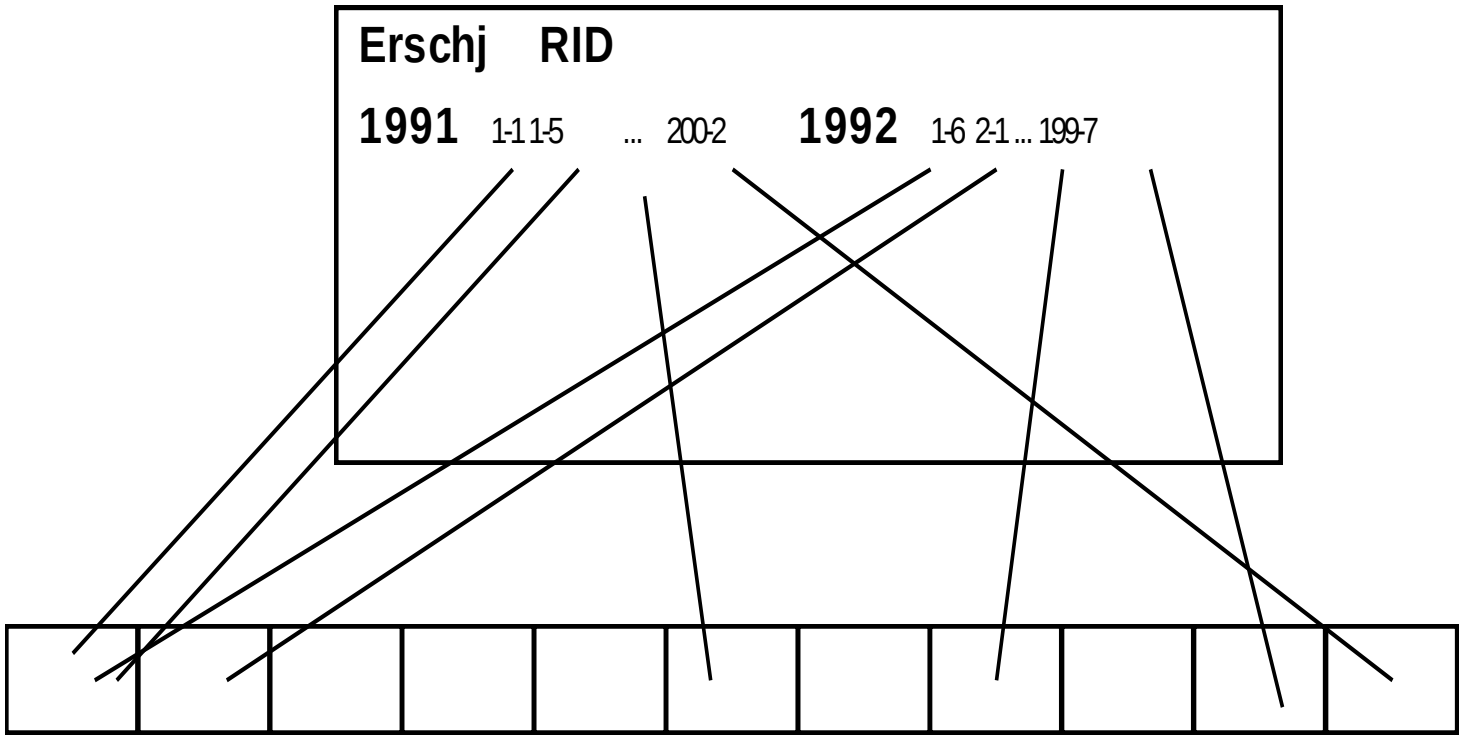
Sequential Prefetch – List Prefetch

SELECT * FROM Tbuch

WHERE Erschj BETWEEN 1991 AND 1992

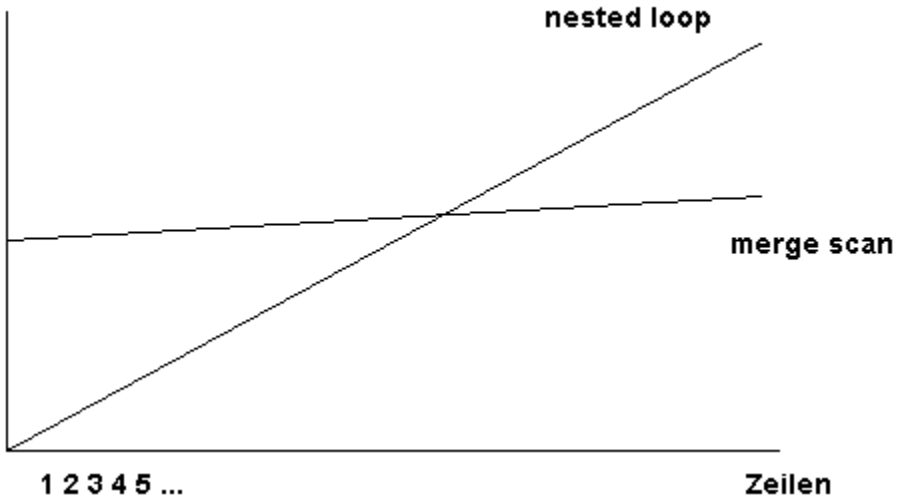
ORDER BY ERSCHJ;

CLUSTERRATIO = 1%



Zugriffsmethoden für den JOIN

- Tables können logisch auf verschiedene Weise (das Was!) gejoint werden:
- INNER JOIN
- LEFT OUTER JOIN bzw. RIGHT OUTER JOIN
- FULL OUTER JOIN
- Der Optimizer hat auf der physischen Ebene mehrere Möglichkeiten für das
  Wie: Merge Scan Join, Nested Loop Join, Hash Join.



- Beim Merge Scan Join werden für jede Zeile der einen Table die passenden Zeilen der anderen Table dazugemischt (Merge-Logik). Die dazu nötigen sortierten Dateien entstehen durch interne Sorts sofern keine geeigneten Indizes vorhanden sind.
- Beim Nested Loop Join werden für jede Zeile der einen Table die passenden Zeilen in der anderen Table gesucht.
-
-
-

Wie schlau ist der Optimizer?

- Überprüfen Sie Ihren Optimizer! Welche Zugriffspfade wählt der Optimizer für Ihre Anweisungen?
-
- SELECT DISTINCT Tbuch.Buchnr, Tbuch.Titel ,erschj
- FROM Tbuch INNER JOIN Tisbn
- ON Tbuch.Buchnr = Tisbn.Buchnr
- WHERE Tbuch.Erschj = 1977
-
- SELECT Buchnr, Titel FROM Tbuch
- WHERE Tbuch.Buchnr IN
- (SELECT Tisbn.Buchnr FROM Tisbn)
- AND Tbuch.Erschj = 1977
-
- SELECT Buchnr, Titel FROM Tbuch
- WHERE EXISTS
- (SELECT * FROM Tisbn
- WHERE Tisbn.Buchnr = Tbuch.Buchnr)
- AND Tbuch.Erschj = 1977

VIEW materialization – VIEW merge

```
DROP VIEW JAHRESSUMMEN
GO
CREATE VIEW Jahressummen (Jahr, Summe)
       AS SELECT Erschj, SUM(Preis)
          FROM Tbuch
          GROUP BY Erschj
GO
SELECT * FROM Jahressummen
;
SELECT * FROM Jahressummen
WHERE JAHR = 1988
;
SELECT * FROM Jahressummen
WHERE Summe > 70.00
;
SELECT MAX(Summe)
FROM Jahressummen
;
```

- VIEW materialization
- Eine Möglichkeit, eine VIEW zu verarbeiten, besteht darin, die Datenzeilen in eine Workdatei zu "materialisieren" und diese materialisierte VIEW wie eine TABLE zu verarbeiten.
-
- VIEW merge
- Die Anweisung, die die VIEW referenziert wird mit dem Fullselect, der die VIEW definiert, kombiniert.

materialized view bzw. materialized query table

- Oracle
- DB2
- SQL Server

14

ANHANG C OLAP, GROUPING SETS, ROLLUP, CUBE, “QUERY UND WINDOW”

GROUP BY

- select buchnr, autornr, praemie
- from tvautor
- order by buchnr, autornr;
- BUCHNR AUTORNR PRAEMIE
- ---------- ---------- ---------
- 1 1 300,00
- 1 2 200,00
- 2 1 300,00
- 2 2 400,00
- 3 2 200,00
- 4 2 200,00
- --QUERY 1 summe aller praemien
- select sum (praemie) as sum1
- from tvautor
- group by ()
- ;
- --QUERY 2 summe aller praemien pro buch
- select buchnr, sum (praemie) as sum2
- from tvautor
- group by (buchnr)
- ;

- --QUERY 3 summe aller praemien pro autor
- select autornr, sum (praemie) as sum3
- from tvautor
- group by (autornr)
- ;
- --QUERY 4 summe aller praemien pro buchnr, autornr
- --sum(praemie) entspricht hier
- --der einzelpraemie pro buchnr, autornr
- select buchnr, autornr, sum(praemie) as sum4
- from tvautor
- --group by (buchnr, autornr); nicht SQL Server
- group by buchnr, autornr
- ;

GROUPING SETS

- -- kombination von query 2 und 3
- -- achtung: das ergebnis ist keine relation!
- select buchnr, autornr, sum(praemie) as sum23
- from tvautor
- group by grouping sets ((buchnr), (autornr))
- ;
- BUCHNR AUTORNR SUM23
- ---------- ---------- ---------------------------
- - 1 600,00
- - 2 1000,00
- 1 - 500,00
- 2 - 700,00
- 3 - 200,00
- 4 - 200,00
-

GROUP BY ROLLUP

```
-- kombination von query 4, 2 und 1
-- rollup (buchnr, autornr)
-- ist eine andere schreibweise für
-- grouping sets ((buchnr, autornr), (buchnr), ());
select buchnr, autornr, sum(praemie) as sumrollup421
from tvautor
group by rollup (buchnr, autornr)
;
BUCHNR     AUTORNR    SUMROLLUP421
---------- ---------- --------------------------
         -          -               1600,00
         1          -                500,00
         2          -                700,00
         3          -                200,00
         4          -                200,00
         1          1                300,00
         1          2                200,00
         2          1                300,00
         2          2                400,00
         3          2                200,00
         4          2                200,00
```

GROUP BY CUBE

- -- kombination von query 4, 3, 2 und 1
- -- cube (buchnr, autornr)
- -- ist eine andere schreibweise für
- -- grouping sets
  -- ((buchnr, autornr), (buchnr), (autornr), ())
- select buchnr, autornr, sum(praemie) as sumcube4321
- from tvautor
- group by cube (buchnr, autornr);
- BUCHNR AUTORNR SUMCUBE4321
- ---------- ---------- ----------
- \- 1 600,00
- \- 2 1000,00
- \- - 1600,00
- 1 - 500,00
- 2 - 700,00
- 3 - 200,00
- 4 - 200,00
- 1 1 300,00
- 1 2 200,00
- 2 1 300,00
- 2 2 400,00
- 3 2 200,00
- 4 2 200,00
-
-

SQL Server und OLAP

- select buchnr, autornr, sum(praemie) as sumrollup421
- from tvautor
- group by buchnr, autornr WITH ROLLUP
- order by buchnr, autornr
- ;
- select buchnr, autornr, sum(praemie) as sumcube4321
- from tvautor
- group by buchnr, autornr WITH CUBE
- order by buchnr, autornr
- ;
- SELECT buchnr, praemie
- FROM tvautor
- ORDER BY buchnr
- COMPUTE SUM(praemie) BY buchnr
- ;
-

TABLE Emp

- select
- emp.deptno as deptno
- ,emp.Sal as sal
- ,emp.Empno as empno
- ,emp.Comm as comm
- ,substring(emp.Ename,1, 10) as ename
- ,substring(emp.Job ,1, 10) as job
- ----,emp.Mgr as mgr
- ----,emp.Hiredate as hiredate
- from emp
- order by deptno , sal, empno
- ;

PARTITION, ROW_NUMBER, RANK

integrata cegos

DEPTNO	SAL	EMPNO	COMM	ENAME	JOB
10	1300,00	7934	-	MILLER	CLERK
10	2450,00	7782	-	CLARK	MANAGER
10	5000,00	7839	-	KING	PRESIDENT
10	12345,67	7777	-	Otto	Putzmann
10	76543,21	8888	-	Fritz	Putzmann
10	99999,99	9999	-	Fritz	Putzmann
20	800,00	7369	-	SMITH	CLERK
20	1100,00	7876	-	ADAMS	CLERK
20	2975,00	7566	-	JONES	MANAGER
20	3000,00	7788	-	SCOTT	ANALYST
20	3000,00	7902	-	FORD	ANALYST
30	950,00	7900	-	JAMES	CLERK
30	1250,00	7521	500,00	WARD	SALESMAN
30	1250,00	7654	1400,00	MARTIN	SALESMAN
30	1500,00	7844	0,00	TURNER	SALESMAN
30	1600,00	7499	300,00	ALLEN	SALESMAN
30	2850,00	7698	-	BLAKE	MANAGER

17 Satz/Sätze ausgewählt.

- 1 GROUP BY oder PARTITION BY
- Pro Zeile soll die Summe der Gehälter in der Abteilung mit angelistet werden:
- select
- emp.deptno as deptno
- ,emp.Sal as sal
- ,emp.Empno as empno
- ,SUM(emp.sal) OVER
- (PARTITION BY emp.deptno) as SUM_sal
- from emp
- order by deptno, sal, empno
- ;
-

DEPTNO	SAL	EMPNO	SUM_SAL
10	1300,00	7934	197638,87
10	2450,00	7782	197638,87
10	5000,00	7839	197638,87
10	12345,67	7777	197638,87
10	76543,21	8888	197638,87
10	99999,99	9999	197638,87
20	800,00	7369	10875,00
20	1100,00	7876	10875,00
20	2975,00	7566	10875,00
20	3000,00	7788	10875,00
20	3000,00	7902	10875,00
30	950,00	7900	9400,00
30	1250,00	7521	9400,00
30	1250,00	7654	9400,00
30	1500,00	7844	9400,00
30	1600,00	7499	9400,00
30	2850,00	7698	9400,00

- 2 ROW_NUMBER() OVER und RANK() OVER
- SELECT
- ROW_NUMBER()
- OVER (ORDER BY sal ASC, empno ASC)
- AS "Rownr"
- ,sal
- AS "Sal"
- ,empno
- AS "Empno"
- ,RANK()
- OVER (ORDER BY sal ASC)
- AS "Ranksal"
- ,DENSE_RANK()
- OVER (ORDER BY sal ASC)
- AS "Denseranksal"
- FROM EMP
- ORDER BY sal, empno
- ;
-

Rownr	Sal	Empno	Ranksal	Denseranksal
1	800.00	7369	1	1
2	950.00	7900	2	2
3	1100.00	7876	3	3
4	1250.00	7521	4	4
5	1250.00	7654	4	4
6	1300.00	7934	6	5
7	1500.00	7844	7	6
8	1600.00	7499	8	7
9	2450.00	7782	9	8
10	2850.00	7698	10	9
11	2975.00	7566	11	10
12	3000.00	7788	12	11
13	3000.00	7902	12	11
14	5000.00	7839	14	12
15	12345.67	7777	15	13
16	76543.21	8888	16	14
17	99999.99	9999	17	15

- 3 ROW_NUMBER() OVER (PARTITION BY … und RANK() OVER (PARTITION BY…
- SELECT
- job AS "Job"
- ,ROW_NUMBER()OVER
- (PARTITION BY JOB ORDER BY sal ASC, empno ASC)
- AS "Rownrbyjob"
- ,sal
- AS "Sal"
- ,RANK()OVER
- (PARTITION BY job ORDER BY sal ASC)
- AS "Rankjob"
- ,empno
- AS "Empno"
- FROM EMP
- ORDER BY job, sal, empno
- ;
-

Job	Rownrbyjob	Sal	Rankjob	Empno
ANALYST	1	3000.00	1	7788
ANALYST	2	3000.00	1	7902
CLERK	1	800.00	1	7369
CLERK	2	950.00	2	7900
CLERK	3	1100.00	3	7876
CLERK	4	1300.00	4	7934
MANAGER	1	2450.00	1	7782
MANAGER	2	2850.00	2	7698
MANAGER	3	2975.00	3	7566
PRESIDENT	1	5000.00	1	7839
Putzmann	1	12345.67	1	7777
Putzmann	2	76543.21	2	8888
Putzmann	3	99999.99	3	9999
SALESMAN	1	1250.00	1	7521
SALESMAN	2	1250.00	1	7654
SALESMAN	3	1500.00	3	7844
SALESMAN	4	1600.00	4	7499

Query und Window

- 1. Beispiel: OVER (PARTITION BY …ORDER BY…ROWS BETWEEN)
- Abteilungsnummer, Gehalt und Personalnummer sollen aufsteigend angelistet werden und zusätzlich pro Abteilung die kumulierte Gehaltssumme innerhalb der Abteilung (Cumulative Window).
- select
- emp.deptno as deptno
- ,emp.Sal as sal
- ,emp.Empno as empno
- ,SUM(emp.sal) OVER
- (PARTITION BY emp.deptno
- ORDER BY emp.sal, emp.empno
- ROWS BETWEEN UNBOUNDED PRECEDING
- AND CURRENT ROW
-) as xxx
- from emp
- order by deptno , sal, empno
- ;
-

Job	Rownrbyjob	Sal	Rankjob	Empno
ANALYST	1	3000.00	1	7788
ANALYST	2	3000.00	1	7902
CLERK	1	800.00	1	7369
CLERK	2	950.00	2	7900
CLERK	3	1100.00	3	7876
CLERK	4	1300.00	4	7934
MANAGER	1	2450.00	1	7782
MANAGER	2	2850.00	2	7698
MANAGER	3	2975.00	3	7566
PRESIDENT	1	5000.00	1	7839
Putzmann	1	12345.67	1	7777
Putzmann	2	76543.21	2	8888
Putzmann	3	99999.99	3	9999
SALESMAN	1	1250.00	1	7521
SALESMAN	2	1250.00	1	7654
SALESMAN	3	1500.00	3	7844
SALESMAN	4	1600.00	4	7499

- 2. Beispiel: OVER (ORDER BY…ROWS BETWEEN…)

- Gehalt und Personalnummer sollen aufsteigend angelistet werden und zusätzlich die kumulierte Gehaltssumme (cumulative window).
- select
- emp.deptno as deptno
- ,emp.Sal as sal
- ,emp.Empno as empno
- ,SUM(emp.sal) OVER
- (
- --PARTITION BY emp.deptno
- ORDER BY emp.sal, emp.empno
- ROWS BETWEEN UNBOUNDED PRECEDING
- AND CURRENT ROW
-) as xxx
- from emp
- order by
- --deptno,
- sal, empno
- ;

deptno	sal	empno	xxx
20	800.00	7369	800.00
30	950.00	7900	1750.00
20	1100.00	7876	2850.00
30	1250.00	7521	4100.00
30	1250.00	7654	5350.00
10	1300.00	7934	6650.00
30	1500.00	7844	8150.00
30	1600.00	7499	9750.00
10	2450.00	7782	12200.00
30	2850.00	7698	15050.00
20	2975.00	7566	18025.00
20	3000.00	7788	21025.00
20	3000.00	7902	24025.00
10	5000.00	7839	29025.00
10	12345.67	7777	41370.67
10	76543.21	8888	117913.88
10	99999.99	9999	217913.87

- 3. Beispiel: OVER (PARTITION BY …ORDER BY… RANGE BETWEEN…)

- Wie viele Mitarbeiter in der Abteilung verdienen bis zu 350 Euro mehr?
- select
- emp.deptno as deptno
- , emp.sal as sal
- , emp.empno as empno
- , COUNT(emp.empno) OVER
- (
- PARTITION BY emp.deptno
- ORDER BY emp.sal ASc
- RANGE BETWEEN 0 PRECEDING
- AND 350 FOLLOWING
-) AS countdrumrum
-
- from emp
- order by deptno, sal, empno
- ;

-

DEPTNO	SAL	EMPNO	COUNTDRUMRUM
10	1300,00	7934	1
10	2450,00	7782	1
10	5000,00	7839	1
10	12345,67	7777	1
10	76543,21	8888	1
10	99999,99	9999	1
20	800,00	7369	2
20	1100,00	7876	1
20	2975,00	7566	3
20	3000,00	7788	2
20	3000,00	7902	2
30	950,00	7900	3
30	1250,00	7521	4
30	1250,00	7654	4
30	1500,00	7844	2
30	1600,00	7499	1
30	2850,00	7698	1

- 4. Beispiel: OVER (ORDER BY… RANGE BETWEEN…)

- Wie viele Mitarbeiter verdienen bis zu 350 Euro mehr?

Window Function, Window Partitioning, Window Ordering, Window Framing

- Ein Window ist eine vom Anwender spezifizierte Auswahl von Zeilen innerhalb einer Query (oder innerhalb einer logischen Partition einer Query) die die Menge von Zeilen bestimmt, die benutzt werden um gewisse Berechnungen auszuführen relativ zur aktuellen Zeile (current row under examination).

Pivoting Daten, Unpivoting Daten, PIVOT, UNPIVOT

integrata cegos

```
/*
schl        jahr        quartal     umsatz
----------- ----------- ----------- ---------------------------------
1           1991        1           0.70
2           1991        1           0.40
3           1991        2           1.20
4           1991        3           1.30
5           1991        4           1.40
6           1992        1           2.10
7           1992        2           2.20
8           1992        3           2.30
9           1992        4           0.80
10          1992        4           0.90
11          1992        4           0.70

(11 Zeile(n) betroffen)
*/
```

- select jahr, quartal, sum(umsatz) as betrag
- from tumsatz group by jahr, quartal
- ;
- /*
- jahr quartal betrag
- ---------- ---------- ---------------------------------------
- 1991 1 1.10
- 1992 1 2.10
- 1991 2 1.20
- 1992 2 2.20
- 1991 3 1.30
- 1992 3 2.30
- 1991 4 1.40
- 1992 4 2.40
-
- (8 Zeile(n) betroffen)
- */

- --pivoting Daten mit Standard SQl
- SELECT
- tumsatz.jahr as jahr
- ,SUM(CASE WHEN tumsatz.quartal = 1 THEN umsatz END) AS q1
- ,SUM(CASE WHEN tumsatz.quartal = 2 THEN umsatz END) AS q1
- ,SUM(CASE WHEN tumsatz.quartal = 3 THEN umsatz END) AS q1
- ,SUM(CASE WHEN tumsatz.quartal = 4 THEN umsatz END) AS q1
- FROM tumsatz
- GROUP BY tumsatz.jahr
- ;
- /*
- jahr q1 q1 q1 q1
- ---------- -------------- --------------- ------------- --------
- 1991 1.10 1.20 1.30 1.40
- 1992 2.10 2.20 2.30 2.40
-
- */

- select jahr, q1, q2, q3, q4 from tjahrq1234
- ;
- /*
- jahr q1 q1 q1 q1
- ----------- --------------- ---------------- -------------- --------
- 1991 1.10 1.20 1.30 1.40
- 1992 2.10 2.20 2.30 2.40
- */
- /*
- jahr quartal betrag
- ----------- ----------- ---------------------------------------
- 1991 1 1.10
- 1991 2 1.20
- 1991 3 1.30
- 1991 4 1.40
- 1992 1 2.10
- 1992 2 2.20
- 1992 3 2.30
- 1992 4 2.40
-
- (8 Zeile(n) betroffen)
-
- */

```
--unpivoting Daten mit Standard SQl
select
 tjahrq1234.jahr as jahr
,quartal.aaa     as quartal
,CASE quartal.aaa
   WHEN 1 then q1
   WHEN 2 then q2
   WHEN 3 then q3
   WHEN 4 then q4
  END AS betrag
from
 tjahrq1234 cross join
   (select 1 as aaa from dual union
    select 2 as aaa from dual union
    select 3 as aaa from dual union
    select 4 as aaa from dual
  ) quartal
;
```

Jahr und Quartale

- jahr quartal betrag
- ------ ------- ------
- 1991 1 1.1
- 1991 2 1.2
- 1991 3 1.3
- 1991 4 1.4
- 1992 1 2.1
- 1992 2 2.2
- 1992 3 2.3
- 1992 4 2.4
- folgende Liste wird gewünscht:
- jahr q1 q2 q3 q4
- ------ ---- ---- ---- ----
- 1991 1.1 1.2 1.3 1.4
- 1992 2.1 2.2 2.3 2.4
-

- folgende Liste wird gewünscht:
- jahr q1 q2 q3 q4 jahressumme
- ------ ---- ---- ---- ---- -----------
- 1991 1.1 1.2 1.3 1.4 5.0
- 1992 2.1 2.2 2.3 2.4 9.0

- folgende Liste wird gewünscht:
- jahr q1 q2 q3 q4 jahressumme
- ------ ---- ---- ---- ---- -----------
- 1991 1.1 1.2 1.3 1.4 5.0
- 1992 2.1 2.2 2.3 2.4 9.0
- qsum 3.2 3.4 3.6 3.8 14.0

-

15

ANHANG D DER SQL STANDARD

- Der „International Standard ISO/IEC 9075:1992, Database Language SQL“ wurde vom Deutschen Institut für Normung (DIN) unverändert in die deutsche Norm DIN 66315 übernommen.
- Der offizielle SQL Standard ist in den entsprechenden, schwierig zu lesenden Dokumenten beschrieben.
- Eine kritische – heute noch in weiten Teilen mit Gewinn lesbaren – Darstellung aller Aspekte von SQL:1992 finden Sie in folgendem Buch:
- SQL – Der Standard
- SQL/92 mit den Erweiterungen CLI und PSM
- Chris J. Date Hugh Darwen Addison-Wesley 1998
- „Die Sprache SQL besteht aus einer Menge von Einrichtungen zur Definition, zum Zugriff und zur anderweitigen Verwaltung von SQL-Daten (SQL-data)1.
- 1. ... war SQL ursprünglich als Sprache zur Verwaltung relationaler Daten gedacht, aber der Standard hat sich bis zu einem Punkt entwickelt, von dem aus es nun wahrlich ein langer Weg zur Relationalität ist – daher die Terminologie der SQL-Daten.“ Seite 27
- Hinweise auf weitere Bücher finden Sie im Literaturverzeichnis.

Copyright und Impressum

Integrata Cegos GmbH

Zettachring 4
70567 Stuttgart