R.I.P. Oracle-Datenbank

Markus Lohn
esentri AG
Ettlingen

Schlüsselworte
Datenbanken, NoSQL, Open Source, Cloud

Abstrakt
Relationale Datenbanken implementieren ein auf der Mengenlehre basiertes Modell. Dieses Modell ist noch dazu für fast für Jedermann leicht zu verstehen und anwendbar. Relationale Datenbanken, insbesondere Oracle, galten bisher als der Defacto-Standard, wenn es um das Speichern von Informationen ging. Mittlerweile hat die Oracle-Datenbank mit über 30 Jahren ein stolzes Alter erreicht. In der IT sind diese Zeiträume im Vergleich zu den üblichen Innovationszyklen sehr lang. Deshalb kann man sich schon die Frage stellen, ob die Oracle-Datenbank so langsam Reif für die Rente ist. Insbesondere die heutigen Anforderungen hinsichtlich Performance, Skalierbarkeit, Vielfalt und Menge der zu speichernden Daten stellen hohe Anforderungen an die Datenbank im Allgemeinen. Auch aus dem Kreise der Entwickler wächst der Druck hinsichtlich CI/CD-Unterstützung bei der Datenbankentwicklung. Sind Prinzipien wie ACID, Lesekonsistenz, Transaktionssicherheit etc. heute noch zeitgemäß und notwendig? Im Bereich der NoSQL-Datenbank gibt es in der Zwischenzeit viele Alternativen mit Potential. In diesem Vortrag werden die verschiedenen Aspekte kritisch beleuchtet und es wird der Versuch unternommen, einen Ausblick auf die Zukunft zu geben.

Einleitung

Die relationale Datenbank ist auch heute noch ein wichtiger Baustein in der IT zum Speichern von vorwiegend strukturierten Daten. Bei Oracle ist die Datenbank trotz der Ausrichtung auf Cloud und vielfach durchgeführten Merger und Erweiterung des Produktportfolios immer noch ein wichtiger Baustein für den Unternehmenserfolg. Datenbanken werden in jedem IT-Projekt benötigt. Viele Jahre wurde ausschließlich auf relationale Datenbanken vertraut, jedoch hat sich das Blatt in den vergangenen Jahren gewendet. Eine Vielzahl von alternativen Lösungen sind entstanden. Sowohl nicht-relationale als auch relationale Datenbanken aus dem Open Source Bereich wurden immer häufiger dem großen, teuren und komplexen Oracle vorgezogen.
Firmen wie Amazon, Facebook und Co. sehen sich mit Anforderungen konfrontiert, die beispielsweise mit Oracle unter technischen Aspekten und Kostengründen nur schwer umsetzbar sind.

Gehen relationale Datenbanken nun in Rente oder ist das wieder nur eine kleine Delle in der Erfolgsgeschichte? Bereits Ende der 90er Jahre wurde ja bereits das Ende der relationalen Datenbank verkündet, durch das Aufkommen von objekt-orientierten Datenbanken. Leider konnten sich diese Lösungen nicht durchsetzen und fristeten ein Nischendasein. Weitere Beispiele sind XML-Datenbanken mit ähnlichem Ausgang.

Historie

Anhand der Historie lässt sich gut eine Evolution bei den Datenbanktechnologien erkennen, immer unter dem Aspekt der technischen Möglichkeiten und Anforderungen. Grundsätzlich kann man die Zeit seit dem 2. Weltkrieg bis heute in drei Phasen einteilen:

Pre-Relationale Phase

Informationen sammeln, speichern und auffindbar zu machen war schon immer die Hauptaufgabe für Datenbanken. Frühe Formen einer Datenbank in physischer Form waren Bücher mit Index, wie Lexika oder Enzyklopädien. Nach dem 2. Weltkrieg wurde die Technik aber immer besser und ausgefeilter.

Zunächst wurden Daten noch auf magnetischen Bändern gespeichert und ermöglichten einen sequentiellen Zugriff. Mit Einführung von Magnetic Discs in den 50er Jahren erlaubte die Technik dann einen direkten Zugriff auf die gespeicherten Daten (ISAM=Index Sequentiell Access Method). Trotz des schnellen Zugriffs bestanden aber immer noch gravierende Nachteile. Für den Zugriff auf die Datenbank musste immer eine eigene Applikation entwickelt werden. Aufgaben wie Konsistenz, Transaktionsmanagement, Caching und Sperren mussten immer neu entwickelt werden. Die immer gleichen Aufgaben wurden schließlich in eine Abstraktionsschicht ausgelagert, die dann als DBMS bezeichnet wurde.

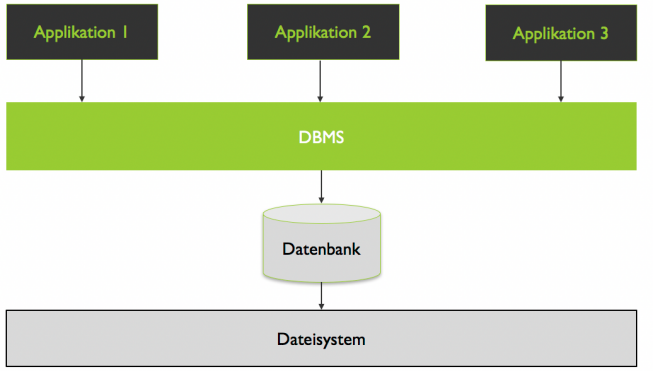

Abbildung 1 Datenbankmanagement System

Das DBMS stellt die notwendigen Funktionen und Mechanismen für die Verwaltung von Daten in der Datenbank zur Verfügung. Ferner ist das DBMS wiederverwendbar. Diese Ideen führten dann Anfang der 70er Jahre zu neuen Datenbanksystemen. Zwei Modelle dominierten hierbei den Markt bis Ende der 70er Jahre:

- Netzwerkdatenbanken
- Hierarchische Datenbanken

Allerdings hatten auch diese Datenbanksysteme einige Nachteile, vor allem in Bezug auf Analyse und Reporting von Daten. Begründet war das vor allem mit der Zusammenlegung des logischen und physischen Datenmodells. Diese Datenbanken konnten nur von Spezialisten bedient werden und eine einheitliche Abfragesprache existierte auch nicht.

Relationale Phase

Mit der Motivation ein DBMS zu entwickeln das einem formalen Modell als Grundlage folgt, logisches und physisches Datenmodell trennt und noch dazu von jedermann nutzbar sein sollte, entwickelte Edgar Codd seine Idee von einem neuen Datenbanksystem. Anfang der 70er entstand ein Konzeptpapier das die Grundideen für die heute bekannten relationalen Datenbanksysteme beschreibt. Die relationale Theorie beschreibt folgend Grundbegriffe:

- Relationen
- Tupel
- Schlüssel

Datenbanken bieten ein Transaktionskonzept. Obwohl nicht darauf beschränkt, ist das Transaktionsmodell ACID eng mit dem relationalen Modell verbunden. Jedes RDBMS implementiert das Transaktionskonzept unter Einhaltung der ACID-Prinzipien. Es entstanden in dieser Zeit mehrere Implementierungen, u. a. Ingres an der Berkeley Universität oder System R bei IBM. Kommerziellen Erfolg und damit auch der Durchbruch von relationalen Datenbanken schaffte die Firma Oracle. Oracle konnte auf Workstations betrieben werden und somit war die Abhängigkeit zum Mainframe gebrochen. Überdies war die Nutzung und Unterstützung der Abfragesprache SQL ein weiterer Erfolgsgarant. Die

relationalen Datenbanken wurden im Laufe der Zeit immer wieder den technischen Anforderungen angepasst und stellten quasi einen de-facto Standard dar. Auch das Aufkommen der Objektorientierung und OO-Datenbanken änderte das nicht.

Post-Relationale Phase
Mit der Einführung von Smartphones und Social Networks änderte sich das Nutzerverhalten und damit auch die Anforderungen an die IT-Infrastruktur, insbesondere Datenbanken. Firmen wie Amazon, Uber etc. orientieren ihr Geschäftsmodell ausschließlich auf das Internet. Google, Amazon, Facebook und Co. sehen sich extremen Herausforderungen hinsichtlich Verfügbarkeit, Performance, Skalierung und Datenmengen gegenüber. Relationale Datenbanken kommen bei diesen Anforderungen an die Grenzen. Auch wenn mit Oracle RAC beispielsweise eine Skalierung einer Oracle DB aus technischen Gesichtspunkten möglich wäre, so ist es jedoch unter Betrachtung der Kosten nicht tragfähig. Deshalb entstanden u.a. neuen Datenbank Architekturen mit Einsatz von Open Source Datenbanken, Caching-Technologien und Frameworks (do-it-yourself). Aber auch völlig neue Ansätze wurden entwickelt und führten zu neuen Datenbanksystemen, z. B. Amazon Dynamo. Da diese Ansätze kein relationales Modell verfolgten, wurden diese auch als NoSQL-Systeme bezeichnet. Vielfältige Lösungsalternativen für verschiedene Anwendungszwecke stehen somit zur Verfügung.

Datenbanken haben im Laufe der Zeit eine Evolution erlebt. Relationale Datenbank stellen nicht das Ende der Evolution dar. Neue Herausforderungen werden mit neuen Ansätzen begegnet. Somit ist der aktuelle Trend auf jeden Fall nachhaltig und nicht von kurzer Dauer.

Anwendungsarchitektur / Web 2.0
Benutzer erwarten einen schnellen und einfachen Zugriff auf Applikationen und Dienste mit unterschiedlichen Geräten. Der Wechsel zu einem alternativen Dienst ist für Benutzer leicht umzusetzen, wenn die "Userexperience" nicht stimmt. Der Verlust von Kunden kann zu einem ernsten Problem für die Anbieter werden. Dies bedingt aber, dass Applikationen immer höhere Anforderungen hinsichtlich Funktionalität, Performance und Skalierung erfüllen müssen. Vor der Dotcom-Krise konnte die Skalierung auf Ebene der Datenbank durch Hinzunahme eines leistungsfähigeren Servers erreicht werden. Das funktioniert jetzt so nicht mehr, vor allem wenn die Kosten betrachtet werden. Bei Nutzung relationaler Datenbanken ist vor allem der Single Point of Failure und der Skalierungsfaktor ein Thema. Single Point of Failure versucht man durch Replikation der Daten zu lösen. Um die Datenbank hinsichtlich Schreib- und Lesevorgänge zu entlasten werden Cachetechnologien eingeführt, die vor die Datenbank geschaltet werden. Eine weitere Maßnahme um die Skalierung von Datenbanken zu verbessern ist „Sharding“. Mit Sharding werden die Daten nach einem Schlüssel auf verschiedene Nodes verteilt. Die Daten liegen also nicht mehr wie üblich an einer zentralen Stelle. Sharding im Zusammenspiel mit Caching und Replikation sind die einzigen Möglichkeiten um ein relationales Datenbanksystem zu skalieren. Jedoch sind damit auch Nachteile verbunden:

- Die Komplexität in der Bereitstellung und Verwaltung steigt enorm an.
- Das ACID-Prinzip funktioniert bei Sharding nicht mehr in der gewohnten Art und Weise.
- JOINS müssen in der Anwendung programmiert werden.

Aus diesem Grund wurden Überlegungen angestellt auf alle oder bestimmte Eigenschaften, z. B. Konsistenz, zu verzichten. Hierdurch entstanden interessanten Neuentwicklungen die mittlerweile unter dem Begriff NoSQL zusammengefasst wurden. Als theoretische Grundlage beschreibt das CAP-Theorem folgenden Umstand: „In verteilten Systemen ist es unmöglich, gleichzeitig die drei Eigenschaften Konsistenz, Verfügbarkeit und Partitionstoleranz zu garantieren.“ Folgende Kategorien von neuen Datenbanksystemen (NoSQL) existieren:

- Key-Value Store
- Document Databases
- Graph Databases

Große Websites mit vielen Benutzern wie Amazon, Google oder Facebook benutzen keine Oracle Datenbank. Oracle Software ist zu komplex und nicht preiswert, um die hohen Anforderungen dieser Applikationen gerecht zu werden. Ferner verfügt Oracle über keine hohe Reputation in der Entwicklergemeinde. Als Beispiel dient die aktuelle Debatte zwischen der Java Community und Oracle zum Thema Java EE. Darüber hinaus existieren nun viele Alternativen, die es ermöglichen eine optimierte Datenbankarchitektur für die jeweilige Applikation zu entwerfen.

Big Data
Die Bedeutung von IT in Unternehmen hat in den letzten Jahren stark zugenommen. IT ist mittlerweile die Basis für die Umsetzung von neuen Geschäftsmodellen und somit ein maßgeblicher Faktor für Unternehmenserfolg. Grundlage hierfür ist das Sammeln und Auswerten von Daten. Es gibt viele Möglichkeiten Daten zu sammeln, z. B. durch soziale Netzwerke, Smartphones, Wearables und vor allem Sensoren. Mit Sensoren kann jede Maschine oder andere Geräte in die Lage versetzt werden Daten zu ermitteln und bereitzustellen. Die Daten sind umfangreich, schnelllebig und schwach strukturiert. Um diese Datenflut bewältigen und zu nutzen sind klassische IT-Architekturen nicht optimal geeignet. Vor allem die Datenbankarchitektur ist besonders herausfordernd. Relationale Systeme sind aufgrund der schwach strukturierten Daten und Probleme beim Skalieren wenig sinnvoll. Vor allem Google als Betreiber einer der größten Website hat sich hier als Innovator hervorgetan. Aufgrund der Notwendigkeit nach neuen Wegen und Lösungen zu suchen hat Google eine neue Architektur für Hardware und Software etabliert.

Die entworfene Architektur beinhaltet mehrere Frameworks die häufig unter dem Namen Hadoop zusammengefasst wird. Auch im Serviceangebot für Big Data Lösungen von Oracle findet sich keine relationale Datenbank. Somit ist auch bei Oracle ersichtlich, dass die Datenbankarchitektur in diesem Kontext nicht alleine mit der relationalen Datenbank bewältigt werden kann.

Cloud
Oracle konzentriert sich seit einigen Jahren auf die Entwicklung und Bereitstellung von Cloud Services. Die Firma ist in der Zwischenzeit in allen wichtigen Segmenten vertreten:
- Infrastructure as a Service (IaaS)
- Platform as a Service (Paas)
- Software as a Service (Saas)

Welchen Stellenwert genießt die Oracle-Datenbank im Vergleich zu den Cloud-Angeboten. Fakt ist auch, dass Innovationen zunächst im Cloud-Service umgesetzt werden. Die Verfügbarkeit einer OnPremise Variante mit den gleichen Funktionen hat nicht mehr Vorrang. Beispiel ist die Veröffentlichung der Oracle Datenbank 12c Release 2 in der Cloud. Der Termin für die OnPremise-Version steht noch nicht fest. Wesentlicher Vorteil cloud-basierter Services ist die flexible Anpassung an sich ändernde Workloads. Dem Kunden verschafft das Flexibilität, allerdings haben relationalen Datenbanken Schwierigkeiten sich flexibel an veränderte Workloads anzupassen (Problem mit der Skalierung). In erwarteten Version 12c Release 2 gibt es mit Sharding, In-Memory DB und RAC Technologien, die eine Anpassung an Workloads ermöglichen. Hier stellt sich die Frage nach den Kosten und Restriktionen des entsprechenden Cloud-Services.
Im Bereich Big Data bietet Oracle bereits eine spezialisierte Lösung ohne relationale Datenbank an. Das Thema NoSQL wird ebenfalls berücksichtigt. Die relationale Datenbank ist nicht mehr das alleinige

Rückgrat von Oracle. Je nach Anforderung und Anwendungsgebiet werden unterschiedliche Architekturen genutzt. Hier stellt sich natürlich die Frage wie viele Innovationen in diese Technologie noch investiert werden. Aufgrund des Servicegedankens ist es letztendlich auch egal wo die Daten gespeichert werden. Entscheidend ist die Verfügbarkeit eines Service der die Daten zur Verfügung stellt. Das Thema relationale Datenbank wird zukünftig wohl an Bedeutung verlieren.

Open Source
Durch die aggressive Expansionsstrategie und strikte Fokussierung auf den Geschäftserfolg hat Oracle das Ansehen in der Entwicklergemeinde verloren. Gerne sucht man nach alternativen Möglichkeiten für die Datenbankarchitektur um Projektanforderungen zu erfüllen. Zusätzlich hat die Lizenzpolitik der letzten Jahre, insbesondere das Thema VMware, dazu beigetragen, die Beliebtheit von Open Source Datenbanken zu beschleunigen. Viele Firmen versuchen durch den Einsatz von Open Source Datenbanken die Abhängigkeit zum Hersteller Oracle zu verringern und Kosten (Lizenzgebühren) einzusparen. Vor allem im Bereich der Applikationsentwicklung im Internet werden häufig Open Source Datenbanken, egal ob relational oder NoSQL eingesetzt. Es ist zusätzlich zu beobachten, das viele Innovationen durch die Open Source Bewegung vorangetrieben werden. Das oberste Ziel ist die Bereitstellung einer technisch guten Lösung, Gewinnmaximierung steht nicht im Vordergrund. Ferner spielt das Thema Transparenz eine wichtige Rolle. In Open Source Projekten kann die Arbeitsweise des Produktes nachvollzogen werden, der Source Code ist frei zugänglich. Bei einer Closed-Source Lösung ist das nicht so ohne weiteres möglich. Der Kontakt mit dem Hersteller wird grundsätzlich über eine Supportorganisation hergestellt. Zugang zum Source Code erhält man normalerweise nicht. Die Nachvollziehbarkeit und Transparenz wird oft beim Hersteller vermisst.

Fazit
Oracle hat es geschafft immer wieder Innovationen in das relationale Datenbanksystem einzuarbeiten und Synergien zu erzielen, z. B. Object Option, XML DB, JSON-Speicherung, Oracle RAC, Sharding etc. Die Datenbank wird so zu einem Alleskönner mit relationaler Basis. Ich glaube nicht das der Ansatz für die Zukunft vielversprechend ist. Je mehr Pakete hinzugefügt werden, um so umfangreicher wird das Produkt und die Administration wird dadurch nicht einfacher. Ferner müssen auch die Kosten eine Rolle spielen. Üblicherweise verlangt Oracle für Erweiterungen zusätzliche Lizenzgebühren. Ich denke die relationale Datenbank wird immer noch benötigt werden, vor allem in Bereichen die mit stark strukturierten Daten arbeiten und das ACID-Prinzip zwingend erforderlich ist. Zunehmend gibt es aber Anwendungsfälle die mit Alternativen besser gelöst werden können. Die alternativen Datenbankarchitekturen (NoSQL) werden auch in der Zukunft Bestand haben und für die entsprechenden Anwendungsfälle bevorzugt eingesetzt werden. Mit zunehmenden Kostendruck werden auch relationale Open Source Datenbanken immer mehr Marktanteile von Oracle „rauben“. Somit steht Oracle einer spannenden Zukunft und Herausforderungen gegenüber.

Kontaktadresse:
Markus Lohn
esentri AG
Pforzheimer Str. 132
D-76275 Ettlingen

Telefon: +49 (0) 7243 354900
Fax: +49 (0) 7243 3549099
E-Mail markus.lohn@esentri.com
Internet: www.esentri.com