ralf ADAMS

Mit Beispielen in MySQL/ MariaDB, PostgreSQL und T-SQL

SQL

Der Grundkurs für Ausbildung und Praxis

3. Auflage

Mit Exkurs zu NoSQL

HANSER

- *Sitzungsverwaltung*: Wann immer ein Befehl an den Server gesendet werden soll, muss man sich in einer Sitzung (engl. *session*) befinden. Dazu muss zuerst eine Sitzung geöffnet werden. Jetzt können beliebig viele SQL-Befehle gesendet und Daten empfangen werden. Zum Schluss wird die Sitzung serverseitig – z.B. durch einen Timeout – oder clientseitig beendet.
- *Randbedingungsprüfer*: Für Tabellen können Randbedingungen (engl. *constraints*) formuliert werden, die immer gelten müssen. Würde die Ausführung eines Befehls dazu führen, dass diese Randbedingungen nicht erfüllt sind, wird die Ausführung des Befehls in der Regel verweigert.
- *Datenschutz*: Durch die Vergabe von Zugriffsrechten kann das Recht auf lesende und schreibende Zugriffe so wie auf das Ausführen von Prozeduren passgenau zugeschnitten werden.
- *Datensicherheit*: Der Verlust von Daten ist der GAU[4] schlechthin. Das DBMS muss sicherstellen, dass nicht durch Serverabsturz oder Ähnliches Daten verloren gehen.
- *Transaktionsmanagement*: Transaktionen ermöglichen parallelen Zugriff und eine Art *undo* im Fehlerfall. Das zu gewährleisten, erfordert eine Menge Mühe. Die Qualität des Transaktionsmanagements ist oft ein entscheidendes Merkmal eines DMBS.
- *API*[5]: Die Daten werden in der Regel durch eine oder mehrere Anwendungen (Clients) bearbeitet. Damit die Anwendung auf das DBMS zugreifen kann, braucht es eine Schnittstelle, über die es zu den Daten gelangt. Der MySQL oder MariaDB Server bietet beispielsweise APIs für C, C++, C#/.NET, PHP, Perl, Python und Tcl. Auch stehen APIs für JDBS[6] und ODBC[7] zur Verfügung. Man nennt sie *Konnektoren*.
- *Metadaten*: Verwaltungsinformationen, Statistiken etc., eben der ganze Rest.

Der Begriff *Datenbankmanagementsystem* wird oft anstelle von *Datenbanksystem* verwendet. Gerade die schematischen Darstellungen in den Dokumentationen der Hersteller unterscheiden nicht zwischen diesen beiden Begriffen.

Sind die Datenbanken eines Datenbankmanagementsystems in Form von Tabellen organisiert, so handelt es sich um ein *relationales Datenbankmanagementsystem (RDBMS)*.

Und noch der Vollständigkeit halber:

Definition 3: Datenbanksystem
Ein System, welches die Datenbanken und das dazugehörige Datenbankmanagementsystem als Komplettpaket anbietet, nennt man *Datenbanksystem*.

[4] Abkürzung für: Größter anzunehmender Unfall
[5] Abkürzung für: Application Programming Interface; engl. für Programmierschnittstelle
[6] Abkürzung für: Java Database Connectivity: Programmierschnittstelle für JAVA
[7] Abkürzung für: Open Database Connectivity: Eine offene standardisierte Schnittstelle. Sie wird von fast allen Datenbanksystemherstellern angeboten. Wer ODBC-Programme schreibt, kann leicht zwischen verschiedenen Datenbanksystemherstellern wechseln.

1.2 Im Buch verwendete Server

Kurzporträts der verwendeten SQL Server: Hersteller und Geschichte

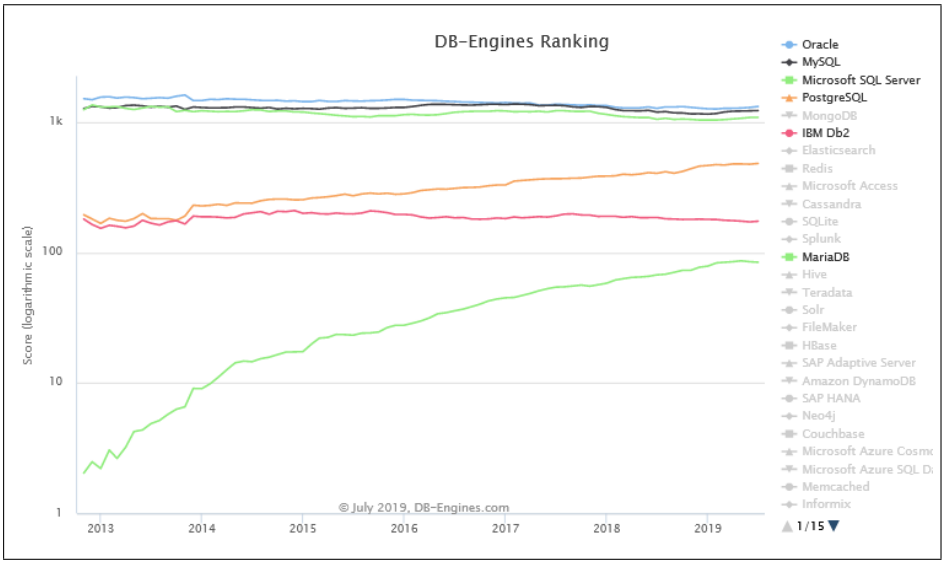

Bild 1.2 Ranking einiger RDBMSe, Quelle [DE19]

In der ersten Auflage dieses Buchs habe ich fast ausschließlich MySQL/MariaDB als Plattform genutzt. Um SQL breiter vorstellen zu können, wurden in der zweiten Auflage die meisten Beispiele auch in PostgreSQL angeboten. Konsequenterweise wird in der dritten Auflage ein weiteres System bedient: der MS SQL Server. Wie Sie Bild 1.2 entnehmen können, decke ich damit eine Vielzahl von Installationen und SQL-Dialekten ab.

1.2.1 MySQL und MariaDB

Die schwedische Firma MySQL AB hat MySQL von 1994 bis 2008 entwickelt. Die ursprüngliche Intention war eine verbesserte und beschleunigte Verarbeitung eines selbst entwickelten Tabellensystems mit dem Namen ISAM. Dazu wurde mSQL[8] genutzt. Von 2008 bis 2010 wurde MySQL AB von Sun Microsystems gepflegt, und seit Januar 2010 wird MySQL unter dem Schirm von Oracle weiterentwickelt.

Der Name *MySQL* kommt nicht vom englischen *my* (mein). Einer der Firmengründer, Michael Widenius, hat sympathischerweise den Vornamen *My* seiner Tochter verwendet. Der Name des Delphins im Logo ist Sakila. Er wurde in einem Wettbewerb ermittelt, den der

8 Kein Tippfehler! Siehe [Ltd10]

Open Source-Entwickler Ambrose Twebaze aus Uganda gewann. Sakila ist ein Mädchenname in der Sprache *siSwati* und auch der Name einer Stadt in Tansania.

Nach der Übernahme von MySQL durch Oracle haben die Spannungen zwischen den Entwicklern und Oracle ständig zugenommen, sodass der *Erfinder* von MySQL – Michael Widenius – sich mit der neu erstellen Engine Aria von MySQL abgespalten und 2009 das Projekt MariaDB ins Leben gerufen hat. Wie MySQL ist auch dieses Projekt nach einer Tochter von Widenius benannt. Wegen seiner offeneren Lizenzpolitik und der schnelleren Umsetzung von Neuerungen und Fehlerkorrekturen hat MariaDB an vielen Stellen – aber nicht, wie oft behauptet, an den meisten – MySQL abgelöst (siehe Bild 1.2 auf der vorherigen Seite).

MySQL und MariaDB sind Client-Server-Datenbanksysteme. Ein Server stellt alleine oder im Verbund mit anderen Servern den Anwendungen (Clients) die Datenbankdienste zur Verfügung. Der MySQL-/MariaDB-Server besteht – wie jedes DBMS – aus vielen Komponenten, die hier kurz angerissen werden.

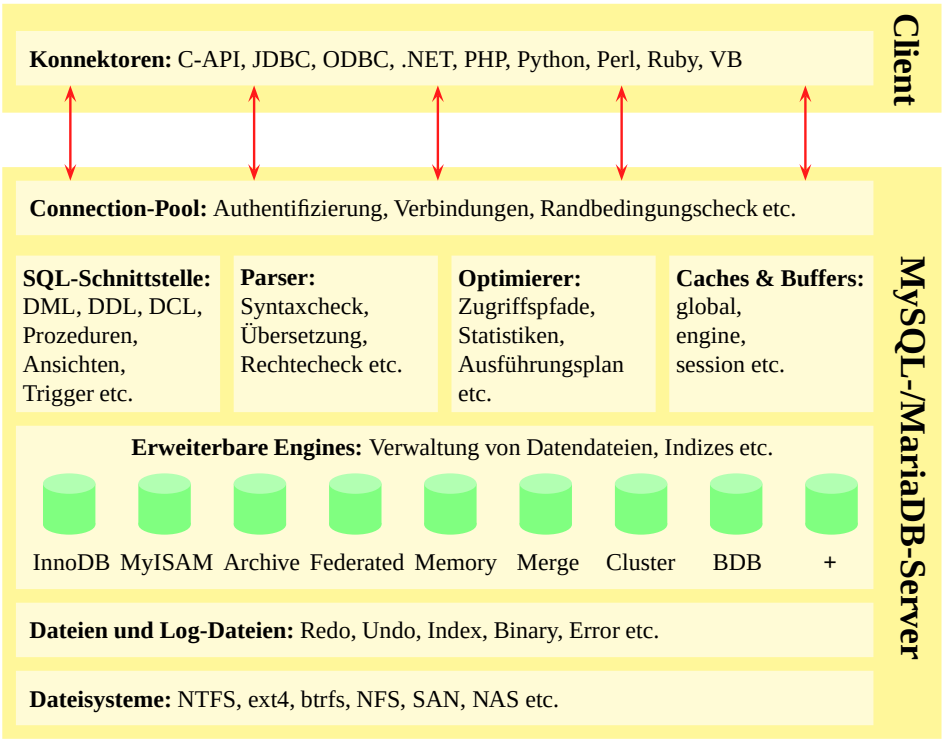

Bild 1.3 Komponenten von MySQL und MariaDB

- *Konnektoren* (Verbinder): Der Client baut über die Konnektoren eine Sitzung zum Server auf. Dies erfolgt über das TCP/IP-Protokoll und in der Regel über den Port 3306.
- *Connection-Pool*: Hier werden die Verbindungen zu den Clients verwaltet. Beim Verbindungsaufbau wird anhand des Benutzernamens und des Passworts die Verbindungsanfrage authentifiziert. Ist die Anzahl der maximal verfügbaren Verbindungen (Connections Limits) nicht überschritten, wird ein Verbindungsthread eingerichtet. Ebenso wer-

den die anderen Grenzwerte für die Verbindung überwacht: Datenübertragungsvolumen, Timeout etc.

- *SQL-Schnittstelle*: Hier werden die SQL-Befehle entgegengenommen. Sie werden dann zum Parser weitergereicht.
- *Parser*: Der Parser überprüft die Syntax eines Befehls und ob man die Ausführungsrechte für diesen Befehl hat.
- *Optimizer* (Optimierer): Anhand von Schätzungen, Statistiken und Algorithmen wird für nicht triviale Befehle ein Ausführungsplan erstellt. Dieser Ausführungsplan berücksichtigt ggf. im Cache vorhandene Ergebnisse.
- *Caches und Buffers* (Zwischenspeicher): Daten und Ergebnisse können in Zwischenspeichern aufgehoben werden. Diese sind nur in der Sitzung verfügbar, die diese erstellt hat (lokal) oder in allen Sitzungen (global).
- *Storage Engines*: Die Motoren des Servers. Hier werden die Daten tatsächlich verarbeitet. Jede Engine ist dabei für bestimmte Aufgabenstellungen besonders gut geeignet. Der große Vorteil von MySQL und MariaDB ist, dass jeder mit einem besonderen Bedarf eine Engine bauen kann. Er muss *nur* die Schnittstellen beachten (siehe [MyS19b]) und kann dann seine Speziallösungen anbieten.
- *File System* (Dateisystem): Je nach Betriebssystem werden hier Daten in unterschiedlichen Dateisystemen (NTFS, BTRFS, ETX4 etc.) abgelegt. Bis auf die Frage, ob das Dateisystem zwischen Groß- und Kleinschreibung unterscheidet, spielt dieses für die SQL-Programmierung keine Rolle.
- *Management Services & Utilities*: Parallel dazu gibt es die vielen kleinen Helferlein, ohne die nichts geht: für das Sichern und Wiederherstellen von Daten, Datenreplikation, Administration und Konfiguration, Datenmigration und Metadaten.

In den nachfolgenden Kapiteln werden wir gemeinsam eine Datenbank planen, installieren, einrichten, verändern und verwenden. Dabei werden die vielen Fragen beantwortet, die Sie nach dieser kurzen Einführung mit Sicherheit haben werden, also nur Geduld ...

1.2.2 PostgreSQL

Geboren wurde PostgreSQL 1986 als Universitätsprojekt an der Universität von Kalifornien, Berkeley. Professor Michael Stonebraker[9] begann das Projekt als Nachfolgeprojekt der Ingres-Datenbank, welche ebenfalls noch heute verwendet wird. Daher leitet sich auch der Name her: *post ingres*. In den nächsten acht Jahren wurde Postgres von Prof. Stonebraker und seinen Studenten immer weiter entwickelt.

Bis 1995 verstand Postgres allerdings kein SQL, sondern nur eine eigene Sprache namens POSTQUEL. Die beiden Doktoranden Andrew Yu und Jolly Chen haben Postgres um SQL erweitert und als Postgres95 veröffentlicht.

1996 wurde Postgres95 als Open-Source-Projekt der Netzgemeinde zur Verfügung gestellt und wird seitdem sehr stark von Unterstützern weiterentwickelt und gefördert. Ebenso

[9] Prof. Stonebraker ging später mit einem Fork von Postgres – Illustra – zu Informix, welche diesen in den Universal Server integrierte. Auch diese Datenbank besteht heute noch.

wurde das Erscheinungsjahr aus dem Produktnamen entfernt und die Datenbank heißt seitdem PostgreSQL.

Weitere Informationen über die Geschichte von PostgreSQL und die aktuellen Features des Servers können Sie auf der Homepage des Projekts (*https://www.postgresql.org*) nachlesen.

Von *http://www.postgresql.org/download/* können Sie die aktuelle PostgreSQL-Version für Linux-Distributionen und Windows herunterladen. Eine sehr ausführliche Online-Dokumentation steht unter *http://www.postgresql.org/docs/* zur Verfügung. Dort wird auch auf eine Reihe von weiteren Dokumentationen wie beispielsweise zum Thema *Sicherheit* verwiesen.

1.2.3 Microsoft SQL Server

Anders als bei den anderen Datenbankservern kann ich hier keine Geschichten über Töchter oder Universitätslaufbahnen erzählen. Die Entwicklung des MS SQL Servers ist da schon etwas nüchterner ([Gar16]).

In Kooperation mit Sybase brachte Microsoft 1989 die erste Version seines Servers auf den Markt. Zielplattform war das Betriebssystem OS/2. Die Kooperation sah so aus, dass beide gemeinsam am Sybase Server arbeiteten und Sybase das Produkt unter seinem Namen auf UNIX-basierenden Systemen anbot und Microsoft auf OS/2.

Spätestens seit 1992 speist sich der Quelltext beider Server-Produkte (Sybase 4.0 und MS SQL Server 4.2) gleichwertig aus beiden Firmen. Ab 1993 ist nicht mehr OS/2, sondern Windows NT die Zielplattform des MS SQL Servers. Mit der Portierung auf Windows NT wurden erhebliche Anpassungen im Quelltext notwendig, die nicht mehr für das Sybase-Produkt verwendet werden konnten; die beiden Systeme begannen sich voneinander zu entfernen. Besonders der Wunsch von Sybase, dass der Quelltext möglichst plattformneutral bleiben sollte, widersprach den strategischen Zielen Microsofts, die eine volle Unterstützung der damals wirklich bedeutsamen Neuerungen von Windows NT anstrebten.

Microsoft entschied, dass der SQL Server ein zentrales Produkt für die Windows NT Strategie werden sollte. Daher wurden von anderen Datenbankherstellern erfahrene Entwickler angeworben. Dieses geballte Know-how mündete 1998 in die Serverversion 7.0 mit dem Arbeitstitel *Sphinx*. Mit dieser völlig überarbeiteten Version wurde auch die Zusammenarbeit mit Sybase überflüssig und beide Produkte sind seitdem voneinander unabhängig.

Der MS SQL Server wurde in der Folge immer weiter ausgebaut: Es kamen Sprachelemente hinzu, die Sicherheit wurde verbessert, die Performance gesteigert, die Stabilität erhöht, andere Betriebssysteme werden unterstützt usw. Besonders das grafische Frontend und die hohe Integration in die Visual Studio Entwicklungsumgebung werden von vielen Entwicklern geschätzt.

Anders als MySQL/MariaDB oder PostgreSQL kommt der MS SQL Server aber nicht aus der Open-Source- oder Uni-Ecke. Er ist ein rein kommerzielles Produkt und hat daher in den entsprechenden Kreisen ein Imageproblem.