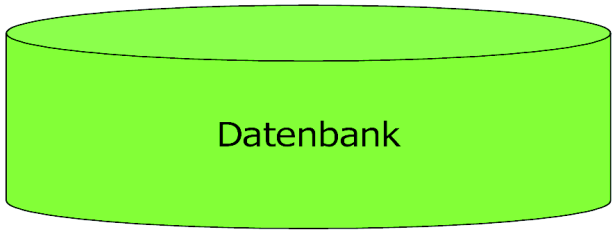
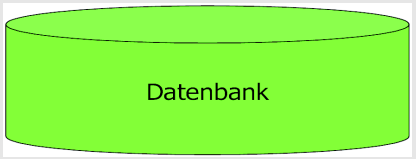
Kapitel 1: Einführung
1.1 Datenbanken?



Willkommen!



Studierenden-Datenbank

- Hans Eifrig hat die Matrikelnummer 1223. Seine Adresse ist Seeweg 20. Er ist im zweiten Semester.
- Lisa Lustig hat die Matrikelnummer 3434. Ihre Adresse ist Bergstraße 11. Sie ist im vierten Semester.
- Maria Gut hat die Matrikelnummer 1234. Ihre Adresse ist Am Bächle 1. Sie ist im zweiten Semester.

1.2 Grundbegriffe relationaler Datenbanken

Student

MatrNr	Name	Adresse	Semester
1223	Hans Eifrig	Seeweg 20	2
3434	Lisa Lustig	Bergstraße 11	4
1234	Maria Gut	Am Bächle 1	2

- Tabellarische Darstellungen dieser Art sind die Grundstrukturen *relationaler Datenbanken*.
- Begriffe: *Relation*, *Relationsbezeichner*, *Attribut*, *Tupel*, *Schlüssel*, *Primärschlüssel*.
  oder auch *Tabelle*, *Tabellenbezeichner*, *Spalte*, *Zeile*.
- Ein *Schlüssel* einer Tabelle ist eine minimale Teilmenge der Attribute der Tabelle, mittels der die einzelnen Tupel der Tabelle unterschieden werden können.
  Der Primärschlüssel ist ein ausgewählter Schlüssel - er wird durch Unterstreichen gekennzeichnet.

Struktur und Inhalt einer Relation

Die abstrakte Struktur einer Relation soll von ihrem Inhalt getrennt werden.

- *Relationsschema*. Struktur der Relation `Student`:

  `Student(MatrNr, Name, Adresse, Semester)`

- *Relationsinstanz*. Inhalt/Zustand der Relation mit Schema `Student`:

Student

MatrNr	Name	Adresse	Semester
1223	Hans Eifrig	Seeweg 20	2
3434	Lisa Lustig	Bergstraße 11	4
1234	Maria Gut	Am Bächle 1	2

- Schlüssel der Relationen sind Teil des Schemas - sie sind für alle Instanzen des Schemas gültig.

Beispiel: Miniwelt Vorlesungsverwaltung

Der Kurs K010 über Datenbanken wird vom Institut DBIS angeboten. Er behandelt die Grundlagen von Datenbanken.

Der Kurs K011 über Informationssysteme wird vom Institut DBIS angeboten. Er baut auf Datenbanken auf und behandelt Grundlagen von Informationssystemen.

Student Hans Eifrig hat Datenbanken im WS2003/04 mit der Note 2.0 bestanden.

Professor Lausen hat die Personalnummer 153062 und ist am Institut DBIS.

Professor Lausen hält die Vorlesung Datenbanken im WS2003/04.

```
Student(MatrNr, Name, Adresse, Semester)
Kurs(KursNr, Institut, Name, Beschreibung)
Belegung(MatrNr, KursNr, Semester, Note)
Professor(PersNr, Name, Institut)
Lehrangebot(PersNr, KursNr, Semester)
```


Beispiel (Ausschnitt)

Student

MatrNr	Name	Adresse	Semester
1223	Hans Eifrig	Seeweg 20	2
3434	Lisa Lustig	Bergstraße 11	4
1234	Maria Gut	Am Bächle 1	2

Kurs

KursNr	Institut	Name	Beschreibung
K010	DBIS	Datenbanken	Grundlagen von Datenbanken
K011	DBIS	Informationssysteme	Grundlagen von Informationssystemen

Belegung

MatrNr	KursNr	Semester	Note
1223	K010	WS2003/2004	2.3
1234	K010	SS2004	1.0

Datenbankschema und Datenbankinstanz

- Die Menge der Relationsschemata einer Miniwelt ergibt das (relationale) *Datenbankschema* der Miniwelt.
- Eine Menge von Instanzen der Relationen eines Datenbankschemas nennen wir *Datenbankinstanz*.

  Die Elemente einer Datenbankinstanz müssen sich auf denselben Zustand der betreffenden Miniwelt beziehen.

1.3 Modellierung einer Miniwelt

Objekte und Beziehungen

- Objekte bilden die elementare Grundlage unserer Betrachtung.

  Objekte werden durch Tupel in Relationen repräsentiert und können somit durch Schlüsselwerte identifiziert werden.
- Beziehungen sind über Objekten oder anderen Beziehungen definiert; sie entstehen somit durch In-Bezug-Setzen von Objekten.

  Objektreferenzen in Beziehungen: *Fremdschlüssel*.
- Menge der als relevant betrachteten Objekte und Beziehungen: *Miniwelt*.
- Objekte und Beziehungen werden getypt.
  Die Menge der Objekt- und Beziehungstypen einer Miniwelt ergibt das *konzeptuelle Schema* der Miniwelt.

Beziehung Belegung zwischen Student und Kurs

Belegung

MatrNr	KursNr	Semester	Note
1223	K010	WS2003/2004	2.3
1234	K010	SS2004	1.0

- Schlüssel der Relation Belegung ist MatrNr und KursNr zusammen.
- Fremdschlüssel sind einmal MatrNr und zum anderen KursNr.

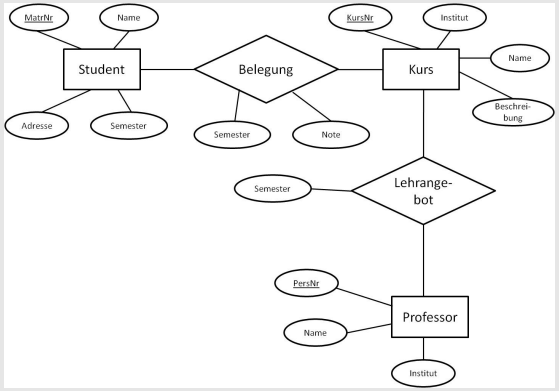
Entity-Relationship-Modell

Ein populärer graphischer Formalismus zur Abfassung eines konzeptuellen Schemas ist das *Entity-Relationship-Modell* (ER-Modell).

- Entities (Objekte) werden getypt; jeder Entity-Typ wird durch ein Rechteck dargestellt.
- Relationships (Beziehungen) werden getypt; jeder Relationship-Typ wird durch eine Raute dargestellt.
- Attribute werden durch Ovale dargestellt.

Die Menge der Entity-Typen und Relationship-Typen einer Miniwelt ergibt das *ER-Schema* der Miniwelt.

ER-Schema der Vorlesungsverwaltung



- Die Mengen der konkreten Objekte und Beziehungen eines Zustandes der Mniwelt werden in Form von Tabellen/Relationen dargestellt.
- Im ER-Schema werden die Fremdschlüssel der Relationship-Typen nicht angegeben, da sie aus den Kanten geschlossen werden können.

1.4 Arbeiten mit einer Datenbanken

- Anwendungsprogramme kommunizieren mit einer Datenbank, indem sie Anfragen über den gespeicherten Zustand der Miniwelt stellen, bzw. diesen Zustand durch Ändern, Einfügen oder Löschen von Daten verändern.

  Besonders von Interesse sind *Anfragen* (engl. queries) an eine Datenbank.
- Ausdrücke einer *Datenbankanfragesprache* haben eine *mengenwertige, deklarative Semantik*.
  - Das Ergebnis einer Anfrage ist eine Menge von Tupeln.
  - Die Anfrage definiert, was für Zusammenhänge aus den Daten der Datenbank gebildet werden sollen, ohne dass die algorithmische Vorgehensweise hierzu spezifiziert werden muss.
- Anfrageoptimierer.

Beispiele:

Was wissen wir über die Studierenden?

```
SELECT *
FROM Student
```

Wie heißen die Professoren des Instituts 'DBIS'?

```
SELECT Name
FROM Professor
WHERE Institut = 'DBIS'
```

Wie heißen die Studierenden, die den Kurs 'K010' belegt haben?

```
SELECT S.Name
FROM Student S, Belegung B
WHERE S.MatrNr = B.MatrNr AND B.KursNr = 'K010'
```

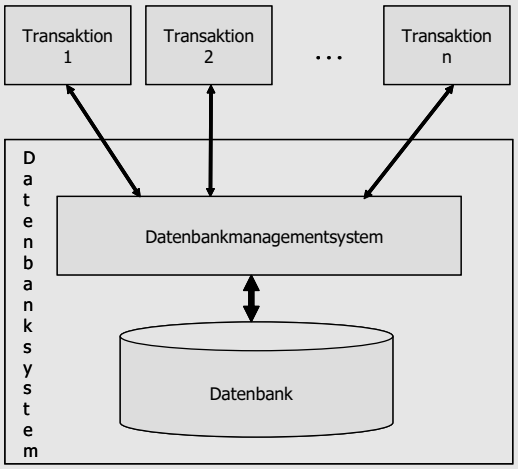
Welche Studierenden belegen welche Kurse?

```
SELECT S.Name, K.Name
FROM Student S, Belegung B, Kurs K
WHERE S.MatrNr = B.MatrNr AND B.KursNr = K.KursNr
```


... allgemein redet man von Transaktionen

- Eine Ausführung (Prozess) eines Anwendungsprogramms, bzw. einer Anfrage über einer Datenbank wird als *Transaktion* bezeichnet.
- Enthalten Transaktionen Änderungs-, Einfügungs- oder auch Löschoperationen, so transformieren sie einen gegebenen Datenbankzustand in einen neuen Datenbankzustand.

1.5 Basisarchitektur



1.6 empfohlene Lektüre

Technology | DOI:10.1145/1735223.1735231 Gary Anthes

Happy Birthday, RDBMS!

The relational model of data management, which dates to 1970, still dominates today and influences new paradigms as the field evolves.

FORTY YEARS AGO this June, an article appeared in these pages that would shape the long-term direction of information technology like few other ideas in computer science. The opening sentence of the article, "A Relational Model of Data for Large Shared Data Banks," summed it up in a way as simple and elegant as the model itself: "Future users of large data banks must be protected from having to know how the data is organized in the machine," wrote Edgar F. Codd, a researcher at IBM.

And protect them it did. Programmers and users at the time dealt mostly with crude homegrown database systems or commercial products like IBM's Information Management System (IMS), which was based on a low-level, hierarchical data model. "These databases were very rigid, and they were hard to understand," recalls Ronald Fagin, a Codd protégé and now a computer scientist at IBM Almaden Research Center. The hierarchical "trees" in IMS were brittle. Adding a single data element, a common occurrence, or even tuning changes, could involve major reprogramming. In addition, the programming language

"People were stunned to learn that complex, page-long [IMS] queries could be done in a few lines of a relational language," says Raghu Ramakrishnan, chief scientist for audience and cloud computing at Yahoo!

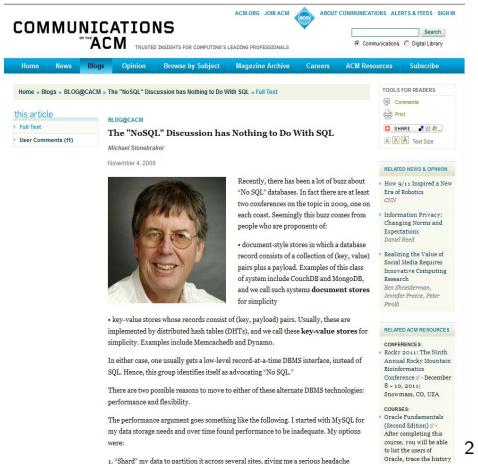
Codd's model came to dominate a multibillion-dollar database market, but it was hardly an overnight success. The model was just too simple to work, some said. And even if it did agement System (IDMS) from the company that would become Cullinet.

Contentious debates raged over the models in the CS community through much of the 1970s, with relational enthusiasts arrayed against CODASYL advocates while IMS users coasted along on waves of legacy software.

As brilliant and elegant as the relational model was, it might have remained confined to computer science curricula if it wasn't for three projects aimed at real-world implementation of the relational database management system (RDBMS). In the mid-1970s, IBM's System R project and the University of California at Berkeley's Ingres project set out to translate the relational concepts into workable, maintainable, and efficient computer code. Support for multiple users, locking, logging, error-recovery, and more were developed.

System R went after the lucrative mainframe market with what would become DB2. In particular, System R produced the Structured Query Language (SQL), which became the de facto standard language for relational databases. Meanwhile Ingres was aimed at UNIX machines and Digital Equipment Corp.

1



COMMUNICATIONS of the ACM

Home | News | Blogs | Opinion | Browse by Subject | Magazine Archive | Careers | ACM Resources | Subscribe

Home » Blogs » BLOG@CACM » The "NoSQL" Discussion has Nothing to Do With SQL » Full Text

BLOG@CACM

The "NoSQL" Discussion has Nothing to Do With SQL

Michael Stonebraker

November 4, 2009

Recently, there has been a lot of buzz about "No SQL" databases. In fact there are at least two conferences on the topic in 2009, one on each coast. Seemingly this buzz comes from people who are proponents of:

• document-style stores in which a database record consists of a collection of (key, value) pairs plus a payload. Examples of this class of system include CouchDB and MongoDB, and we call such systems **document stores** for simplicity

• key-value stores whose records consist of (key, payload) pairs. Usually, these are implemented by distributed hash tables (DHTs), and we call these **key-value stores** for simplicity. Examples include Memcachedb and Dynamo.

In either case, one usually gets a low-level record-at-a-time DBMS interface, instead of SQL. Hence, this group identifies itself as advocating "No SQL."

There are two possible reasons to move to either of these alternate DBMS technologies: performance and flexibility.

The performance argument goes something like the following. I started with MySQL for my data storage needs and over time found performance to be inadequate. My options were:

1. "Shard" my data to partition it across several sites. Giving me a serious headache

2

[1] In: Communications of the ACM, Volume 53 , Issue 5 (May 2010). Kann aus dem Institutsnetz heraus vom ACM-Portal heruntergeladen werden.

[2] http://cacm.acm.org/blogs/blog-cacm/50678-the-nosql-discussion-has-nothing-to-do-with-sql/fulltext