Referat: Datenbanken

von Tobias Schulz

26.11.2008

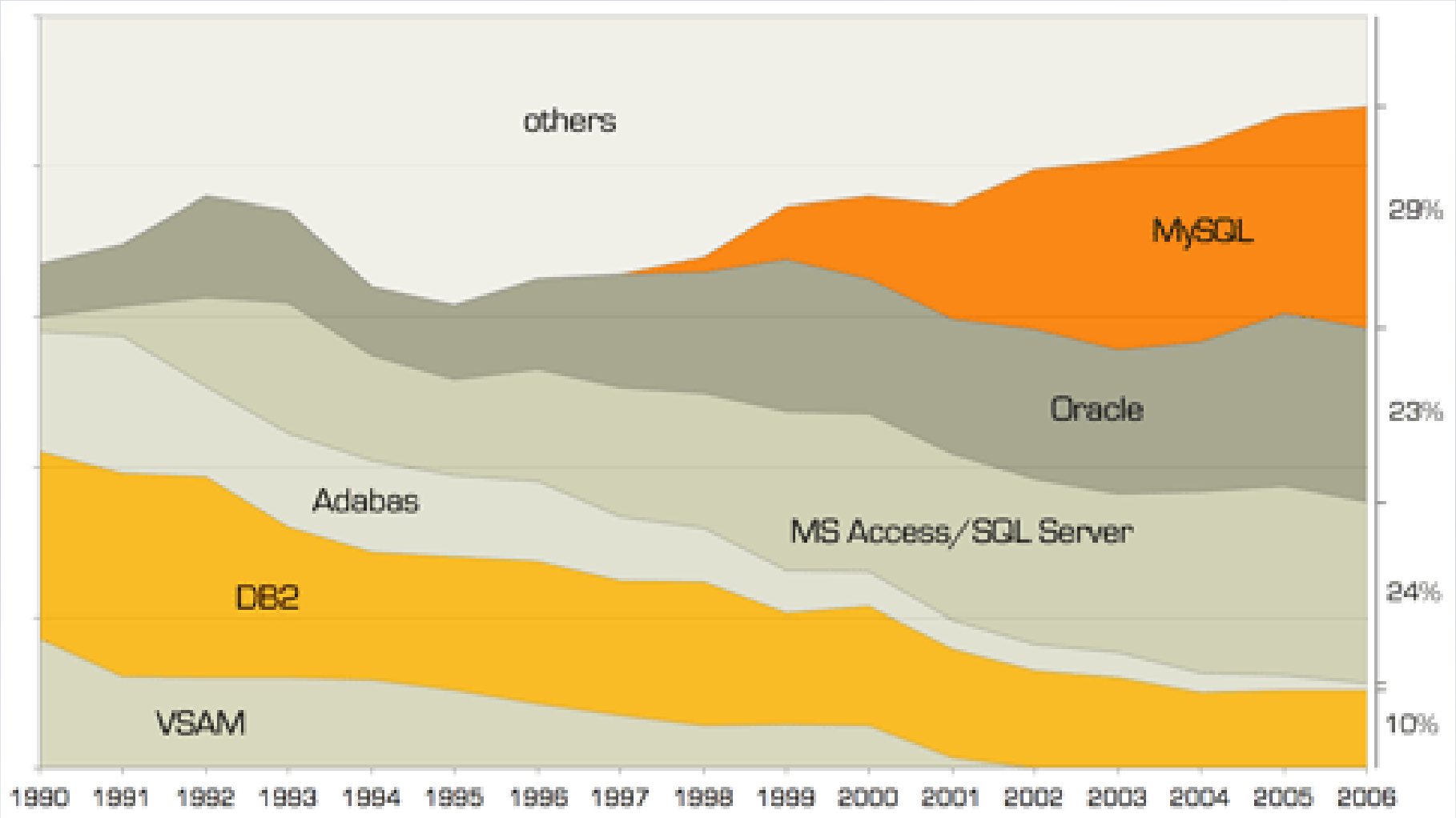
Definitionen

- **DBS:**
  - Die Aufgabe eines Datenbanksystems ist es, Daten konsistent (widerspruchsfrei), effizient und dauerhaft zu speichern
- **DBMS:**
  - Ein Datenbankmanagementsystem ist die Software des Datenbanksystems und übernimmt die Verwaltung der Daten
  - Lese- und Speicherprozesse werden vom DBMS ausgeführt
  - Das DBMS organisiert die Struktur der Daten
  - Das DBMS kann mit einer Sprache wie SQL programmiert werden
- **DB:**
  - Die eigentliche Datenbank enthält die Daten.

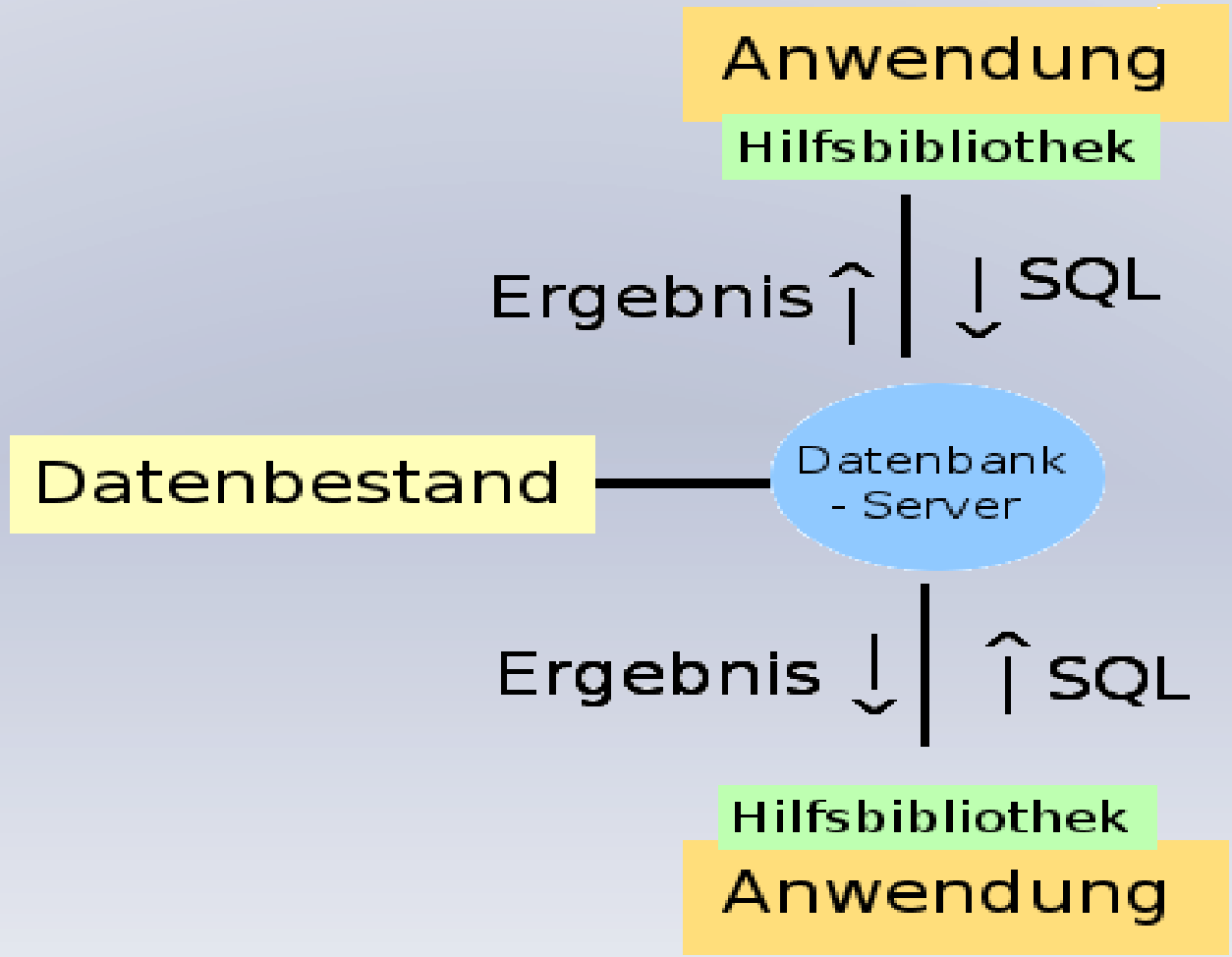
Funktionen von DBMS

- Verwaltung von Datenbeständen
- Einfacher, standardisierter Zugriff auf Daten mittels SQL
- effiziente Zugriffsmechanismen
- Integritätsregeln zur Sicherstellung der Korrektheit der Daten
- Benutzerauthorisierung

Verbreitung der DBMS



Client-Server-Architektur



Datenbankmodelle

- **Relationales Datenbankmodell**
- Hierarchisches Datenbankmodell
- Netzwerkdatenbankmodell
- Objektrelationales Datenbankmodell
- Objektorientiertes Datenbankmodell

Das Relationale Datenbankmodell

- Eine Relation kann durch eine Tabelle beschrieben werden:
  - Die Attributnamen entsprechen den Spaltennamen, die Attributwerte den in den Spalten vorhandenen Einträgen
  - Ein Tupel entspricht einer Zeile einer Tabelle (Datensatz)

Andere Datenbankmodelle

- Hierarchisches Datenbankmodell
  - Hierarchische Baumstruktur
- Netzwerkdatenbankmodell
  - Eine Art Erweiterung des hierarchisches Datenbankmodells, durch das relationale Datenbankmodell weitgehend verdrängt
- Objektrelationales Datenbankmodell
  - Eine Zwischenschicht zwischen einem relationalen Datenbanksystems und einer objektorientierten Software, die objektorientiert auf diese Daten zugreift
- Objektorientiertes Datenbankmodell
  - Daten werden als Objekte im Sinne der objektorintierten Programmierung

SQL (Structured Query Language)

- SQL ist eine Sprache zur Definition, Manipulation und Abfrage von Daten in relationalen Datenbanksystemen
- SQL hat eine einfache Syntax und ähnelt der englischen Sprache

Funktionen von SQL

- Definition der Datenstrukturen
- Hinzufügen, Bearbeiten und Löschen der Daten
- Abfrage der Daten
- Zusätzliche, nicht immer implementierte Funktionen wie:
  - Zugriffsberechtigung
  - Transaktionen (Commit- und Rollback-Funktion)

Erstellen der Datenstruktur (Tabelle)

- Allgemeine Syntax:
  - CREATE TABLE `Tabellenname` (Attribut-Definitionen)
- Beispiel:

```
CREATE TABLE `kunden`
(
   `vorname` varchar(30),
   `name` varchar(30),
   `strasse` varchar(80),
   `plz` integer(10)
)
```

Problem: Redundanz

```
CREATE TABLE `kunden`
(
    `vorname` varchar(30),
    `name` varchar(30),
    `strasse` varchar(80),
    `plz` integer(10)
)
```

Daraus ergibt sich das Problem, dass sich Personen bei dieser Struktur beliebig oft eintragen können.

Beim Einfügen der Datensätze können außerdem Angaben weggelassen werden.

Lösung

```
CREATE TABLE `kunden`
(
   `vorname` varchar(30) NOT NULL,
   `name` varchar(30) NOT NULL,
   `strasse` varchar(80) NOT NULL,
   `plz` integer(10) NOT NULL,
   UNIQUE (`vorname`, `name`, `strasse`, `plz`)
)
```

Datensätze Speichern

- Allgemeine Syntax:

```
INSERT INTO `tabelle` [( `spalten`+ )] DATENKONSTRUKT
```

- DATENKONSTRUKT kann sein:
  - `VALUES ( WERT+ )`
  - Eine beliebige SELECT-Abfrage
- Beispiel:

```
INSERT INTO `kunden`
( `vorname`, `name`, `strasse`, `plz` )
VALUES
( „ABCD“, „EFGH“, „IJKLMNOP Straße“, „010101“ )
```

Daten Abfragen

- Allgemeine Syntax:

SELECT [DISTINCT] Auswahlliste

FROM Tabelle

[WHERE Where-Klausel]

[GROUP BY (Group-by-Attribut)+

[HAVING Having-Klausel]]

[ORDER BY (Sortierungsattribut [ASC|DESC])+]

- Eine solche Abfrage nennt man SFW-“Block“ (SELECT-FROM-WHERE).

Beispiel: Daten Abfragen

```
SELECT ( `vorname`, `name`, `strasse`, `plz` )
FROM `kunden`
WHERE vorname = „ABCD“
ORDER BY name
```

vorname	name	strasse	plz
ABCD	EFGH	IHJK Straße	10101

Mengenoperatoren

- **UNION** vereinigt die davor und danach stehenden Ergebnismengen
  - Doppelte Datensätze werden entfernt
- **UNION ALL** entspricht **UNION**, doppelte Datensätze bleiben erhalten
- **In manchen Dialekten:** **EXCEPT**: Alle Satensätze, die in A, aber nicht in B enthalten sind
- **In anderen Dialekten:** **MINUS** = **EXCEPT**
- **INTERSECT**: Die Schnittmenge von A und B. **INTERSECT** weist das gleiche Verhalten wie **DISTINCT** auf

Beispiel: Mengenoperatoren

- Alle Kunden mit dem Vornamen „ABCD“, deren Nachname nicht mit „E“ beginnt:

```
SELECT ( `vorname`, `name`, `strasse`, `plz` )
FROM `kunden`
WHERE vorname = „ABCD“
ORDER BY name
EXCEPT
SELECT ( `vorname`, `name`, `strasse`, `plz` )
FROM `kunden`
WHERE name LIKE „E%“
```

Optimierung

- Diese Abfrage ist jedoch unnötig kompliziert, verbraucht bei großen Datenmengen mehr Speicher als nötig und ist langsam.
- Optimierung:

```
SELECT ( `vorname`, `name`, `strasse`, `plz` )
FROM `kunden`
WHERE vorname = „ABCD“ AND `name` NOT LIKE „E%“
ORDER BY name
```