Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

SQL für DB2

Christoph Thiele & Alexander Kuscher

Übersicht

- DB2 auf Isis
  - Zugriff
  - Command Center
- SQL Grundlagen
  - Tabellen Aufbau
  - Datentypen
- Data Definition Language
  - CREATE / ALTER / DROP
- Data Manipulation Language
  - SELECT / INSERT / UPDATE / DELETE / LIKE / DISTINCT
  - Aggregation & Gruppierung
  - JOIN

Übersicht

- DB2 auf Isis
  - Zugriff
  - Command Center
- SQL Grundlagen
  - Tabellen Aufbau
  - Datentypen
- Data Definition Language
  - CREATE / ALTER / DROP
- Data Manipulation Language
  - SELECT / INSERT / UPDATE / DELETE / LIKE / DISTINCT
  - Aggregation & Gruppierung
  - JOIN

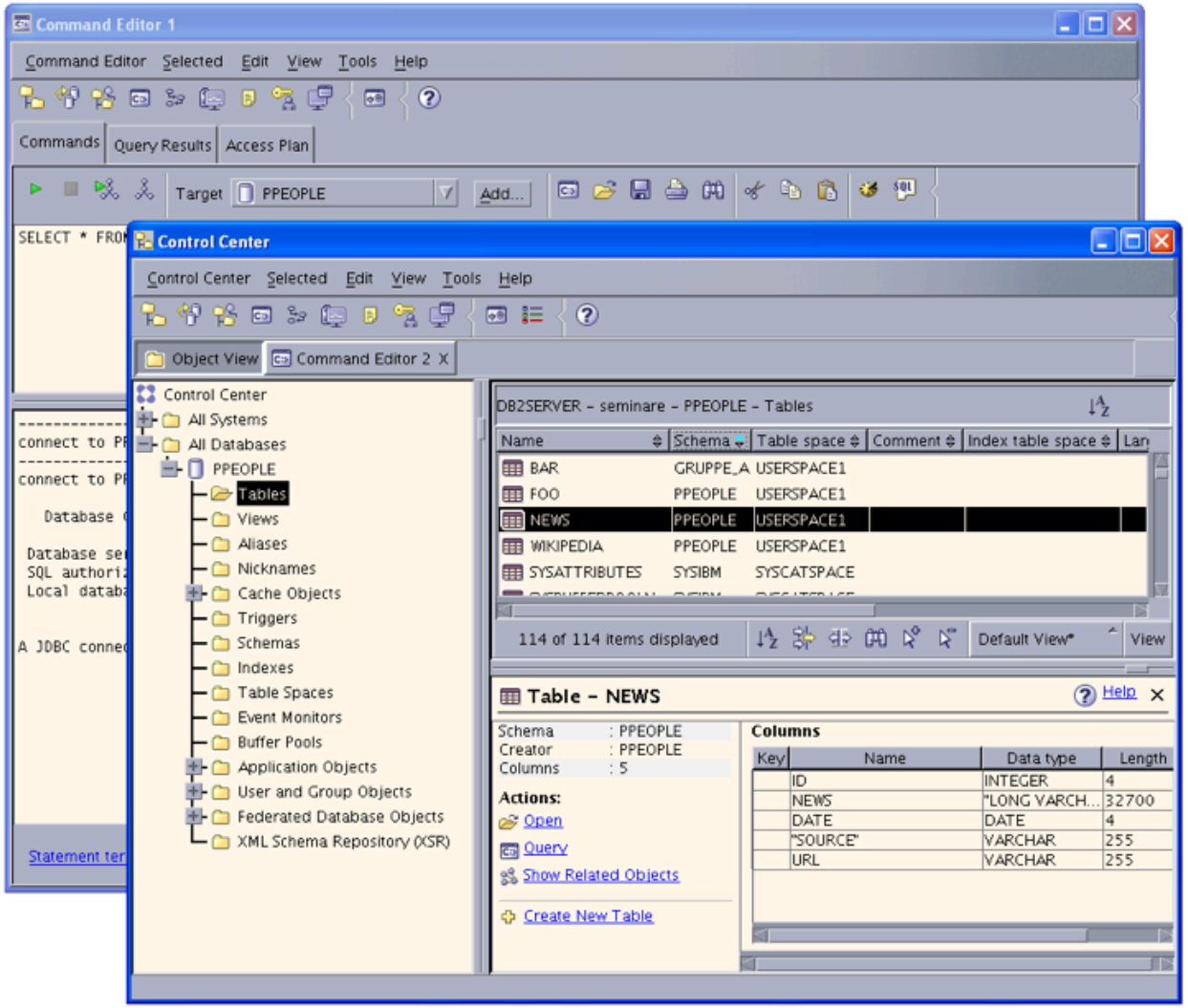
Zugriff auf DB2

- Im Sourcecode Zugriff über
  - `jdbc:db2://isis:50100/PPEOPLE`
- Zugriff auf DB2 Control Center
  - X11 Server (für Windows Cygwin)
  - `ssh -X ppeople@isis`
  - Zugriff von Extern über Studentenserver tunneln
    - `ssh -g -L <localport>:isis:22 placebo.hpi. ... -l <login>`
    - `ssh -X -p <localport> ppeople@localhost`

Auf Isis

- Kommandozeile
  - `db2 “connect to ppeople“`
  - `db2 “<SQL-stmt>“`
  - `db2 “connect reset“`
- DB2 Control Center
  - Wird aufgerufern mit `db2cc [&]`
  - Über X11 wird DB2CC Oberfläche an PC geleitet

DB2 Command Center



Übersicht

- DB2 auf Isis
  - Zugriff
  - Command Center
- SQL Grundlagen
  - Tabellen Aufbau
  - Datentypen
- Data Definition Language
  - CREATE / ALTER / DROP
- Data Manipulation Language
  - SELECT / INSERT / UPDATE / DELETE / LIKE / DISTINCT
  - Aggregation & Gruppierung
  - JOIN

Tabellenaufbau

- Tabellen sind Relationen die auf Tabellen abgebildet werden
  - siehe Vorlesung DBS I
- Spalten haben bestimmte Bedingungen (z.B. Datentypen)

ID	VName	NName	Alter	Geschlecht	Titel
0	Alexander	Kuscher	21	M	
1	Christoph	Thiele	21	M	

ID	Kalenderwoche	Anzahl
0	44	5678
1	44	5678

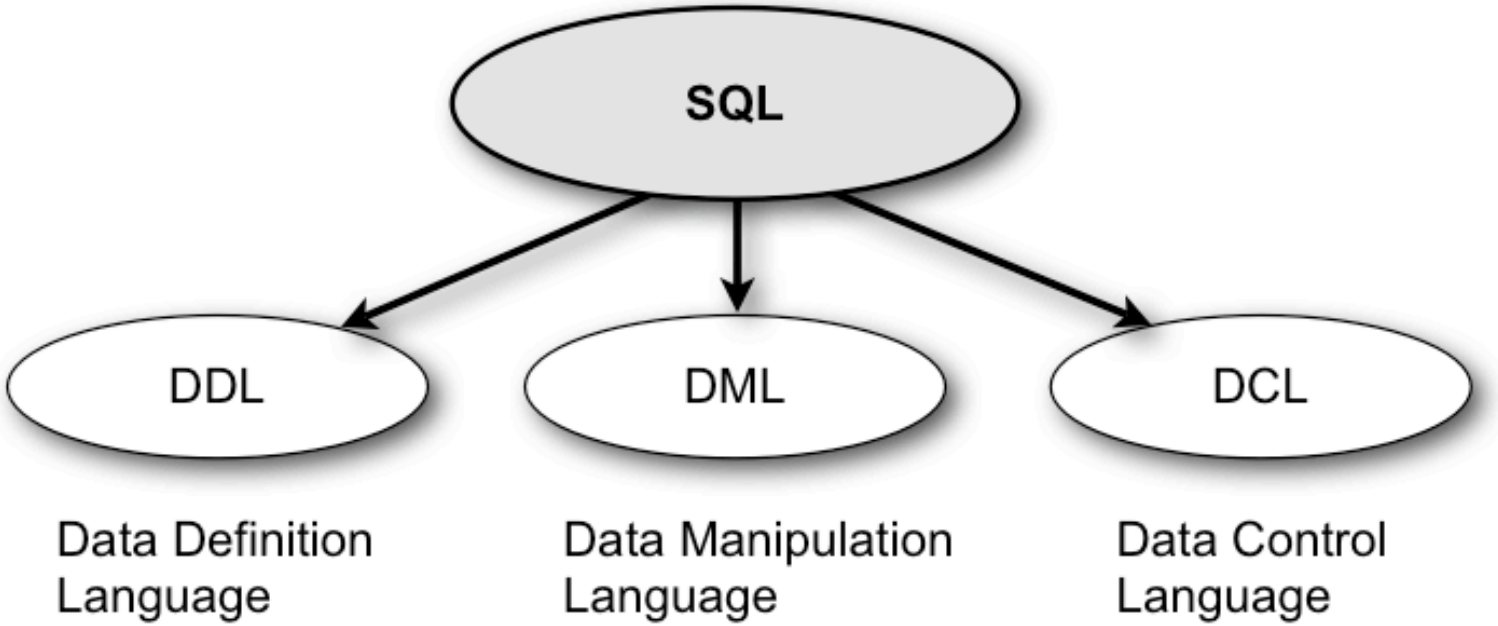
Datentypen

INTEGER	Wertebereich -2^{31} bis $2^{31}-1$
SMALLINT	Wertebereich -2^{15} bis $2^{15}-1$
CHAR [(n)]	n ist fixe Länge
VARCHAR (n)	n ist maximale Länge
BOOLEAN	Konstanten TRUE, FALSE und NULL
DECIMAL [(prec [,scale])]	Festkommazahl
FLOAT [(prec)]	Gleitkommazahl
DATE	Format richtet sich nach Zeitzone
TIMESTEMP	YY-MM-DD-HH.MM.SS

Übersicht

- DB2 auf Isis
  - Zugriff
  - Command Center
- SQL Grundlagen
  - Tabellen Aufbau
  - Datentypen
- Data Definition Language
  - CREATE / ALTER / DROP
- Data Manipulation Language
  - SELECT / INSERT / UPDATE / DELETE / LIKE / DISTINCT
  - Aggregation & Gruppierung
  - JOIN

Sprachbereiche



Übersicht

- DB2 auf Isis
  - Zugriff
  - Command Center
- SQL Grundlagen
  - Tabellen Aufbau
  - Datentypen
- Data Definition Language
  - CREATE / ALTER / DROP
- Data Manipulation Language
  - SELECT / INSERT / UPDATE / DELETE / LIKE / DISTINCT
  - Aggregation & Gruppierung
  - JOIN

Data Definition Language (DDL)

- CREATE TABLE
- ALTER TABLE
- DROP TABLE
- CREATE INDEX
- DROP INDEX
- CREATE VIEW
- DROP VIEW

SQL Grundlagen - DDL

Tabelle erzeugen

- Erstes erstellen einer Tabelle
- Hier schon Möglichkeit Spaltenbedingungen etc. festzulegen

```
CREATE TABLE <Tabellenname>
(<Spaltendefinition>
{, Spaltendefinition}
[, <PRIMARY KEY-Definition>]
{, <UNIQUE-Definition>}
{, <FOREIGN KEY-Definition>}
{, <CHECK-Definition>}
);
```

Tabelle erzeugen

- Beispiel

```
CREATE TABLE VIPS (
   ID          INTEGER        NOT NULL,
   VName       VARCHAR (40)   DEFAULT NULL,
   NName       VARCHAR (40)   NOT NULL,
   'Alter'     INTEGER        DEFAULT NULL,
   Geschlecht  CHAR,
   Titiel      VARCHAR (10)
);
```

SQL-Grundlagen - DDL

Tabelle verändern

- Nachträgliches hinzufügen / entfernen von Spalten / Bedingungen
- Bedingungen Bedingungsname geben / zum Entfernen wichtig

```
<ALTER TABLE-Anweisung ::=
ALTER TABLE <Tabellenname> <ALTER TABLE-Aktion>;

<ALTER TABLE-Aktion>::=
[ADD [COLUMN]] <Spaltendefinition>] |
[DROP [COLUMN]] Spaltenname [RESTRICT|CASCADE]] |
[ADD <Tabellenbedingung>] |
[DROP CONTRAINT Contraintname [RESTRICT|CASCADE]]
```

SQL-Grundlagen - DDL

Tabelle verändern

- Beispiel

```
ALTER TABLE VIPS (
   ADD COLUMN Testspalte
);


ALTER TABLE VIPS (
   ADD CONTRAINT Check_VIP
   CHECK (
        Geschlecht IN (‘M‘, ‘W‘)
);
```

Tabelle löschen

```
<DROP TABLE-Anweisung> ::=
DROP TABLE <Tabellenname> [RESTRICTED | CASCADE];
```

- Restricted ist Standart (muss nicht geschrieben werden)
- Bei Cascade auch Tabellen die einen foreign key in der zu löschenden Tabelle haben

Übersicht

- DB2 auf Isis
  - Zugriff
  - Command Center
- SQL Grundlagen
  - Tabellen Aufbau
  - Datentypen
- Data Definition Language
  - CREATE / ALTER / DROP
- Data Manipulation Language
  - SELECT / INSERT / UPDATE / DELETE / LIKE / DISTINCT
  - Aggregation & Gruppierung
  - JOIN

Data Manipulation Language

- INSERT
- DELETE
- UPDATE
- SELECT
  - DISTINCT
  - LIKE
  - Aggregation
  - Gruppierung
  - JOIN

Einfügen von Daten in die Tabelle

```
INSERT INTO <Tabellenname>
[(Spaltenname {, Spaltenname})] <INSERT-Daten>

      <INSERT-Daten>::=
      VALUES (<Wertespezifikation> {, <Wertespezifikation>})
      {, (<Wertespezifikation> {, Wertespezifikation})}
```

- Bei Spaltendefinition funktionieren trotzdem Default-Werte

Einfügen von Daten in die Tabelle

- Beispiel

```
INSERT INTO VIPS (VNAME, NNAME, Geschlecht)
VALUES
   ('Christph','Thiele','M'),
   ('Alexander','Kuscher','M')
;
```

Daten aus Tabelle löschen

```
DELETE FROM <Tabellenname>
[WHERE <Suchbedingung>]
```

- WHERE Klausel nur weglassen wenn wirklich alles gelöscht werden soll

Daten aus Tabelle löschen

- Beispiel

```
DELETE FROM VIPS;


DELETE FROM    VIPS
WHERE          VNAME = ‘Kuscher‘;
```

Datensatz in der Tabelle aktualisieren

```
UPDATE <Tabellenname>
SET Spaltenname = <Wertespezifikation>
{, Spaltenname = <Wertespezifikation>}
[WHERE <Suchbedingung>]
```

- Bei Wertespezifikation varchar in ` schreiben
- Integer einfach als Zahl ausschreiben
- WHERE um Zeile einzuschränken

Daten aus Tabelle löschen

- Beispiel

```
UPDATE VIPS
SET   VName = ‘Christina‘,
      Titel = NULL
WHERE VName = ‘Tina‘;
```

Data Manipulation Language (2)

Abfragen der Datenbank

- INSERT
- DELETE
- UPDATE
- SELECT
  - DISTINCT
  - LIKE
  - Aggregation
  - Gruppierung
  - JOIN

SQL Grundlagen - DML

Daten aus Tabelle lesen (vertikal)

```
SELECT <Spaltenliste> FROM <Tabellenname>
Spaltenliste::= * | <Spalte> {, <Spalte>}
```

- * selektiert alle Spalten (vollst. Projektion)
- Mittels einer Spaltenliste kann eingeschränkt werden (eingeschr. Projektion)

Aliasbezeichner (Correlation names / AS-Klausel)

- Standart Spaltenname wird in Ergebnistabelle übernommen
- Verkürzen von Namen der Ergebnisspalten
- Verbessert Lesbarkeit
- AS-Bezeichner optional
- Auch für Tabellennamen nutzbar

```
<Werteausdruck> [[AS] Spaltenbezeichner]
```

SELECT & ALIAS Beispiel

- Liefert alle Datensätze und alle Spaltenwerte

```
SELECT * FROM VIPS;
```

- Liefert alle Datensätze mit den Werten der Spalten „VNAME“ und „NNAME“

```
SELECT VNAME, NNAME FROM VIPS;
```

- Benennt Spalten für die Ausgabe um (Alias)

```
SELECT VNAME AS Vorname, NNAME Nachname FROM VIPS;
```

- Gibt der Tabelle einen Alias

```
SELECT VNAME, M.ID FROM VIPS AS V, Mention M;
```

Einschränkungen mit DISTINCT

- Bei Abfragen mittel eingeschränkter Projektion kann es zu Duplikaten (multiset of rows) kommen
- Abhilfe schafft DISTINCT


```
SELECT [ALL|DISTINCT] <Spaltenliste> FROM <Tabellenname>
Spaltenliste::= * | <Spalte {, Spalte}>
```

DISTINCT BSP


```
SELECT KW AS Kalenderwoche FROM Mention;
```

- Liefert 8 Datensätze

```
SELECT DISTINCT KW AS Kalenderwoche FROM Mention;
```

- Liefert 3 Datensätze, da es nur 3 Verschiedene Kalenderwochen in der Beispieltabelle gibt.

Einschränkungen mit WHERE (Restriktion)

- Wird an SELECT-Abfrage angehängt
- Bedingung wird auf abgefragte Datensätze angewendet
- Es gelangen nur Datensätze in Ergebnisstabelle, für die die Suchbedingung TRUE ergibt

```
SELECT <Spaltenliste> FR                    me>
Spaltenliste::= * | <Sp                     >
WHERE <Suchbedingung>
```

an SELECT-Abfrage anhängen

SQL Grundlagen - DML

Vergleichsoperatoren

- =, >, >=, <, <=, <> wie in Mathematik

Logische Operatoren

- NOT, AND, OR (Priorität absteigen)
- Verneingung von Vergleichsoperatoren (NOT=)

- Priorität normal (mon. vor diad., Punkt vor Strichrechnung)

WHERE - Klausel


```
SELECT VNAME, NNAME FROM VIPS
WHERE Alter = 20;

SELECT VNAME, NNAME FROM VIPS
WHERE       (
    Alter = 20              AND
    Geschlecht = ‘M‘        AND
    NOT VNAME = ‘Fabian‘
    ) OR
    NNAME = ‘Thiele‘;
```

Patternmatching

- Mit Hilfe des LIKE-Prädikats Zeichenkette auf Auftreten eines Musters testen
- In WHERE-Prädikat bei SELECT-Abfragen
- Benutzung von Wildcards
  - % für bel. viele bel. Zeichen
  - _ für genau ein bel. Zeichen

```
<Spaltenname> [NOT] LIKE Vergleichsmuster
[ESCAPE] Escapesymbol
```

zum maskieren der Wildcards als normale Symbole

SELECT Beispiel (LIKE)

- Vorname enthällt ‘ist’

```
SELECT VNAME FROM VIPS
WHERE VNAME LIKE ‘%ist%‘;
```

- Vorname endet mit ‘er’

```
SELECT VNAME FROM VIPS
WHERE VNAME LIKE ‘%er‘;
```

Bereichsabfrage (IN) und Wertebereichsabfrage

- BETWEEN-Prädikat
  - Abfrage ob Wertausdruck innerhalb eines best. Intervalls liegt
  - Teil des WHERE-Prädikats

```
<Wertausdruck> [NOT] BETWEEN Untergrenze AND Obergrenze
```

- IN-Prädikat
  - Prüfen ob Werteausdruck zu Menge von Literalen gehört
  - Teil des WHERE-Prädikats

BETWEEN & IN - Prädikat

- BETWEEN - Prädikat

```
SELECT * FROM VIPS WHERE Alter BETWEEN 19 AND 21;
```

- IN - Prädikat

```
SELECT * FROM Mention WHERE KW IN (43, 45);
```

Übersicht

- DB2 auf Isis
  - Zugriff
  - Command Center
- SQL Grundlagen
  - Tabellen Aufbau
  - Datentypen
- Data Definition Language
  - CREATE / ALTER / DROP
- Data Manipulation Language
  - SELECT / INSERT / UPDATE / DELETE / LIKE / DISTINCT
  - Aggregation & Gruppierung
  - JOIN

Aggregationsfunktionen

- Aggregationsfunktionen beziehen sich auf mehrere Datenwerte
- Berechnen u.A. Durchschnitt, Maximalwert, Summe
- Folgende Fkt. sind in DB2 eingebaut:
  - COUNT, AVG, MIN, MAX, SUM
- Gehen Beispielhaft auf COUNT, AVG (und SUM später)

Aggregatsfunktionen

- Anzahl der Datensätze in der Tabelle ‘VIPS’

```
SELECT COUNT(*) FROM VIPS;
```

- Durchschnittsalter

```
SELECT AVG(Alter) FROM VIPS;
```

Gruppierung

- GROUP BY ()
  - Folgt der WHERE-Klausel
  - Gliedert Ergebnisstabelle in Gruppen (Partitioniert diese)
  - Aggregatfunktionen beziehen sich dann auf Gruppen

- HAVING (Gruppenrestriktion)
  - Wie WHERE-Klausel für GROUP BY-Gruppen
  - Nur wenn HAVING-Bedingung TRUE taucht Gruppe auf

GROUP BY - Klausel

- Alle unterschiedlichen Kalenderwochen

```
SELECT DISTINCT KW FROM Mention;
SELECT KW FROM Mention GROUP BY(KW);
```

- Anzahl der Personen je Geschlecht

```
SELECT Geschlecht,COUNT(ID) FROM VIPS GROUP BY(Geschlecht);
```

- Anzahl der Overall-Nennungen je Person (mehr als 5 Nennungen)

```
SELECT ID, SUM(Anzahl) as Anzahl FROM Mention
GROUP BY(ID)
HAVING ( SUM(Anzahl) > 5 );
```

ORDER BY (Sortierung)

- Man kann nicht nur Spaltenweise sortieren
- ORDER BY sortiert Datensätze (Zeilenweise)
- Sortierschlüssel muss nicht in Ergebnisstabelle auftauchen

```
ORDER BY <Sotierschlüssel> [ASC|DESC]
{, <Sortierschlüssel> [ASC|DESC]}
```

Übersicht

- DB2 auf Isis
  - Zugriff
  - Command Center
- SQL Grundlagen
  - Tabellen Aufbau
  - Datentypen
- Data Definition Language
  - CREATE / ALTER / DROP
- Data Manipulation Language
  - SELECT / INSERT / UPDATE / DELETE / LIKE / DISTINCT
  - Aggregation & Gruppierung
  - JOIN

Abfragen über mehrere Tabellen (1)

- Einfaches Kreuzprodukt durch `FROM A, B`
  - Restriktionen möglich
  - Identisch hiermit CROSS JOIN
- Natürlicher innerer Verbund (NATURAL INNER JOIN)
  - Sucht gleiche Spalten
  - Wenn diese im Äquivalenzvergleich TRUE ergeben werden sie verknüpft
  - In DB2 muss NATURAL umschrieben werden

Abfragen über mehrere Tabellen (2)

- Qualifizierter innerer Verbund (INNER JOIN)
  - Zu vergleichende Spalten explizit deklarierbar
  - Anderes Vergleichsprädikat kann genutzt werden
  - Berücksichtigt keine NULL-Daten in Verbundspalten

```
<Tabellenname> [INNNER] JOIN <Tabellenname> <Verbunddef>

<Verbunddef>::=  ON<Suchbedingung> |
                 USING (Spaltenname {, Spaltenname})
```

JOIN Beispiele

OverAll – Nennungen

- kommaseparierte Schreibweise

```
SELECT V.VNAME, SUM(M.Anzahl) AS Anzahl
FROM VIPS V, Mention M
WHERE V.ID = M.ID
GROUP BY(M.ID,V.VNAME);
```

- JOIN

```
SELECT V.VNAME, SUM(M.Anzahl)
FROM VIPS V INNER JOIN Mention M
ON V.ID = M.ID
GROUP BY(M.ID,V.VNAME);
```

Abfragen über mehrere Tabellen (3)

- Äußerer Verbund (OUTER JOIN)
  - Äußerer Verbund berücksichtigt auch NULL-Spalten
  - Natürlich und Qualifiziert wie bei INNER JOIN
  - Es existieren 3 Subtypen
    - (NATURAL) LEFT OUTER JOIN
    - (NATURAL) RIGHT OUTER JOIN
    - (NATURAL) FULL OUTER JOIN

Abfragen über mehrere Tabellen (3)

- Linker äußerer Verbund (LEFT OUTER JOIN)

```
<Tabellenname> LEFT [OUTER] JOIN <Tabellenname>
<Verbunddef>
```

  - Linke Tabelle wird vollständig aufgenommen (Haupttabelle)
  - Rechte Tabelle ist nachgeordnete Tabelle
  - Falls keine Übereinstimmung mit linker Tabelle NULL

- Rechter äußerer Verbund (RIGHT OUTER JOIN)
  - Analog zu Linkem
  - Rechte Tabelle ist Haupttabelle

JOIN Beispiele

- INNER JOIN

```
SELECT V.VNAME, SUM(M.Anzahl)
FROM VIPS V INNER JOIN Mention M
ON V.ID = M.ID
GROUP BY(M.ID,V.VNAME);
```

- OUTER JOIN

```
SELECT V.VNAME, SUM(M.Anzahl)
FROM VIPS V LEFT OUTER JOIN Mention M
ON V.ID = M.ID
GROUP BY(M.ID, V.VNAME);
```

  - Es werden alle Namen berücksichtigt, unabhängig davon, ob Nennungen gespeichert wurden.

Quellen

- SQL Das Praxisbuch, Franzis Verlag, 2003
- http://www.sql-und-xml.de/sql-tutorial/index.html
- http://publib.boulder.ibm.com/infocenter/db2luw/v8//index.jsp