Grundlagen der Informatik

Datenbanken-Technik
SQL und Marktübersicht

Prof. Dr.-Ing. Thomas Wiedemann
Fachgebiet Informatik / Mathematik

Überblick zur Vorlesung

Datenbanken I I

- Einführung in SQL
- SQL in C
- Marktübersicht

Einführung in SQL

Die Verwaltung von Daten in einer Datenbank kann auf 2 Arten erfolgen :

- Mit einem **Datenbankmanagementsystem** (wie z.B. Access) können ohne Programmierung Daten über Masken, Tabellen oder Webformulare (z.B. Tool phpmyadmin) eingegeben, geändert und teilweise auch deren Definition in der DB geändert werden
- Für den universellen Einsatz in Programmiersprachen wird ein möglichst standardisierter Zugang zu ALLEN Funktionen ALLER verfügbaren Datenbanken auf Basis einer Programmiersprache benötigt
  - Mit SQL existiert eine solche weitgehend standardisierte Sprache.
  - Fast jede Datenbank kann teilweise oder ganz den SQL-Sprachumfang verstehen und ausführen.
  - Das aufrufende Programm muss „nur“ die SQL-Befehle als Textkommando zusammen bauen und an eine Datenbankschnittstelle senden – die Datenbank analysiert die Befehle und gibt die Ergebnisse (i.d.R. eine Tabelle oder auch ein Einzelwerte) an das Programm zurück

  Tip: Komplexe SQL-Befehle lassen sich günstig über die Tools erzeugen.

Einführung in SQL

- SQL steht für **S**tructured **Q**uery **L**anguage - zu deutsch etwa: **strukturierte Abfragesprache**.

- SQL ist eine **nichtprozedurale Sprache**, d.h. im Gegensatz zu C oder ähnlichen Sprachen beschreibt SQL , **welche Daten** abzurufen, zu löschen oder einzufügen sind und nicht, wie das zu geschehen hat.
- SQL kann in vier Funktionsgruppen unterteilt werden :
  - SQL selbst als Sprache zur Datenabfrage (engl. Query)
  - DML - Data Manipulation Language zur Datenmanipulation (Einfügen, Ändern, Löschen von Daten)
  - DDL - SQL Data Definition Language zur Definition von Tabellen und Abfragen
  - DCL - SQL Data Control Language zur Vergabe von Rechten an Nutzer (spezifische Berechtigung zu den Funktionen)

Historie von SQL

- Entwicklung Ende der 70er Jahre bei IBM-Labor in San Jose (Kalifornien)
- ursprünglich für IBM-Produkt DB2 geplant (DB2 auch heute noch existent)
- Im Gegensatz zu früheren Ansätzen verfolgt SQL einen mengenorientierten Ansatz, d.h. die Operationen der Sprache basieren auf Mengenoperationen (auch mit mathematischer Grundlage) . Damit besteht eine gute theoretische Basis, erstellt durch Arbeiten von Edgar F. Codd mit seiner Forschungsarbeit am IBM Almaden Research Center in den 60 er und 70 iger Jahren .
- Der Erfolg von SQL führte zu einer starken Verbreitung und einer Standardisierung durch das ANSI (American National Standards Institute) und die ISO (International Standards Organization). (z.B. ANSI 86 / 92 / 99 / 2003 / 2006 – SQL – Standard)
- **Resultierende Problem(e) :**
  - Leider sind die Standards (wie oft in der Informatik) sehr umfangreich und werden von Datenbankherstellern nicht zu 100% umgesetzt !
  - Damit besteht trotz SQL leider keine absolute Kompatibilität zwischen den Datenbanken (und wird wahrscheinlich von den Herstellern auch nicht gewünscht)

Programmierung von Datenbankanwendungen

- nicht alle Operationen lassen sich auf einfache Sortier- oder Filteroperationen zurückführen, sondern erfordern algorithmische Lösungen
- zur Steuerung des Datenbankbasissystem aus Programmen heraus wurde durch die führenden Hersteller eine einheitliche Sprache entwickelt :

SQL - die Structured Query Language (dt. strukturierte Abfragesprache)

Funktionsweise von SQL:

- das Programm generiert (=meist Textoperationen) einen Kommandotext und sendet diesen an das Datenbanksystem (häufig als DB-Engine bezeichnet)
- die DB-Engine führt den Befehl aus
- Ergebnisse werden über Pointer oder spezielle Strukturen (Arrays, dyn. Listen) zurück übergeben

Abfrage von Datenbanken mit dem Select-Befehl

- **Abfrage einer Tabelle mit dem Select-Befehl**

Syntax: "Select * from Produkte where Preis > 100 order by PName"

- Select gibt den Wunsch nach einer reinen Anzeige an
- mit dem * werden alle Spalten angezeigt (sonst Spalten aufzählen mit , , , ,)
- der from-Abschnitt definiert die Herkunftstabelle(n),
- der where-Abschnitt definiert einen Filter (hier alle Preise > 100 €),
- der order by -Abschnitt definiert eine Sortierung (hier nach Produktname)
- Es können bei where und order by auch mehrere Angaben erfolgen.
- Als Minimalversion reicht : select * from table

Typische Anwendungen

- Select Vorname, Nachname from ... - Spaltenselektion
- Select ... From Personen Where PID>100 and Alter < 20 - Filter

Abfrage von Datenbanken mit dem Select-Befehl - II

- **Vergleich von Texteinträgen mit dem like – Operator oder =**

unter Access :

- Select ... Where Nachname like "*mann*" - sucht alle Namen mit mann
- ... Where Nachname like "*mann" - sucht alle Namen mit mann am Ende
- ... Where Nachname like "*m?nn" - sucht alle Namen mit Muster m*nn

Achtung – in anderen Datenbanken sind die Platzhalter ggf. anders – **MySQL** :

- Where Nachname like "%mann" - sucht alle Namen mit mann am Ende
- ... Where Nachname like "%m_nn" - sucht alle Namen mit Muster m*nn

Filter über Mengen

- Where Alter >16 **And** Alter < 30
- Where Alter **between** 16 And 30 - analog zu erster Option
- Where Alter **in** (16, 18, 20) - nur die angegebenen Altersgruppen

Abfrage von Datenbanken mit dem Select-Befehl - III

Sortieroptionen

- Select ... Order by Nachname, Vorname – Sortierung immer in der Reihenfolge der Felder von links nach rechts
- ... Order by Alter, PLZ ergibt andere Sort. Als ... Order by Alter, PLZ
- Sortierrichtung von ASC (Ascending – aufsteigend = Default) oder Desc
- ... Order by Alter Desc, PLZ Asc - zuerst nach Alter absteigend, dann bei gleichem Alter nach PLZ aufsteigend

Alias – Namen für Spalten (Um- oder Neubenennung von Spalten)

- SELECT feldName1 "Nachname", feldName2 "Vorname" ...
  - Alias-Vergabe ist sehr hilfreich bei Anpassung verschiedener DB mit unterschiedlichen Namensräumen, ohne daß die Namen in den Ursprungstabellen wirklich geändert werden müssen
- Select [Lagerbestand]*[Preis] AS Lagerwert - auch mit vorheriger Berechnung

Abfrage von Datenbanken mit dem Select-Befehl - 4

- **Verknüpfung von mehreren Tabellen mit JOIN :**

Select * FROM Autor INNER JOIN medien ON Autor.UID = medien.AutorID;

- Der Join verbindet zwei Tabellen über die mit ON-angegebenen Felder (i.d.R Primär- und Fremdschlüssel) – alle Filter/Sortierungen gelten analog
- Bei inner join müssen auf beiden Seiten die entsprechenden Schlüsselwerte vorhanden sein.
- Mit der zusätzlichen Option left join / right join können auch jeweils auf einer Seite Leermengen akzeptiert werden (Vorsicht dann beim Programmieren bei Zugriff auf diese mit NULL belegten Felder)
- Es können auch mehr als 2 Tabellen miteinander verbunden werden – das Verhalten der Datenbanken bzgl. des Schreibens in die Resultatmenge ist dabei aber leider unterschiedlich - Empfehlung : Joins suksessive aufbauen, auch zum Austesten der Ergebnisse

Abfrage von Datenbanken mit dem Select-Befehl - 5

Aggregatfunktionen (Mengenbasierte Rechenfunktionen)

- SELECT COUNT(*) "Anzahl" FROM Autor – liefert Autor-Anzahl
- Als Rechenoperationen stehen COUNT() (Anzahl), MIN(), MAX(), AVG() (Durchschnitt) und SUM() zur Verfügung.
- Mit dem Group by-Befehl können Datensätze zusammengefasst werden :

  SELECT AutorID, **Sum**(Lagerwert) AS [Summe von Lagerwert] ...

  GROUP BY Medien_Autoren.AutorID;

- Mit dem Operator Distinct können auch bei einfachen Tabellen gleiche Datensätze zusammengefasst werden.
- Select **Distinct** Nachname ... - jeden Nachnamen nur einmal

Manipulation von Daten

Einfügen von Daten

INSERT INTO Autor (Nr, Nachname, Vorname, GebJahr)
VALUES (1, 'Böll', 'Heinrich', 1917);
- die genaue Schreibweise ist wieder DB-abhängig (falls ein Feld nicht gesetzt werden soll – NULL angeben)

Ändern von Daten

- **UPDATE** Autor **SET** Vorname = Otto, GebJahr = 1954, Beruf = NULL **WHERE Nr = 10**;
  Achtung : Wichtig ist i.d.R. der Filter, da sonst ALLE Datensätze geändert werden !!!

Daten löschen

- **DELETE FROM** Autor **WHERE** Geburtsdatum < (SYSDATE - 36500);
- hier Löschen von (veralteten) Datensätzen

Datendefinition

Datenbanken definieren :

CREATE TABLE Autor (Nr **INT**, Name **VARCHAR**(80), GebJahr **INT**, Gehalt FLOAT , Geschl **CHAR**(1))

- mit zusätzlichen Angaben (Constraints) können spezielle Eigenschaften der Felder definiert werden :
  - CONSTRAINT ... PRIMARY KEY definiert Primärschlüsselspalten.
    CONSTRAINT ... REFERENCES definiert Fremdschlüsselspalten.
    CONSTRAINT ... NOT NULL erzwingt Eingabewerte

Datenbankfelder ändern oder hinzufügen :

ALTER TABLE Autor **MODIFY** (UID NUMERIC(5)); - Feldgröße ändern

ALTER TABLE Autor **ADD** (Tel NUMERIC(20)) ; - Feld hinzufügen

Datenbanktabelle löschen (Vorsicht – nicht verwechseln mit Delete !)

DROP TABLE Autor ;

Datenbankrechte definieren

Vergabe von Rechten auf der Datenbank :

GRANT SELECT, DELETE, UPDATE, REFERENCES(Nr) **ON** Autor **TO** Mueller;

-> Nutzer Müller darf Daten ändern, aber keine Datenbankdefinition

Entzug von Rechten auf der Datenbank :

REVOKE DELETE **ON** Autor **FROM** Mueller;

Die genaue Funktionalität dieser Funktionen hängt von der DB ab.

Weitere Begriffe und Technologien in der Datenbanktechnik

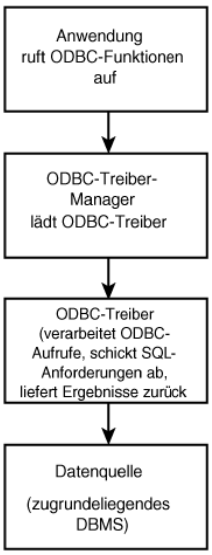
- **Die nachfolgenden Begriff werden nicht von allen DB unterstützt, sondern meist nur von den größeren (und teueren) DB-Systemen.**
- Stored Procedures
  - immer wiederkehrende Abfragen können in der DB in einem internen Format gespeichert werden und sind durch die entfallende Syntaxanalyse schneller bei der Ausführung
  - Zusätzlich ist auch die Manipulationsgefahr (Hacker) geringer (-> Web)
- Trigger
  - Es können Bedingungen (Trigger) auf den Daten definiert werden, bei denen automatisch eine Datenbankaktion ausgelöst wird (z.B. Neuanlage eines Kunden, oder Eintreten von Fehlerzuständen ...)
- Transaktionskontrolle
  - Der Befehl **COMMIT** startet eine Transaktion
  - Alle DML-Befehle (INSERT, UPDATE, DELETE) werden erst beim nächsten **COMMIT** wirklich ausgeführt.
  - Mit **ROLLBACK** kann der gesamte Vorgang rückgängig gemacht werden.

Weitere Begriffe und Technologien in der Datenbanktechnik - 2

Spezielle Datentypen

- **BLOB – Binary Large Objects (dt.: Binäre große Objekte)**
  - zur Speicherung von größeren Datenmengen, z.B. Multimediadaten (Fotos, Musikstücke, verschlüsselte Daten, auch XML-Daten)
  - DB eignen sich jedoch (noch) nicht zur allgemeinen Ablage großer Binärdaten – sinnvoll ist immer eine Verarbeitung (z.B. internes Auslesen der Daten im aufrufenden Programm)
  - Alternativ kann auch ein Pfadname als Text in der DB gespeichert werden und dann extern die Anzeige und Verarbeitung erfolgen
  - immer wiederkehrende Abfragen können in der DB in einem internen Format
- **Datetime, Timestamp**
  - Zur Ablage von Datumsangaben und Zeitstempeln (z.B. letzte Änderung)
  - einige DB erfordern für Änderungen immer einen Zeitstempel
- **Mengen**
  - Enum - Aufzählung von eineindeutigen Werten (“Mo“, “Di“, ..)
  - Set -String-Objekt mit max. 64 Elementen

Weitere Begriffe und Technologien in der Datenbanktechnik - 3



Datenbankschnittstellen - ODBC und JDBC

- Zur universellen Anbindung von DB unter Windows wurde von Microsoft der ODBC- (Open Database Connectivity) Standard eingeführt.
- Es gibt ODBC-Treiber für nahezu jede Datenbank.
- **Vorteil:** Datenbanken sind austauschbar, ohne die Applikation umschreiben zu müssen (falls Standard-SQL oder ANSI 89 verwendet wird).
- **Nachteil:** Performance geringer als bei datenbank-spezifischen APIs
- In ähnlicher Weise bieten JDBC – Treiber eine universelle Schnittstelle zu Java-Programmen an.

Typische C-Programmierung mit SQL

- da der SQL-Befehl als Text an die DB übergeben wird, sind meist Textoperationen notwendig zur Generierung des Befehls
- oft müssen Filterwerte von einer Benutzeroberfläche (Bedienfenster oder Browserrequest an Webserver) in den Befehl integriert werden
- in C-Befehl läßt sich diese Aufgabe sehr elegant mit der sprintf-Funktion (oder auch mit strcat-Funktionen) realisieren

Bsp.: - Annahme - ein bestimmtes Produkt soll mit Details gezeigt werden

- die eindeutige Produktnummer wird dazu als Parameter geliefert :

```
void  zeige_produktdaten (long ProduktID)
{  char sqlbefehl[500] ; /*ganz normaler Text*/ dbdat *p; // DB-Ergebnisse
   sprintf( sqlbefehl , "select * from Produkte where PID=%i" ,  ProduktID) ;
   p =  dbexecute( sqlbefehl );
   printf("Produkt %s kostet %i €", p.attribute("PName"),p.attribute("Preis"));  }
```

Marktübersicht zu Datenbanksystemen

Kommerzielle Datenbanken (meist auch eingeschränkte Freeware-Versionen)

- **Oracle (Marktführer) - Oracle Database 11g**
  - sehr komplexes System, Preise bis zu 20.000 €
  - Führend bzgl. Funktionalität und Performance, harter Kampf gegen Microsoft
  - Neuorientierung auf Produktions-Planungs-Systeme (ERP) als Konkurrenz zu SAP
- **Microsoft** - Access und Microsoft SQL Server
  - MS Access als kleine Bürodatenbank mit bis zu 100.000 Datensätzen, Entwicklungsoberfläche aber sehr effizient
  - bei größeren Datenvolumen Migration auf MS SQL Server sinnvoll
  - MS SQL Server ca. 3000 –6000 € , sehr günstig im Paket (Small Business Server) für Windows-Applikationen
- **IBM**
  - DB2 – ebenfalls sehr leistungsfähiges System
  - Weiterhin bietet IBM mit Informix eine weitere, kommerzielle Datenbank an.

Marktübersicht zu Datenbanksystemen

Freie / Kostenlose Datenbanken

- **MySQL - MySQL 5.1**
  - freie DB der schwedischen Firma MySQL AB , **www.mysql.com**
  - optimiert für Einsatz als Webdatenbank (sehr viele kleine Zugriffe)
  - im Januar 2008 von Sun übernommen
- **PostgreSQL 7.2 ... 8.0**
  - freie DB
  - schnell bei vielen gleichzeitigen Benutzern und komplexeren Operationen;
  - Verarbeitung geographischer Daten (GIS)
- **MaxDB**
  - Nachfolger der SAP® DB (und damit auch von Adabas) und prädestiniert für ERP-Anwendungen.
  - auch gut geeignet für andere komplexe Anwendungen im kommerziellen Umfeld; z.B. Data Mining,

Marktübersicht zu Datenbanksystemen

Marktvolumen bei der kommerziellen DB im Mrd. $

- Gesamtmarkt von ca. 15 Mrd. $

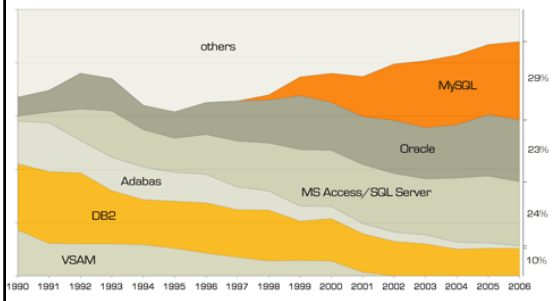
Company	2006	2006 Market Share (%)	2005	2005 Market Share (%)	2005-2006 Growth (%)
Oracle	7,168.0	47.1	6,238.2	46.8	14.9
IBM	3,204.1	21.1	2,945.7	22.1	8.8
Microsoft	2,654.4	17.4	2,073.2	15.6	28.0
Teradata	494.2	3.2	467.6	3.5	5.7
Sybase	486.7	3.2	449.9	3.4	8.2
Other Vendors	1,206.3	7.9	1,149.0	8.6	5.0
Total	**15,213.7**	**100.0**	**13,323.5**	**100.0**	**14.2**

Quelle : Source: Gartner Dataquest (June 2007) - http://www.gartner.com/it/page.jsp?id=507466

*** Teradata : ehemals NCR Systems (vor allem im Handel und Bankwesen)**

Marktübersicht zu Datenbanksystemen

Marktanteil allgemein



Quelle : Mysql AB (!!!)
http://www.mysql.com/why-mysql/marketshare/

Einsatzhäufigkeit
(mit Mehrfachnennungen!)