Welche Datenbank? Warum PostgreSQL?

Open-Source-Tag Magdeburg

Andreas 'ads' Scherbaum

Web: http://andreas.scherbaum.la/
E-Mail: andreas@scherbaum.biz
PGP: 9F67 73D3 43AA B30E CA8F 56E5 3002 8D24 4813 B5FE

11. Oktober 2008

Inhaltsverzeichnis

Welche Datenbank? Warum PostgreSQL?

- Warum noch ein Vergleich von Datenbanken?
- In vielen Fällen sind bloß ein oder zwei Datenbanken bekannt
- Es gibt jedoch vielfältige Alternativen
- Dieser Vortrag stellt einige bekannte Datenbanken vor -
- Und vergleicht diese unter verschiedenen abstrakten Gesichtspunkten:
- Anwender, Anwendung, Zugriffsmethoden, Administration
- Betriebssysteme, Entwickler, Programmiersprachen
- Integrierbarkeit, Sicherheit, BSI
- Und warum dann PostgreSQL?

Der Dozent

- Name: Andreas Scherbaum
- Selbstständig im Bereich Datenbanken, Linux auf Kleingeräten, Entwicklung von Webanwendungen
- Arbeit mit Datenbanken seit 1997, mit PostgreSQL seit 1999
- Gründungsmitglied der Deutschen und der Europäischen PostgreSQL User Group
- Board of Directors - European PostgreSQL User Group

Datenbanken

- Es gab und gibt eine Vielzahl von Datenbanklösungen
- Manche Programme sind wieder vom Markt verschwunden,
- Manche Datenbanken waren nur Nischenprodukte für spezielle Anwendungen
- Andere existieren bis heute
- Frage: Wer kennt mehr als 3 Datenbanken?
- Eine kleine Auswahl soll in diesem Vortrag vorgestellt und verglichen werden

Klassifizierung

- Relationale Datenbank
- Objektrelationale Datenbank
- Objektdatenbank
- Filebasierte Datenbank
- Aber zuerst eine Übersicht:

Aktuell am Markt verfügbare Datenbanken 1/4

- **Oracle Database Server** (Oracle Corporation)
- **DB2** (IBM)
- **Microsoft SQL Server** (Microsoft Corporation)
- **PostgreSQL** (PostgreSQL Global Development Group)
- **Ingres** (Ingres Corporation)
- **MySQL** (MySQL AB/Sun Microsystems)
- **Firebird** (Firebird Project)
- **InterBase** (Borland/CodeGear)

Aktuell am Markt verfügbare Datenbanken 2/4

- **Informix** (IBM)
- **Adabas** (Software AG)
- **Adabas D** (Software AG)
- **MaxDB** (SAP AG)
- **Apache Derby** (Apache Software Foundation)
- **CouchDB** (Apache Software Foundation)
- **Caché** (InterSystems)
- **SQLite** (SQLite-Team)

Aktuell am Markt verfügbare Datenbanken 3/4

- **Berkeley DB** (Oracle)
- **HSQLDB** (Thomas Müller)
- **dBASE** (dBASE dataBased Intelligence, Inc.)
- **xBase**
- **FileMaker** (FileMaker Inc.)
- **FrontBase** (FrontBase, Inc.)
- **Information Management System - IMS** (IBM)
- **Microsoft Jet Engine** (Microsoft Corporation)

Aktuell am Markt verfügbare Datenbanken 4/4

- **Microsoft Access** (Microsoft Corporation)
- **Lotus Notes** (IBM)
- **Paradox**
- **Papyrus** (R.O.M. logicware)
- **Adaptive Server Enterprise** (Sybase)
- **Teradata** (Teradata Corporation)
- **Zope Object Database**
- **Borland Database Engine - BDE** (Borland Software Corporation)

Aktuell am Markt verfügbare Datenbanken

- Das waren 4 Seiten á 8 Datenbanklösungen
- Frage: Wieviele Datenbanken kamen jetzt bekannt vor?
- Es gibt noch eine Reihe weiterer Produkte, die hier nicht aufgeführt sind
- Einige der aufgeführten Datenbanken im Detail:

Die wichtigsten: Oracle Database Server (Oracle Corporation)

- Speichert relationale und objektrelationale sowie XML-Daten
- Skaliert sehr gut, unterstützt große Datenmengen und viele Plattformen
- Bietet Data Warehouse Funktionen und umfangreiche Tuningmöglichkeiten
- Erweiterungen für Geodaten und Volltextsuche verfügbar
- Eine der drei am meisten verbreiteten Datenbanken, erste Version 1977
- Lizensierung erfolgt nach Anzahl User oder Anzahl Prozessoren, kostenlose Express Edition verfügbar
- Weltweit viele Usergruppen vorhanden, die auch auf Messen und Veranstaltungen auftreten

Die wichtigsten: DB2 (IBM)

- Einsatz auf Mainframes, Unix, Linux und Windows
- Skaliert sehr gut, unterstützt große Datenmengen
- Eine der ersten verfügbaren relationalen Datenbanken, erlaubt auch objektrelationales Arbeiten
- Unterstützt XML, Data Warehousing, Geodaten und Volltextsuche
- Eine der drei am meisten verbreiteten Datenbanken, basiert auf System R, erste Entwicklungen 1975
- Lizensierung nach Usern, Prozessoren, Servern, pro Anwendung, ..., kostenlose Express-C Version verfügbar
- Weltweit diverse Usergruppen vorhanden

Die wichtigsten: Microsoft SQL Server (Microsoft Corporation)

- Entwicklung erfolgte um 1989 zusammen mit **Sybase** für OS/2, erster **MS-SQL Server** v4.2 um 1992, völlige Neuentwicklung für Version 7
- Erlaubt große Datenbestände, unterstützt Volltextsuche und Data Warehouse, seit neuestem auch Geodaten
- Nur für Windows verfügbar, gute Integration in andere Windows Anwendungen und Entwicklungsumgebungen von Microsoft
- Eine der drei am meisten verbreiteten Datenbanken
- Lizenzen pro Rechner/Server, pro User/Client und pro Prozessor, kostenlose Express Edition verfügbar

Die wichtigsten: PostgreSQL (PostgreSQL Global Development Group)

- Entstand aus **Ingres**, der Name bedeutet Post-Ingres-SQL
- Unterstützt viele Betriebssysteme, große Datenmengen und Volltextsuche
- Weitgehende Einhaltung des SQL-Standards
- Ist unter der BSD-Lizenz verfügbar, kostenfreier Einsatz in eigenen Anwendungen ist möglich und erwünscht
- Weltweit viele aktive Usergruppen

Die wichtigsten: MySQL (MySQL AB/Sun Microsystems)

- Entstand als kleine Datenbank um den Overhead der "großen" Datenbanken zu vermeiden
- Unterstützt viele Betriebssysteme und mittelgroße Datenmengen
- Volltextsuche und Transaktionen nur für einige Tabellentypen
- Duale Lizensierung: GPL und Lizenz für Integration in eigene Anwendungen
- **Drizzle**: ein Fork von **MySQL 6** zum Wiederholen der gleichen Fehler

Die wichtigsten: Firebird/InterBase (Firebird Team/Borland)

- **InterBase** wurde von **Borland** im Jahr 2000 freigegeben und wird seitdem parallel als Firebird entwickelt
- Unterstützt Unix, Linux, Windows (**InterBase** nur Solaris, Linux, Windows)
- Folgt weitgehend dem SQL-Standard, einige Fähigkeiten fehlen jedoch
- Große Datenmengen werden unterstützt sowie Features wie Read-Only Datenbanken oder gesplittete Datendateien
- **Firebird** hat eine aktive Community, die sich auch um die Weiterentwicklung kümmert

Die wichtigsten: Informix (IBM)

- Bietet gute Performance bei geringem Wartungs- und Administrationsaufwand
- **IBM** kaufte 2001 **Informix** und integriert Funktionalitäten in **DB2** um Kunden zum Wechseln zu bewegen

Die wichtigsten: Adabas/Adabas D/MaxDB (Software AG/SAP AG)

- **Adabas** entstand Anfang der 70er Jahre und wurde zuerst auf Großrechnern eingesetzt
- Sehr performant, unterstützt große Datenmengen, jedoch nicht einfach zu nutzen
- Unterstützt Mainframes, Unix, Linux, Windows
- **Adabas D** entstand Ende der 70er Jahre aus Arbeiten der **TU Berlin** und **Nixdorf**
- **SAP** übernahm die Datenbank 1997, entwickelte daraus **SAP DB** und später **MaxDB** (bei **MySQL** 2003-2007)
- Ausgelegt für Data Warehouse, unterstützt große Datenmengen und skaliert gut

Die wichtigsten: Apache Derby (Apache Software Foundation)

- Komplett in **Java** geschrieben, das fertige Programm ist recht klein (~2 MB)
- Datenbanken sind binär portabel auf allen unterstützten Plattformen

Die wichtigsten: CouchDB/Lotus Notes (Apache Software Foundation/IBM)

- Dokumentorientierte Datenbanken (nicht relational)
- **CouchDB** ist in der Entwicklung, Ziel ist hohe Skalierbarkeit und Zuverlässigkeit bei großen Datenmengen
- Nutzt **Apache Lucene** für Volltextsuche in Dokumenten
- **Lotus Notes** ist ein verteiltes Datenbanksystem mit starker Orientierung auf Dokumente - z. B. E-Mails
- Entstand Anfang der 70er Jahre aus den ersten Message-Boards
- Der **Domino-Server** ist sowohl Datenbank-Server wie auch Mail-, Kalender- und Webserver
- Ein Bearbeiten der Datenbanken ist offline möglich, Änderungen werden später repliziert

Die wichtigsten: SQLite/Berkeley DB/HSQLDB (SQLite-Team/Oracle/Thomas Müller)

Dateibasierte Datenbanken, die Anwendung integriert die Datenbank (**HSQLDB** auch als Server möglich)

- **SQLite** bietet einen SQL Dialog mit eingeschränkten Fähigkeiten (einfache Datentypen, kein Mehrbenutzerbetrieb)
- Einsatz in: Mozilla Browser, Symbian, Apple Safari, Apple Mail, Photoshop Lightroom, Google Gears, Google Desktop, Google Android, McAfee Antivirus, Philips MP3 Player, Skype

Die wichtigsten: SQLite/Berkeley DB/HSQLDB (SQLite-Team/Oracle/Thomas Müller)

- **Berkeley DB** ist eine Key/Value-Datenbank, bietet jedoch Transaktionen, Locking, Hot-Backup, parallele Zugriffe
- Einsatz in: Movable Type (Blog Software), OpenLDAP, Subversion, Motorola Smartphone, DBD-Engine in MySQL, Amazon Frontent, Juniper und Cisco Geräte
- **HSQLDB** ist in **Java** geschrieben, bietet SQL als Zugriffssprache, Standalone sowie Server-Mode
- Einsatz in: **OpenOffice.org 2.0**, Mathematica, JBoss, Hibernate

Die wichtigsten: Microsoft Access/Microsoft Jet Engine (Microsoft Corporation)

- **Jet Red** ist für kleinere Datenbanken gedacht und bietet eingeschränkten Mehrbenutzerbetrieb
- Ist Teil anderer Anwendungen, z. B. MS Access oder MS Visual Basic
- **Jet Blue** ist auf Mehrbenutzerbetrieb ausgelegt, bietet Transaktionen und weitere Fähigkeiten relationaler Datenbanken
- Wird z. B. im Exchange Server oder in Active Directory eingesetzt

Die wichtigsten: LDAP (diverse Anbieter)

- **LDAP**: Lightweight Directory Access Protocol
- Bildet einen Verzeichnisdienst für eine im Netzwerk verteilbare hierarchische Datenbank
- Ist eine Teilmenge von X.500/DAP, welches sehr schwer zu implementieren ist
- Die Struktur ist ein hierarchischer Baum mit Wurzeln, Zweigen und Blättern, jeder Datensatz wird als Objekt abgelegt
- LDAP ist auf Auslesen optimiert und erlaubt viele Tausend Anfragen in der Sekunde
- Anwendung: Benutzerverwaltung, Authentifizierung, Adressbuch

Vergleich verschiedener Datenbanken

- Ein Vergleich verschiedener Datenbanken blendet immer wesentliche Eigenschaften irgendeiner Datenbank aus
- Vergleiche werden unter verschiedenen möglichen Gesichtspunkten angestellt
- Die ausgewählte Datenbank sollte möglichst viele der geforderten Kriterien unterstützen

Kleingedrucktes

Eine in den folgenden Vergleichen nicht aufgeführte Datenbank bedeutet nicht automatisch, das diese Datenbank ein Feature nicht unterstützt.
Entweder waren Informationen dafür nicht zu finden oder die Umsetzung eines konkret geforderten Features ist nur umständlich möglich.

Webanwendungen

- Client-Server Prinzip (bis auf wenige Ausnahmen) gefordert
- Unterstützung in gängigen Scriptsprachen für schnelle Entwicklung gewünscht

Webanwendungen - Scriptsprachen

- **PHP** (nur SQL): MySQL, PostgreSQL, Oracle, DB2, MS-SQL, SQLite, Sybase, Informix, Firebird, InterBase, (ODBC)
- **PHP PDO**: MySQL, PostgreSQL, Oracle, DB2, MS-SQL, SQLite, Firebird, InterBase, (ODBC)
- **Perl DBD**: quasi alle bekannten Datenbanken und Formate
- **Python**: MySQL, PostgreSQL, Oracle, MS-SQL, SAP DB, DB2, Firebird, InterBase, Informix, Ingres, Sybase, SQLite, (ODBC)
- **Ruby DBI**: MySQL, PostgreSQL, Oracle, DB2, Frontbase, SQLite, (ODBC)
- **Ruby on Rails**: MySQL, PostgreSQL, Oracle, DB2, MS-SQL, Sybase, Firebird, InterBase, SQLite (derzeit kein ODBC)
- **ASP.NET/C#**: über ODBC

Webanwendungen - Zusammenfassung

- Unter Unix lauffähige Scriptsprachen unterstützen die gesamte Palette relationaler Datenbanken
- Unter Windows lauffähige Scriptsprachen werden per **ODBC** angesprochen
- Der Schwerpunkt bei Webanwendungen liegt auf relationalen Datenbanken
- Gleich danach kommen eingebettete Datenbanken bzw. -formate (CSV, XML, SQLite)

In ein Programm integriert

Dies betriff Anwendungen die eine Datenbank integrieren, ohne auf einen externen Dienst zurückzugreifen

- **Vorteile**:
- Kein extra Datenbankprozess bzw. -service notwendig
- Geringer Mehraufwand bei der Entwicklung
- Das komplette Programm kann ausgeliefert werden (keine zusätzliche Software notwendig)
- Datenbank liegt als Datei vor
- **Nachteile**:
- Möglicherweise schlechtere Performance (kein dedizierter Server)
- Zugriffskontrolle nicht gegeben
- **Hinweis**: Embedded SQL ist etwas anderes

In ein Programm integriert - Datenbanken

- **SQL**: SQLite, HSQLDB, Microsoft Access, Microsoft Jet Engine
- **Embedded**: Lotus Notes, Berkeley DB, Apache Derby, Borland Database Engine, xBase, Paradox, Filemaker, dBase

In ein Programm integriert - Zusammenfassung

- Es gibt viele verschiedene Möglichkeiten, eine Datenbank direkt in die Anwendung einzubinden
- Die Nutzung von SQL als Anfragesprache erleichtert das Portieren auf andere Datenbanken

Mehrbenutzerbetrieb

- **Vorteile**:
- Mehrere Benutzer können gleichzeitig an den Daten arbeiten
- Daten können ggf. auf einem anderen Server liegen
- **Nachteile**:
- Locking-Mechanismen sind notwendig (auf Datei- oder Datenbankebene)

Mehrbenutzerbetrieb - Datenbanken

- Alle SQL Datenbanken (entweder als Client-Server oder Embedded)
- Apache Derby, Couch DB, Lotus Notes, Berkeley DB, Zope DB, Paradox, IMS, Caché

Mehrbenutzerbetrieb - Zusammenfassung

- Beim Mehrbenutzerbetrieb sollte man auf die Performance bei parallelen Anfragen achten
- Wenn ganze Tabellen oder Dateien für eine Änderung geLOCKt werden müssen bremst dies andere Anfragen aus
- Typische massiv parallele Anwendungen wie Webseiten erfordern teilweise viele hundert parallele Zugriffe, dies muss bei der Auswahl der Datenbank berücksichtigt werden

Speichern binärer Daten

- Binäre Daten (Grafiken, Dokumente, Audiodateien) enthalten Zeichen außerhalb des üblichen ASCII-Zeichensatzes
- Speziell die Verarbeitung von \0 (binär Null) bereitet in Sprachen wie C Probleme
- Datenbanken kennen einen speziellen Datentyp (Blob, Bytea) der eine Längenangabe des Strings enthält:

binärer Datentyp:
Feld 1: Länge der Daten
Feld 2: Daten

Speichern binärer Daten - Datenbanken

- Alle Dokumentbasierten Datenbanken: CouchDB, Lotus Notes
- Informix, Oracle, DB2, MySQL, PostgreSQL, MS-SQL, Firebird, InterBase, SQLite, Adabas, Adabase D, MaxDB, Apache Derby, Berkeley DB, HSQLDB, Microsoft Access (über OLE), Caché

Speichern binärer Daten - Zusammenfassung

- Einige Datenbanken unterstützen direkt das Ablegen von Dokumenten - und damit binäre Daten
- Einige Datenbanken erfordern keine besondere Aufmerksamkeit (z. B. Berkeley DB)
- Die meisten Datenbanken erfordern kleine Umwege in der Programmierung (z.B. binärer Datentyp und Nutzung anderer Funktionen)
- Viele Datenbanken beschränken sich auf das einfache Speichern und Abrufen binärer Objekte
- Geo- oder Wetterdaten sowie Objekte in objektorientierten Datenbanken werden schon heute direkt in der Datenbank verarbeitet
- Die Erkenntnis über eine "intelligente Datenbank" setzt sich jedoch nur schwer durch

Speichern von Objekten

- Objekte eines Programms (objektorientierte Programmierung) müssen nicht umständlich serialisiert und als String gespeichert werden
- Der direkte Zugriff auf Teile des Objekts in der Datenbank ist möglich (über Methoden des Objekts)
- Nachteil: Performanceprobleme bei komplexen Objekten oder umfangreichen Suchen
- Teile der Ideen wurden in SQL:99 für objektrelationale Datenbanken aufgegriffen

Speichern von Objekten - Datenbanken

- Caché, Zope Object DB, ObjectStore, GemStone (Smalltalk), Enterprise Objects Framework (NeXT)

Speichern von Objekten - Zusammenfassung

- Objektdatenbanken schließen eine bestehende Lücke zwischen objektorientierter Programmierung und relationalen Datenbanken
- Objekte werden in der Datenbank gekapselt und der Zugriff findet über die Methoden des Objekts statt
- Späte Bindung (late binding), Persistenz, Objektklassen und Vererbung sind möglich
- Durch den Einsatz objektrelationaler Datenbanken wird die Anwendung relationaler Datenbanken ermöglicht (zumeist eine objektorientierte Schicht auf relationaler Datenbank aufgesetzt)

Daten auf verschiedene Festplatte verteilen

- Größere Datenbanken (Gigabyte- oder Terrabytebereich) benötigen mehrere Datenträger
- Workarounds auf Betriebssystemebene (LVM, RAID) möglich
- Nachteil: keine gezielte Verteilung der Daten anhand bekannter Performancekriterien möglich
- Für SQL-Datenbanken heißt die Lösung: Tablespace (DBSpace, ...)

Daten auf der Festplatte verteilen - Datenbanken

- Oracle, DB2, MySQL (nur NDB), PostgreSQL, Informix, MS-SQL, Adabas

Daten auf der Festplatte verteilen - Zusammenfassung

- Nur "ausgewachsene" Datenbanken unterstützen verteilte Datenbereiche
- Die Notwendigkeit besteht nur bei großen Datenmengen und häufigen Zugriffen
- Workarounds auf Betriebssystemebene sind möglich (LWM, RAID, symbolische Links)
- Wird der Einsatz von Tablespaces notwendig sind bei geschickter Anwendung spürbare Performance Verbesserungen möglich

Replikation

- Replikation wird als Backup und zur Lastverteilung eingesetzt
- Große Datenbanken lassen sich in vertretbarer Zeit nicht komplett sichern
- Daher ist eine Replikation ein gangbarer Weg zum Sichern aller Änderungen
- Replikation soll im laufenden Betrieb ohne Unterbrechung möglich sein
- Es wird zwischen synchroner und asynchroner Replikation unterschieden
- Es wird zwischen Multi-Master und Master-Slave Replikation unterschieden

Replikation - synchroner und asynchroner Replikation

- **Synchron**:
- Vorteil: alle Datenbanken sind immer auf dem gleichen Stand
- Nachteil: Langsamer, da auf Bestätigung aller Datenbanken gewartet werden muss
- Ein Commit-Protokoll zum Gewährleisten der Atomarität ist zu implementieren
- **Asynchron**:
- Vorteil: Schneller da die Änderungen bloß auf einer Datenbank committed werden
- Nachteil: die anderen Datenbanken sind nicht immer aktuell
- Nachteil: Konflikte müssen aufgelöst werden

Replikation - Multi-Master und Master-Slave Replikation

- **Multi-Master** (Master-Master):
- Vorteil: in jede Datenbank kann geschrieben werden
- Nachteil: Konfliktfälle zwischen zwei Datenbanken müssen aufgelöst werden
- Sehr schwer zu implementieren
- **Master-Slave** (Single-Master):
- Vorteil: nur in eine Datenbank werden Änderungen geschrieben
- Nachteil: Verbindungen zu mindestens zwei Datenbanken sind notwendig
- Relativ einfach zu implementieren

Replikation - Datenbanken

- **Multi-Master Replikation**: Oracle, Ingres, MySQL (nicht fehlertolerant), PostgreSQL (mit Einschränkungen), MS-SQL, Microsoft Active Directory, (DBReplicator)
- **Master-Slave Replikation**: Oracle, DB2, MySQL, PostgreSQL, InterBase, Firebird, Adabas, Adabas D, MaxDB, Lotus Notes, ...
- **Dateibasiert**: Berkeley DB

Replikation - Zusammenfassung

- Die Erfordernisse für eine Replikation sind vorab genau festzulegen
- Nicht jede Datenbank unterstützt alle Features - oder implementiert diese komplett

Weitere Hinweise

Die folgenden Folien geben weitere Hinweise zu Auswahlkriterien, ohne konkret auf einzelne Datenbanken einzugehen.

Portierbarkeit

Ein weitgefasster Begriff ...

- Für kommerzielle Anwendungen ist große Anzahl unterstützter Betriebssysteme gewünscht
- Probleme durch Speziallösungen je Plattform kosten Zeit bei der Entwicklung und Geld für den Support
- Sowohl Datenbank wie Programmiersprache sollten auf allen Zielanwendungen vorhanden sein

Release Cycles und Verfügbarkeit von Updates

Ein oft unterschätzter Punkt

- Die Lebensdauer der eigenen Anwendung orientiert sich zwangsweise an der Verfügbarkeit der eingesetzten Datenbank+Version
- Dabei sind auch Betriebssystemupdates und damit eventuelle Updates zu betrachten (speziell bei Embedded Datenbanken)

Betriebssystem

- Für die Privatnutzung ist bloss relevant, ob die Datenbank das Betriebssystem daheim unterstützt
- In Unternehmen existieren homogene Betriebssystemlandschaften mit oft auch mehreren Versionen eines OS
- Bei der Auswahl einer passenden Datenbank muss sowohl die Serverlandschaft (ggf. Remote Administration) wie die Zusammenstellung der Entwicklerarbeitsplätze berücksichtigt werden

Anforderungen des BSI

Im folgenden werden einige zutreffende Anforderungen aus dem Grundschutzhandbuch des BSI aufgeführt.
Aufgrund der GSHB Kriterien werden Datenbanken ausgeschlossen die den Kriterien nicht entsprechen.

Sicherheitskonzept (M 2.126)

Daten sind das Kapital eines Unternehmens - Warum werden Daten so schlecht geschützt?

- Abgrenzung der Zugriffsrechte (wer darf direkt, wer über Applikationen auf die Datenbank zugreifen)
- Speicherung der Daten, Datensicherung
- Kontrolle der Datenbank (Funktion, Aktivitäten, Kapazität)
- Datenschutzrichtlinien müssen erstellt, durchgesetzt und kontrolliert werden

Das Sicherheitskonzept für die Datenbank(en) kann nur in Zusammenarbeit mit dem Sicherheitskonzept des Unternehmens funktionieren.

Auswahl einer Datenbank-Software (M 2.124)

- Integration in das Sicherheitskonzept
- Identifikation und Authentifizierung der Nutzer muss möglich sein
- Einschränken der Zugriffsrechte muss möglich sein
- Datensicherheit muss gewährleistet sein (ACID, Backup)
- Kontrolle durch die Revision muss möglich sein

Anforderungen für die Programmierung (M 2.134)

- Views und Proceduren verwenden, nicht auf Tabellen direkt zugreifen
- Alle Spalten explizit angeben, "SELECT *" vermeiden
- Transaktionen nutzen, ggf. vorhandenes Autocommit abschalten
- Fehlerstatus nach jeder Operation prüfen

Backup und Restore (M 6.49)

- Relevante Daten müssen regelmäßig gesichert werden (welche Verluste sind vertretbar?)
- Die richtige Funktion des Backups muss regelmäßig geprüft werden
- Die sichere Verwahrung des Backups muss gewährleistet sein
- Notfallpläne für die Wiederherstellung müssen vorhanden sein

Protokollierung und Kontrolle (M 2.64/M 2.110/M 2.133)

- Protokollieren sicherheitsrelevanter Ereignisse
- Protokoll ohne Auswertung ist sinnlos
- Regelmäßiges Löschen der Protokolldateien
- Der Zugriff auf die Protokolldateien muss geregelt sein
- Datenschutzbestimmungen müssen eingehalten werden

Accounts und Administratoren (M 2.31/M 2.128/M 2.129/ M 2.131)

- Accounts zugelassener Benutzer (und deren Rechte) müssen dokumentiert sein
- Accounts müssen regelmäßig auf Aktualität geprüft werden
- Zugang zur Datenbank (remote/lokal) muss protokolliert werden
- Restriktiver Zugang zu den Daten muss gewährleistet sein, niemand benötigt Zugriff auf alle Daten
- Administrationstätigkeiten müssen aufgeteilt sein (Admin+Vertretung) und protokolliert werden

Zusammenfassung

Es gibt nicht **die** Datenbank. Ein Blick über den sprichwörtlichen Tellerrand lohnt sich immer.
Dabei ist vor allem eine genaue Planung der eigenen Wünsche und Anforderungen notwendig.

Warum PostgreSQL?

Wenn Sie eine SQL-fähige Datenbank für kleine oder große Aufgaben benötigen, sollten Sie **PostgreSQL** in Ihre Auswahl aufnehmen

Warum?

Warum PostgreSQL? - Das spricht dafür

- **PostgreSQL** positioniert sich im Bereich der relationalen Datenbanken im oberen Drittel
- Im Bereich der Open Source Datenbanken im Spitzenfeld
- Bietet relationale und objektrelationale Features
- Steht unter der BSD-Lizenz, erlaubt damit die Integration in eigene Anwendungen
- Läuft auf allen gängigen Plattformen
- Skaliert linear sowohl bei steigender CPU-Zahl wie bei der Anzahl paralleler Verbindungen
- Viele Firmen bieten kommerziellen Support
- Die Community ist weltweit sehr aktiv

Warum PostgreSQL? - Features

- Features: Tablespaces, ausgefeilter Query Planer, Datenkomprimierung, Unterstützung für binäre Daten, ACID Konformität, Constraints, Stored Procedures, Trigger, Replikation, Schemata, viele vorhandene sowie selbstdefinierbare Datentypen, XML, Subselects - auch geschachtelt, MVCC, beliebig grpße Datenbanken, Views, SP in vielen verschiedenen Programmiersprachen, GIS - Geoinformationssysteme, Cursor, Plugins, PITR - Point in Time Recovery, Two Phase Commit, Warm Standby, LDAP und GSSAPI Anbindung, SSL und IPv6 Support
- Indextypen: Funktional/Expression, Partiell, Multiple, Multi-column, Bitmap, Non-blocking, Volltext

Warum PostgreSQL? - Was denken andere

- Best Database: LinuxWorld Editor's Choice (1999), Linux Journal Editors' Choice (2000), Linux New Media Editors Choice (2002), Linux Journal Editors' Choice (2003), Linux New Media (2004), Linux Journal Editors' Choice (2004), ArsTechnica Best Server Application (2004), Linux Journal Editors' Choice (2005), Linux Journal Editors' Choice (2006),
- Developer.com Product of the Year, Database Tool

Warum PostgreSQL? - Wer nutzt diese Datenbank?

- Durch die freizügige BSD-Lizenz ist niemand gezwungen, die Nutzung in seinem Produkt bekanntzugeben :-(
- http://www.postgresql.org/about/users
- http://www.postgresql.org/about/casestudies/
- http://www.postgresql.org/about/quotesarchive

Wann lohnt sich PostgreSQL nicht?

- Im Embedded Bereich (Embedded Deviced und Einbettung in die Applikation)
- Read-Only Datenbanken
- OLAP und Data Warehouse, Windowing Funktionen (PostgreSQL 8.4)
- Unstrukturierte Daten (ungeeignet für relationale Datenbanken)
- Temporäre Daten

Ende

`http://andreas.scherbaum.la/`

Fragen?

Andreas 'ads' Scherbaum <andreas@scherbaum.biz>
PostgreSQL User Group Germany
European PostgreSQL User Group