Einführung in Datenbanken

Kapitel 4: SQL: Geschichte und Standards

Prof. Dr. Stefan Brass

Martin-Luther-Universität Halle-Wittenberg

Wintersemester 2020/21

http://www.informatik.uni-halle.de/~brass/db20/

Lernziele

Nach diesem Kapitel sollten Sie Folgendes können:

- Das Jahrzehnt nennen, in dem das relationale Modell und die erste Version von SQL entwickelt wurden.
- Den Namen des Forschers nennen, der das relationale Modell erfunden hat.
- Das Jahr nennen, in dem der erste SQL-Standard erschienen ist, sowie ein paar Worte zur weiteren Entwicklung sagen.

  Z.B. das Jahr des aktuellen Standards ungefähr benennen, oder auch, wie häufig ungefähr neue Standards erscheinen.
- Sich eine Version des Standards besorgen.

  Kostenpflichtig oder inoffizielle Vorversionen im Internet.
  Zusätzlich auch sich eine SQL-Grammatik im Internet besorgen.

Inhalt

1. Geschichte des Relationalen Modells
2. SQL: Geschichte
3. SQL Standards
4. SQL in der Vorlesung
5. DBMS-Geschichte

Geschichte des relationalen Modells

- Edgar F. Codd (1923–2003) gilt als Vater des relationalen Modells.

  Er arbeitete damals im IBM Forschungszentrum in San Jose (Kalifornien).

  [https://en.wikipedia.org/wiki/Edgar_F._Codd]

- Seine Publikation „A Relational Model of Data for Large Shared Data Banks“ erschien 1970.

  In der Fachzeitschrift „Communications of the ACM“, Vol. 13, No. 6, Juni 1970, Seiten 377–387.

  [https://dl.acm.org/doi/10.1145/362384.362685]

  [https://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf]

- 1981 bekam er den Turing Award.

  Ein Art Nobelpreis für Informatik.

  [https://dl.acm.org/doi/pdf/10.1145/358396.358400]

Inhalt

1. Geschichte des Relationalen Modells
2. SQL: Geschichte
3. SQL Standards
4. SQL in der Vorlesung
5. DBMS-Geschichte

SQL: Bedeutung

- Heute ist SQL die einzige DB-Sprache für relationale DBMS (Industriestandard). (NoSQL-DBen sind nicht relational.)

  Andere Sprachen wie QUEL sind ausgestorben. Die Sprache QBE hat zumindest einige graphische Schnittstellen zu Datenbanken inspiriert (z.B. in Access). Die Sprache Datalog hat SQL:1999 beeinflusst und wird noch in der Forschung studiert und weiterentwickelt, und hat eventuell eine Zukunft als Datenbank-Programmiersprache. Jedes kommerzielle RDBMS muss heute eine SQL-Schnittstelle haben. Es kann zusätzlich noch andere (z.B. graphische) Schnittstellen geben. Nicht alle Datenbanken sind relational und haben dann auch andere Anfragesprachen bzw. Programmier-Schnittstellen.

- SQL wird verwendet für:
  - Interaktive „Ad-hoc“-Befehle und
  - Anwendungsprogrammentwicklung

    Es gibt für fast alle Programmiersprachen (z.B. Java, PHP) Schnittstellen, die SQL anbieten (geschrieben als Strings oder „eingebettet“).

SQL: Geschichte

- SEQUEL, eine frühere Version von SQL, wurde von Chamberlin, Boyce et al. bei IBM Research, San Jose (1974) entwickelt.

  SEQUEL steht für „Structured English Query Language“ („Anfragesprache in strukturiertem Englisch“). Manche Leute Sprechen SQL noch heute auf diese Art aus. Der Name wurde aus rechtlichen Gründen geändert (SEQUEL war bereits eine eingetragene Marke). Codd war auch in San Jose, als er das relationale Modell erfand.

- SQL war die Sprache des System/R (1976/77).

  System/R war ein sehr einflussreicher Forschungs-Prototyp.

- Die ersten SQL-unterstützenden kommerziellen Systeme waren Oracle (1979) und IBM SQL/DS (1981).

Inhalt

1. Geschichte des Relationalen Modells
2. SQL: Geschichte
3. SQL Standards
4. SQL in der Vorlesung
5. DBMS-Geschichte

SQL: Standards (1)

- Erster Standard 1986/87 (ANSI/ISO).

  Das war sehr spät, weil es schon viele SQL-Systeme auf dem Markt gab. Der Standard war der „kleinste gemeinsame Nenner“. Er enthielt im wesentlichen nur die Schnittmenge der SQL-Implementierungen.

- Erweiterung für Fremdschüssel usw. ’89 (SQL-89).

  Diese Version wird auch SQL-1 genannt. Alle kommerziellen Implementierungen unterstützen heute diesen Standard, aber jede hat viele Erweiterungen. Der Standard hatte 120 Seiten.

- Größere Erweiterung: SQL2 oder SQL-92 (1992).

  626 Seiten, aufwärts kompatibel zu SQL-1. Der Standard definiert drei Kompatibilitätsgrade: „Entry“, „Intermediate“, „Full“ (später wurde „Transitional“ zwischen ersten beiden ergänzt). Die meisten Systeme (z.B. Oracle 8.0 und SQL Server 7.0) waren nur „Entry Level“-kompatibel, und hatten nur einen Teil der höheren Konstrukte.

SQL: Standards (2)

- Weitere große Erweiterung: SQL:1999.

  Anstelle von Levels definiert der Standard nun „Core SQL“ (im wesentlichen der Entry Level von SQL-92) und eine Vielzahl von Paketen von Sprach-Features.

- Wichtige neue Möglichkeiten in SQL:1999:

  - Nutzer-definierte Datentypen, strukturierte Typen.

    Man kann nun „distinct types“ definieren, die Domains sehr ähnlich sind, aber ein Vergleich zwischen verschiedenen „distinct types“ ist nun ein Fehler. Es gibt auch Typ-Konstruktoren „`ARRAY`“ und „`ROW`“ für strukturierte Attributwerte und „`REF`“ für Zeiger auf Zeilen.

  - OO-Fähigkeiten, z.B. Vererbung/Untertabellen.
  - Rekursive Anfragen.
  - Trigger, persistent gespeicherte Module.

SQL: Standards (3)

- Erweiterungen 2003 u.a.
  - XML-Unterstützung

    Man kann jetzt eine XML-Datei (und damit auch XHTML) als einen Tabelleneintrag in der Datenbank abspeichern. Es gibt Funktionen zum Zugriff auf die Baumstruktur und zur Generierung von XML aus relationalen Daten.
  - OLAP-Unterstützung (Online Analytical Processing) u.a. mit „Window Functions“.

    Dies ist ein Kapitel der Fortsetzungs-Vorlesung „Datenbank-Programmierung“.
  - Sequenzen (Generatoren für eindeutige Nummern)
  - „`MULTISET`“ type constructor.
- 2006: Überarbeitung des XML-Teils.

SQL: Standards (4)

- Erweiterungen 2008 u.a.:
  - `TRUNCATE`
  - `ORDER BY` auch in Unteranfragen (für `FETCH FIRST`).
  - `INSTEAD OF` Trigger.
- Erweiterungen 2011 u.a.:
  - Zeitbezogene Daten (PERIOD FOR)
  - Erweiterungen der Window Functions.
- Erweiterungen 2016 u.a.:
  - JSON-Unterstützung
  - Row Pattern Matching.

SQL: Standards (5)

- Aktuelle Version des Standards ist SQL:2016.

ISO/IEC 9075-1:2016	Framework (SQL/Framework)
ISO/IEC 9075-2:2016	Foundation (SQL/Foundation)
ISO/IEC 9075-3:2016	Call-Level Interface (SQL/CLI)
ISO/IEC 9075-4:2016	Persistent Stored Modules (SQL/PSM)
ISO/IEC 9075-9:2016	Management of External Data (SQL/MED)
ISO/IEC 9075-10:2016	Object Language Bindings (SQL/OLB)
ISO/IEC 9075-11:2016	Information and Definition Schemas (SQL/Schemata)
ISO/IEC 9075-13:2016	SQL Routines and Types Using the Java TM Programming Language (SQL/JRT)
ISO/IEC 9075-14:2016	XML-Related Specifications (SQL/XML)

Teil 1, 2, 11: minimale Anforderungen, andere Teile: Erweiterungen.

Die Lücken in den Nummern der Bände sind nur historisch zu verstehen.

Es gibt einen verwandten Standard ISO/IEC 13249:

„SQL multimedia and application packages“.

- Teil 2 definiert den Kern der Sprache und hat 1707 Seiten.

SQL: Standards (6)

- Der offizielle SQL Standard ist nicht kostenlos.

  [https://webstore.ansi.org/industry/software/software-engineering/sql]

  [https://www.iso.org/standard/63556.html]

  Man kann 24 Seiten „Preview“ mit dem Inhaltsverzeichnis kostenlos einsehen:

  [https://webstore.ansi.org/preview-pages/ISO/preview_ISO+IEC+9075-2-2016.pdf]

- Man kann aber Entwürfe kostenlos bekommen, z.B.:
  - [http://www.wiscorp.com/sql200n.zip] (Stand 2006)
- Es gibt auch Grammatiken der Sprache SQL nach verschiedenen Versionen des Standards im Netz, z.B.:
  - [https://jakewheat.github.io/sql-overview/sql-2016-foundation-grammar.html]
  - [https://ronsavage.github.io/SQL/]

Inhalt

1. Geschichte des Relationalen Modells
2. SQL: Geschichte
3. SQL Standards
4. SQL in der Vorlesung
5. DBMS-Geschichte

SQL-Syntax in der Vorlesung

- SQL:2016 ist zu groß, um es hier vollständig zu behandeln.

  Ein großer Teil ist auch noch nicht in derzeitigen DBMS implementiert.

- SQL/89 ($\sim$ Einstiegs-Level von SQL-92) wird vollständig behandelt und ein Teil von SQL:2016, der in fast allen DBMS implementiert ist.

  Manchmal werden Details der SQL-Syntax spezieller Systeme erklärt, meist im Kleingedruckten. Dies ist irrelevant für Klausuren. Sie sollen einen Eindruck von der Portabilität der Konstrukte geben (und helfen, wenn man mit diesem DBMS arbeiten muss). Verwenden Sie in Klausuren möglichst nur Konstrukte, die im Skript stehen (das sollte sehr portabel sein).
  Ggf. können Sie bei der Klausureinsicht diskutieren, wenn das Konstrukt in vielen DBMS läuft, und Standard-konform ist, aber als falsch gewertet wurde.

- In der Vorlesung „Datenbank-Programmierung“ werden weitere fortgeschrittenen SQL-Konstrukte behandelt.

Bedeutung der Portabilität (1)

- Für ad-hoc Anfragen (einmal eingetippt, dann vergessen) spielt Portabilität keine Rolle.
- SQL-Anfragen werden aber auch in Programmen eingesetzt, und wir üben hier dafür.
- Dann ist sehr wohl möglich, dass man irgendwann das DBMS wechseln muss.

  Die Leistungsanforderungen steigen, oder Sie benötigen neue Features, die Ihr bisheriges DBMS nicht bietet. Für das andere DBMS gibt es besseren Support, oder mehr Expertise in Ihrer Firma. Ihre Firma wird von einer anderen Firma gekauft, die alle Daten in Oracle speichert, und das für Ihre Daten auch will. Sie wollen von einem kommerziellem DBMS zu Open Source wechseln.

- Wenn man vorher nicht auf Portabilität geachtet hat, wird der Wechsel mühsam.

  Datenbank-Managementsysteme unterscheiden sich in ihren SQL-Dialekten.

Bedeutung der Portabilität (2)

- Für mangelnde Portabilität zahlt man einen Preis:
  - Ein späterer Wechsel wird mühsam (teuer)
  - oder Sie stellen fest, dass sie nicht mit vertretbarem Aufwand wechseln können, und beim alten DBMS mit seinen Schwächen bleiben müssen.

    Auch wenn der Wechsel im Moment nicht angedacht ist, bleibt das Risiko für die Zukunft.
- Wenn man durch Nutzung DBMS-spezifischer Features Vorteile hatte, hat man einen Gegenwert für diesen Preis bekommen:
  - Die Anfragen sind kürzer (Ersparnis beim Programmieraufwand).
  - Die Anfragen laufen schneller (Ersparnis bei der Hardware).

Bedeutung der Portabilität (3)

- Aber wenn die nicht-portable Lösung keine Vorteile gegenüber einer portablen Lösung hatte, zahlt man einen Preis für nichts.

  Man zahlt für sein Unwissen von SQL. Das sollte nach dieser Vorlesung aber nicht nötig sein.

- PostgreSQL ist ein System mit vielen Erweiterungen gegenüber klassischem SQL.
- Wenn Sie irgendetwas basteln, was in PostgreSQL funktioniert, heisst das nicht, dass es auch in den meisten anderen Systemen funktioniert.
- Halten Sie sich an die in dieser Vorlesung vorgestellten Konstrukte!

Bedeutung der Portabilität (4)

- In der Vorlesung gibt es gelegentlich Anmerkungen zum Vergleich verschiedener Systeme.
- Es wird nicht von Ihnen erwartet, dass Sie auswendig lernen, was genau in welchem System funktioniert.
- Es wird von Ihnen erwartet, dass Sie einen Eindruck davon haben, was in (fast) allen Systemen funktioniert.
- Diesem gemeinsamen Kern von SQL entsprechen auch die Beispiele und Syntaxgraphen der Vorlesung.

  Erst in der Fortsetzungs-Vorlesung „Datenbank-Programmierung“ wird es schwieriger, weil dort recht neue SQL-Konstrukte behandelt werden, die bisher nur in einigen Systemen funktionieren. Sie bringen aber einen echten Vorteil (kürzere und performantere Anfragen).

Inhalt

1. Geschichte des Relationalen Modells
2. SQL: Geschichte
3. SQL Standards
4. SQL in der Vorlesung
5. DBMS-Geschichte

Geschichte von DB-Managementsystemen

- Wenn Sie sich für die Geschichte relationaler DBMS interessieren, schauen Sie sich folgenden Stammbaum an:
  [https://hpi.de/naumann/projects/rdbms-genealogy.html]

  Eine Liste von Datenbanksystemen finden Sie auch in der Wikipedia:
  [https://de.wikipedia.org/wiki/Liste_der_Datenbankmanagementsysteme]

- Die beiden ersten bekannten relationalen DBMS waren Ingres und System/R (beide ca. 1974 gestartet).

  [https://en.wikipedia.org/wiki/Ingres_(database)]
  [https://www.mcjones.org/System_R/]
  [https://people.eecs.berkeley.edu/~brewer/cs262/SystemR.pdf]

- Nicht relational geht es noch weiter zurück, z.B. ist IMS von IBM sehr bekannt (hierarchisches Modell, ab 1966).

  Es wird bis heute verwendet und weiter entwickelt.
  [https://www.ibm.com/it-infrastructure/z/ims]