

KAPITEL 1: DIE EIGENSCHAFTEN VON POSTGRESQL

1.1 DIE GESCHICHTE VON POSTGRESQL

Postgres ist ein relationales Datenbank-Managementsystem (RDBMS). Eigentlich stimmt diese Aussage nicht so ganz, denn es müßte noch ein Zusatz hinzugefügt werden: ein objektrelationales Datenbanksystem (ORDBMS)¹.

Postgres kann in seinen Anfängen zurückverfolgt werden an die University of California in Berkeley (UCB). Dort wurde zwischen 1977 und 1985 ein relationales Datenbanksystem namens Ingres entwickelt. Ingres wurde recht populär und war bald gut vertreten im universitären Sektor. Um diesen Erfolg auch im kommerziellen Umfeld zu erreichen, wurde der Sourcecode von der Firma Relational Technologies/Ingres Corporation aufgekauft und weiterentwickelt. Diese Firma wurde später von Computer Associates übernommen. Heute gibt es eine Weiterführung dieses ursprünglichen Produkts unter dem Namen CA-INGRES II von Computer Associates. In jüngster Zeit wurde eine Version des CA-Ingres DBMS als OpenSource-Release herausgegeben.

In der Zwischenzeit wurde in den Jahren 1986 bis 1994 ein weiteres Produkt aus den Erfahrungen mit Ingres entwickelt: Postgres (*post* steht für lateinisch *nach*; also nach Ingres). Und wieder wurde dieses Produkt durch eine kommerzielle Firma aufgekauft und vermarktet, nämlich durch Illustra Information Technologies. Der Code von Illustra wiederum wurde von Informix aufgekauft und Teile davon in den Informix Universal Server integriert. Vor ein paar Jahren erwarb IBM Informix.

1995 wurde dem Code von Postgres SQL als Abfragesprache hinzugefügt und der Name änderte sich in Postgres95. Bis zum Jahr 1996 wurde diese Version sehr beliebt und der Name wurde noch einmal geändert, und zwar in den heute üblichen Namen PostgreSQL.

Postgres lag zum Zeitpunkt der Drucklegung des Buchs in der Version 7.4.5 vor, wobei die ersten beiden Stellen der Versionszählung immer die Hauptversionsnummern sind für Versionen, bei denen entscheidende Änderungen vorgenommen wurden, und die dritte Stelle der Versionsbezeichnung die Bedeutung einer kleineren Versionsänderung (vielleicht wegen Sicherheitsproblemen) hat. Die Version 8 des Datenbankservers war zu diesem Zeitpunkt in der Vorbereitungsphase (Beta). Ihre Besonderheiten sind in Anhang erläutert.

¹ Ein objektrelationales Datenbank-Managementsystem ist ein relationales Datenbank-Managementsystem mit objektorientierten Zusätzen wie zum Beispiel Subtabellen, die bestimmte Eigenschaften von den Eltern-Tabellen übernehmen. Dabei werden die ursprünglichen relationalen Eigenschaften wie die Möglichkeit der Datenabfrage durch SQL beibehalten. Diese DBMS fußen also stark auf dem relationalen Paradigma.

Heute wird Postgres von einem immer größer werdenden Team von Programmierern, die sich praktisch überall auf der Welt befinden, weiterentwickelt.

1.2 DIE LIZENZ VON POSTGRESQL

Postgres wird unter einem Open-Source-Lizenzmodell weiterentwickelt, und zwar der Berkeley-Software-Lizenz, die wiederum auf der BSD-Lizenz fußt. Open-Source bedeutet dabei, daß der Sourcecode von Postgres für jedermann frei verfügbar, einsehbar und änderbar ist. Und dabei muß man keine Lizenzgebühren an irgend jemand bezahlen.

Dies ist ein Unterschied zu sogenannten »Shared Source«-Initiativen, die besonders durch Microsoft bekannt wurden. Im Fall der Initiative von Microsoft können besonders Privilegierte (wie beispielsweise Regierungen) den Sourcecode einsehen, diesen aber nicht nach ihrem Gutdünken verändern. Außerdem muß man über das, was man gesehen hat, Stillschweigen bewahren.

Die Lizenz von Postgres erlaubt es dagegen, Änderungen am Code vorzunehmen und diese Änderungen dann selbst zu verkaufen, auch muß der geänderte Code nicht für die Allgemeinheit freigegeben werden. Ein Beispiel, wie dies benützt werden kann, ist die RedHat Database – Postgres mit Zusatztools aufgepeppt und mit Benutzersupport, für den man natürlich dementsprechend bezahlen muß.

1.3 GRUNDLEGENDE KONZEPTE VON POSTGRES

Das Hauptziel der Entwickler von Postgres ist die hundertprozentige Konformität mit dem ANSI-SQL-Standard. Und da ist man schon recht weit gekommen. Wie vorher schon erwähnt wurde, ist es daher relativ leicht, Abfragen, Skripten und ganze Applikationen von anderen Datenbanken nach Postgres zu portieren. Auch kann man seine mühsam erworbenen Kenntnisse von SQL hier gut weiter verwenden.

Postgres besitzt objektorientierte Zusätze. Damit können auch sehr komplexe Sachverhalte gut in den zugrunde liegenden Modellen formuliert werden. Die Hauptmerkmale des objektorientierten Datenmodells von Postgres sind:

- ◆ Klassen
- ◆ Vererbung
- ◆ Überladung von Funktionen

Objektorientierte Zusätze bei Postgres

Das *Klassenkonzept* ist in Postgres dadurch gegeben, daß einzelne Instanzen einer Klasse eine systemweit eindeutige Objekt-ID (object ID oder OID) besitzen und dadurch eindeutig charakterisiert sind. Eine Klasse (die Blaupause für Instanzen) kann in Postgres eine Tabelle darstellen. Eine Instanz (ein reales Objekt) dieser Klasse wäre dann eine einzelne Spalte der Tabelle.

Vererbung wird in Postgres durch die Möglichkeit zur Verfügung gestellt, Kind-Tabellen abzuleiten, wobei die Kind-Tabelle die Eigenschaften der Mutter-Tabelle erbt, wobei die Eigenschaften der Kind-Tabelle in der Muttertabelle nicht sichtbar sind, aber umgekehrt schon. Wenn wir zum Beispiel davon ausgehen, daß wir eine Arbeitertabelle haben mit Vorname, Name, Alter, Gehalt und Filialzugehörigkeit, dann könnten wir davon eine »bayerische Arbeitertabelle« ableiten, die zusätzlich noch ein Feld für die nur in Bayern angebotenen Freizeiteinrichtungen hat. Dort könnten wir vermerken, welche Einrichtungen von dem Mitarbeiter genutzt werden.

Als weiteres Beispiel könnten wir eine Tabelle für Menschen anlegen, wobei Felder für Haare, Hautfarbe und so weiter vorgesehen sind. Davon könnten wir eine Tabelle für Babies ableiten, die ein zusätzliches Feld für die tägliche Trinkmenge an Milch besitzt und ein Feld für die tägliche Gewichtszunahme. Diese »Babytabelle« würde die anderen Felder von der Elterntabelle (»Mensch«) mit übernehmen, wir könnten also dann noch immer feststellen, welche Hautfarbe das Baby hat oder welche Haarfarbe. In diesem Beispiel wüßten aber die »Eltern« (die Tabelle »Mensch«) nichts über die tägliche Gewichtszunahmen oder die tägliche Milchtrinkmenge, da diese Informationen nur weiter unten im Vererbungsbaum sichtbar sind.

Überladung von Funktionen bedeutet, daß eine Funktion mit einem bestimmten Namen mehrfach definiert werden kann. Der Unterschied muß dann in der Anzahl oder/und dem Typ der übergebenen Parameter an diese Funktion liegen, damit zur Laufzeit das System feststellen kann, welche dieser Funktionen verwendet werden soll.

Die Programmiersprache PL/pgSQL bietet diese Möglichkeit an. Sie könnten also dann eine Funktion *subtrahiere* mehrfach definieren (falls Sie etwas hier nicht verstehen, ist dies nicht so wichtig. Später werden wir uns intensiv mit PL/pgSQL beschäftigen):

```
CREATE OR REPLACE FUNCTION subtrahiere(int4 a, int4 b) RETURNS int4 AS '  
DECLARE  
    result int 4;  
BEGIN  
    result := a - b;  
    return result;  
END;  
' LANGUAGE 'plpgsql';
```

```
CREATE OR REPLACE FUNCTION subtrahiere(int4 a, int4 b, int4 c)
    RETURNS int4 AS '
DECLARE
    result int4;
BEGIN
    result := a - b - c;
    return result;
END;
' LANGUAGE 'plpgsql';
```

Zur Laufzeit weiß dann das Datenbanksystem, welche Funktion angewandt werden soll, je nachdem, wie viele Parameter mitgegeben wurden. Nicht vergessen sollte dabei allerdings werden, daß es oft schwer sein wird, bei Fehlern in überladenen Funktionen herauszufinden, welche der Funktionen nun eigentlich betroffen ist und korrigiert werden muß. Dies kann eben nicht über den Namen, sondern nur durch die übergebenen Parameter herausgefunden werden, beispielsweise:

```
SELECT Name, Vorname, subtrahiere(Gesamtgehalt, Monatszahlung_Tennis)
    FROM Bayerische_Mitarbeiter_tabelle;
```

Hier würde natürlich die Funktion mit zwei int4-Parametern, also die erste Definition, aufgerufen werden.

Dies sollte hier kurz als Exkurs über die objektorientierten Möglichkeiten von Postgres dienen. Im weiteren Verlauf werden wir uns nicht näher mit diesen Möglichkeiten beschäftigen. Falls Sie Interesse oder Bedarf an diesen Funktionalitäten haben, muß Ihnen die Standarddokumentation zu Postgres dienen, die Ihnen weiterhelfen wird.

Im folgenden sehen Sie eine Aufzählung der grundlegenden Eigenschaften von Postgres.

1.3.1 Die Eigenschaften von Postgres im Überblick

<i>Beschreibung</i>	<i>Bemerkungen</i>
<p>Postgres läuft auf allen modernen unix-artigen Betriebssystemen wie AIX, Be-OS, BSD/OS (x86 und Sparc), Compaq Tru64 UNIX, Digital UNIX, FreeBSD (x86 und Alpha), HP-UX, Linux (x86, Alpha, ARM, m68k, PowerPC, Sparc, S/390), Mac OSX, NetBSD (x86, Alpha, ARM, m68k, PowerPC, Sparc, VAX), SCO Open Server, SCO UnixWare, SGI IRIX, SunOS 4, Sun Solaris (x86, Sparc).</p> <p>Auf Windows: Windows 2000/NT. Bis zur Version 7.x läuft Postgres in der Posix-Emulation (Cygnum), ab Version 8 steht auch eine native Windows-Version zur Verfügung.</p>	<p>Nicht unterstützte Betriebssysteme sind DG/UX, Nextstep, System V R4 (m88k, MIPS), Ultrix (MIPS, VAX). Diese Betriebssysteme könnten unterstützt werden, es gab aber bisher keine positiven Rückmeldungen.</p> <p>PostgreSQL verwendet Standard-configure- und make-Optionen.</p> <p>Binärpakete sind für die meisten Linux-Distributionen vorhanden.</p> <p>Postgres kann ohne root-Berechtigung installiert und betrieben werden.</p> <p>Postgres ist bei den meisten Linux- und BSD-Distributionen mit enthalten.</p>

Tabelle 1.1: Die Installation und Plattformen

<i>Beschreibung</i>	<i>Bemerkungen</i>
Foreign keys	Die Standard-SQL-Syntax mit <i>CREATE TABLE ... FOREIGN KEY</i> wird unterstützt. Verschiedene Aktionen bei Updates einschließlich <i>CASCADE</i> , <i>RESTRICT</i> oder <i>DEFAULT</i> werden ebenfalls unterstützt.
Joins	Alle SQL99-Join-Arten werden unterstützt.
Views	–
Trigger	Before- und/oder After-Trigger werden unterstützt. Die Trigger-Funktionen können in C oder einer anderen prozeduralen Sprache geschrieben werden.
Unterstützung der meisten SQL99-Datentypen, zusätzliche geometrische Datentypen, Datentypen für Netzwerkadressen, Ethernetkarten-IDs, ISBN/ISSN und so weiter.	Beispielsweise <i>INTEGER</i> , <i>NUMERIC</i> , <i>BOOLEAN</i> , <i>CHAR</i> , <i>VARCHAR</i> , <i>DATE</i> , <i>INTERVAL</i> , <i>TIMESTAMP</i> .

Tabelle 1.2: Die Abfrageoptionen (Teil 1, Fortsetzung auf nächster Seite)

<i>Beschreibung</i>	<i>Bemerkungen</i>
Neue Datentypen können gemeinsam mit den Support-Funktionen und -Operatoren durch den Benutzer definiert werden.	
Unterstützung für BLOBS (Binary Large Objects; beispielsweise Bilder, Videos oder Klänge).	
Support für internationale Zeichensätze.	Kyrillisch und Kanji werden beispielsweise auch unterstützt.
Örtliche Sortierungen und Formatierungen (Währungen, etc.) werden unterstützt.	
Support für die Standard-SQL-Bedingungen wie <i>CASE ... WHEN</i> , <i>COALESCE</i> und <i>NULLIF</i> .	
Unterstützung für Subqueries.	
Unterstützung für <i>SELECT DISTINCT</i> und <i>SELECT DISTINCT ON()</i> .	Damit nur eindeutige Datenzeilen angezeigt werden.
Voller Support für <i>GROUP BY</i> und Aggregat-Funktionen (<i>COUNT</i> , <i>SUM</i> , <i>AVG</i> , <i>MIN</i> , <i>MAX</i> , <i>STDDEV</i> und <i>VARIANCE</i>). Neue Aggregat-Funktionen können durch den Benutzer definiert werden.	
Sub-Selects in <i>FROM</i> .	
Support für <i>UNION</i> , <i>UNION ALL</i> , <i>INTERSECT</i> und <i>EXCEPT</i> .	
Zusätze für <i>LIMIT</i> und <i>OFFSET</i> in Abfragen.	Beispielsweise <i>SELECT * FROM Artikel ORDER BY Kosten LIMIT 5</i> . Dann werden nur die ersten fünf Datensätze angezeigt. Dies ist besonders für Webseiten interessant.
Verschiedene Indexarten.	B-tree, R-tree, Hash und Gist; auch benutzerdefinierte.

Tabelle 1.2: Die Abfrageoptionen (Teil 2, Fortsetzung auf nächster Seite)

<i>Beschreibung</i>	<i>Bemerkungen</i>
Rules	Sind etwas Spezielles für Postgres, die es dem Entwickler erlauben, für <i>SELECT</i> , <i>INSERT</i> , <i>DELETE</i> oder ähnlich irgend eine beliebige Operation durchführen zu lassen (beispielsweise ein Löschkennzeichen setzen und den Datensatz in eine History-Tabelle schreiben).
Temporäre Tabellen, die bei Beendigung der Serververbindung automatisch wieder gelöscht werden.	
<i>LISTEN</i> und <i>NOTIFY</i> zur Benachrichtigung über bestimmte Ereignisse.	
Unterstützung von <i>SCHEMA</i> .	

Tabelle 1.2: Die Abfrageoptionen (Teil 3)

<i>Beschreibung</i>	<i>Bemerkung</i>
ACID-konform	<ul style="list-style-type: none"> • A steht für <i>Atomicity</i>: Eine Transaktion wird als Ganzes betrachtet (Ausführung nur ganz oder gar nicht). • C steht für <i>Consistency</i>: Die Datenbank wird in einem konsistenten Zustand hinterlassen. • I ist <i>Isolation</i>: Eine Transaktion läuft immer isoliert ab und wird durch außen nicht beeinflusst. • D steht für <i>Durability</i>: Sobald eine Transaktion abgeschlossen wurde, dürfen die Änderungen nicht mehr verlorengehen.
Rollback-Unterstützung	Der bisher durchgeführte Teil einer Transaktion kann rückgängig gemacht werden.
Serialisierbare Transaction Isolation	Funktioniert transparent für die Transaktionsfunktionen von Interface-Programmen wie Perl DBI, Zope, JDBC und ODBC.

Tabelle 1.3: Transaktionen

<i>Beschreibung</i>	<i>Bemerkung</i>
Lesende Zugriffe blockieren nie schreibende Zugriffe und umgekehrt. Zusätzlich sind noch verschiedene Row- und Table-Locking-Mechanismen verfügbar.	Das gleiche Verfahren wird auch bei Oracle verwendet, weil es beim alleinigen Lesezugriff sehr rasche Datenzugriffe garantiert.

Tabelle 1.4: Zugriffskontrolle

<i>Beschreibung</i>	<i>Bemerkung</i>
psql (textbasiert)	<i>psql</i> wird im Buch hinreichend verwendet und besprochen.
pgAccess	
phpPgAdmin	webbasiert
Webmin	

Tabelle 1.5: Interfaces

Server-Administration

- ◆ Backup- und Recovery-Werkzeuge (*pg_dump* und *pg_restore*).
- ◆ Benutzer- und Gruppen-Sicherheitskonzept für Datenbankobjekte.
- ◆ Der Zugriff auf den Server kann begrenzt werden auf Basis von Hostnamen, Benutzername oder Datenbankname.
- ◆ Unterstützung der Authentifizierung durch das Kerberos-Protokoll.
- ◆ Verschlüsselte Datenbankverbindungen mittels SSL oder SSH.
- ◆ Postgres kann mehrere CPUs gleichzeitig verwenden.
- ◆ Netzwerkverbindungen entweder über TCP/IP oder über lokale Unix-Sockets.
- ◆ Unterstützung für virtuelle Hosts.
- ◆ Praktisch unbegrenzt große Datenbanken, Tabellen, Datenzeilen. Unbegrenzte Anzahl von Indizes pro Tabelle.

Sprachen und Interfaces

Postgres unterstützt server- und clientseitig eine ganze Reihe Programmiersprachenschnittstellen und Datenbank-Interfaces:

- ◆ C, Embedded C, C++
- ◆ SQL
- ◆ PL/pgSQL
- ◆ Tcl
- ◆ Perl
- ◆ Python
- ◆ Ruby
- ◆ Perl (natives Interface oder mit DBI/DBD)
- ◆ Python (PyGreSql oder PoPy)
- ◆ PHP

- ◆ ODBC (Clients inklusive-MS Access, StarOffice, ApplixWare etc.)
- ◆ JDBC
- ◆ Emacs LISP
- ◆ Scheme (Guile)
- ◆ R
- ◆ C#

1.3.2 Limits

Postgres hat aber auch Einschränkungen, die in der folgenden Tabelle aufgeführt sind:

<i>Funktion</i>	<i>Beschreibung</i>
Maximale Größe einer Datenbank	Unbeschränkt. In der Dokumentation von Postgres ist dazu vermerkt, daß es Datenbanken mit einer Größe bis zu vier Terabyte gibt.
Maximale Größe einer Tabelle	16 TByte bei allen Betriebssystemen ¹ .
Maximale Größe einer Datenzeile	1 GByte.
Maximale Anzahl Datenzeilen in einer Tabelle	Unbegrenzt (beschränkt nur durch den physischen Plattenplatz).
Maximalanzahl von Spalten in einer Tabelle ²	250 bis 1600, abhängig von den Spaltentypen.
Maximale Anzahl von Indizes einer Tabelle	Unbegrenzt.

Tabelle 1.6: Die Einschränkungen von Postgres

Natürlich wird man sich bemühen, seine Tabellen, Indizes, Spalten und so weiter so klein wie möglich zu halten, denn die Geschwindigkeit des Systems kann rasch abnehmen, wenn diese sehr groß sind. Denken Sie einmal an die Geschwindigkeitseinbußen, wenn das System große Datenmengen andauernd auf Platte auslagern und bei Bedarf wieder von der Platte einlesen muß (Swapping).

¹ Bei größeren Tabellen wäre der Kernel eventuell neu zu übersetzen, denn dann würde man einen sogenannten »large file support« benötigen. Die 16 TByte sind die Grenze, die standardmäßig vom Betriebssystem (Linux/Unix) gesetzt wird. Große Tabellen werden als mehrere ein GByte große Dateien im Dateisystem abgespeichert.

² Die maximale Spaltenzahl einer Tabelle und auch die maximale Tabellengröße können geändert (vergrößert) werden, wenn man die Standard-Blockgröße auf 32 Kbyte erhöht.