Datenbanken und Informationssysteme

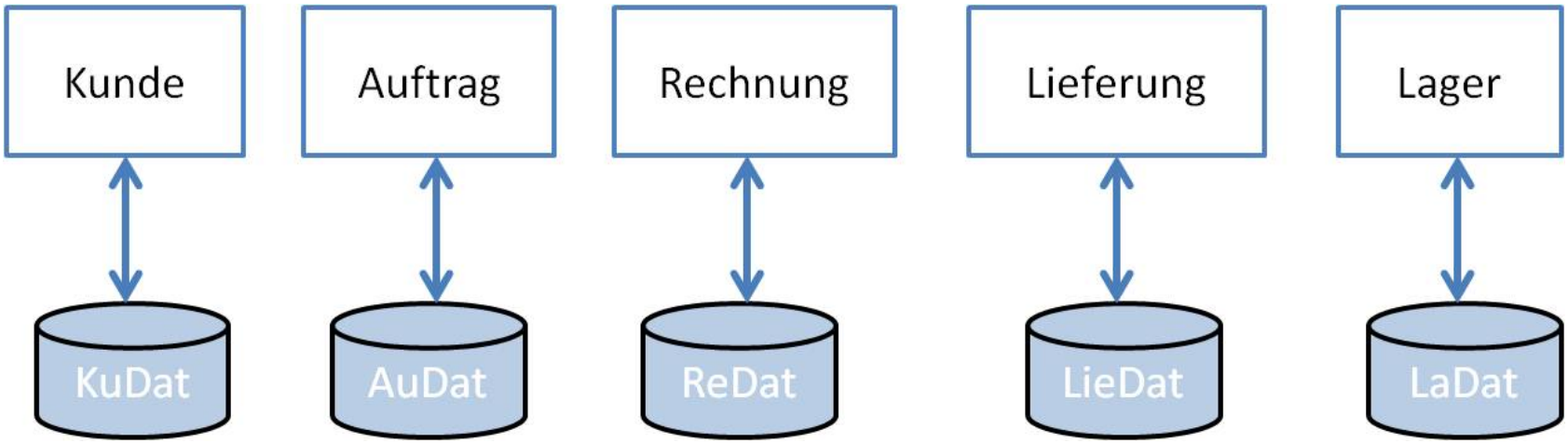
Georg Lausen

Lehrstuhl für Datenbanken und Informationssysteme
Universität Freiburg

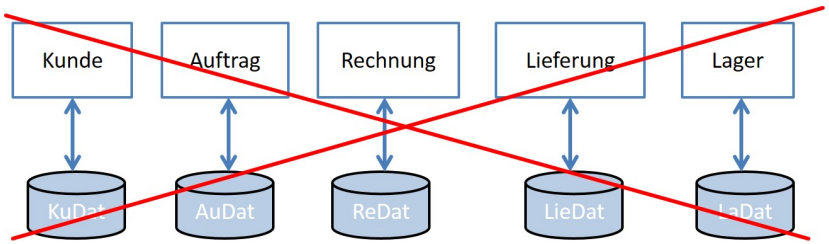
WS 2017/2018

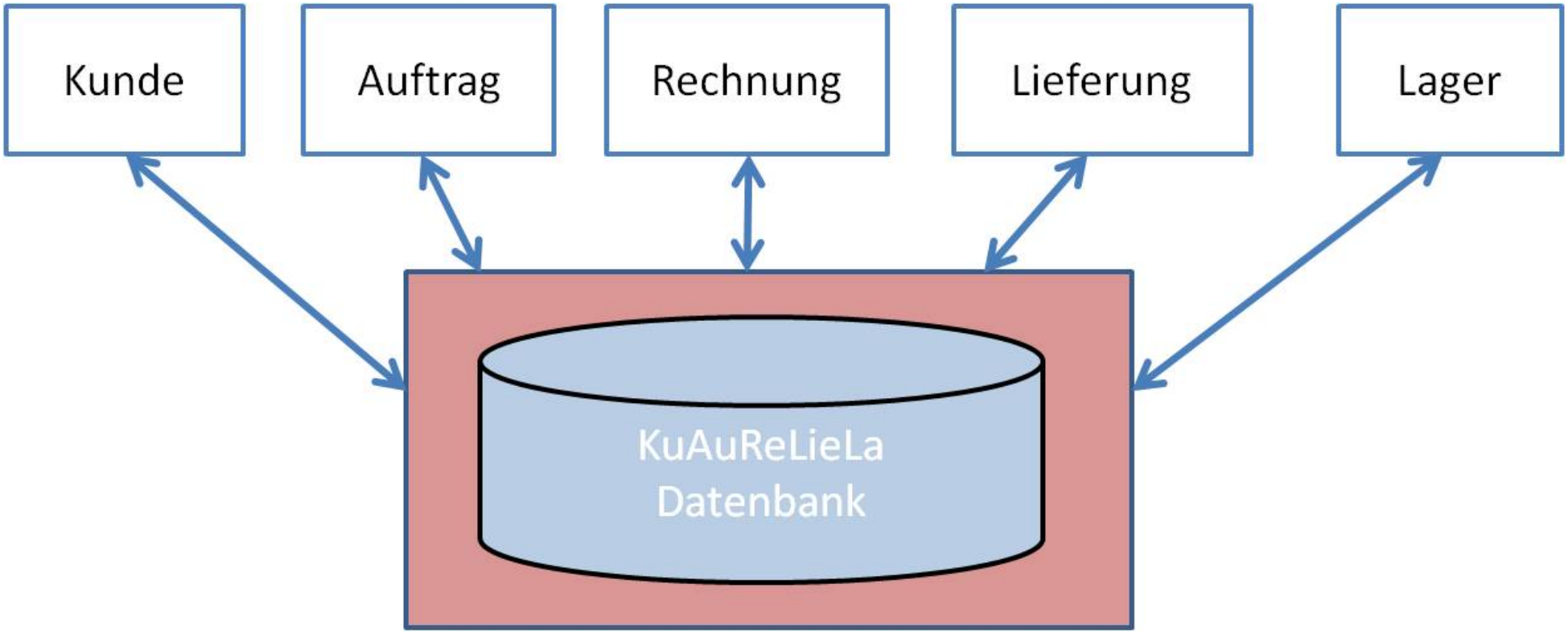
Datenbanken warum?

Konventionelle Datenverarbeitung kommt mit vielen Problemen:



... Datenbanken sind die Lösung!





Gliederung:

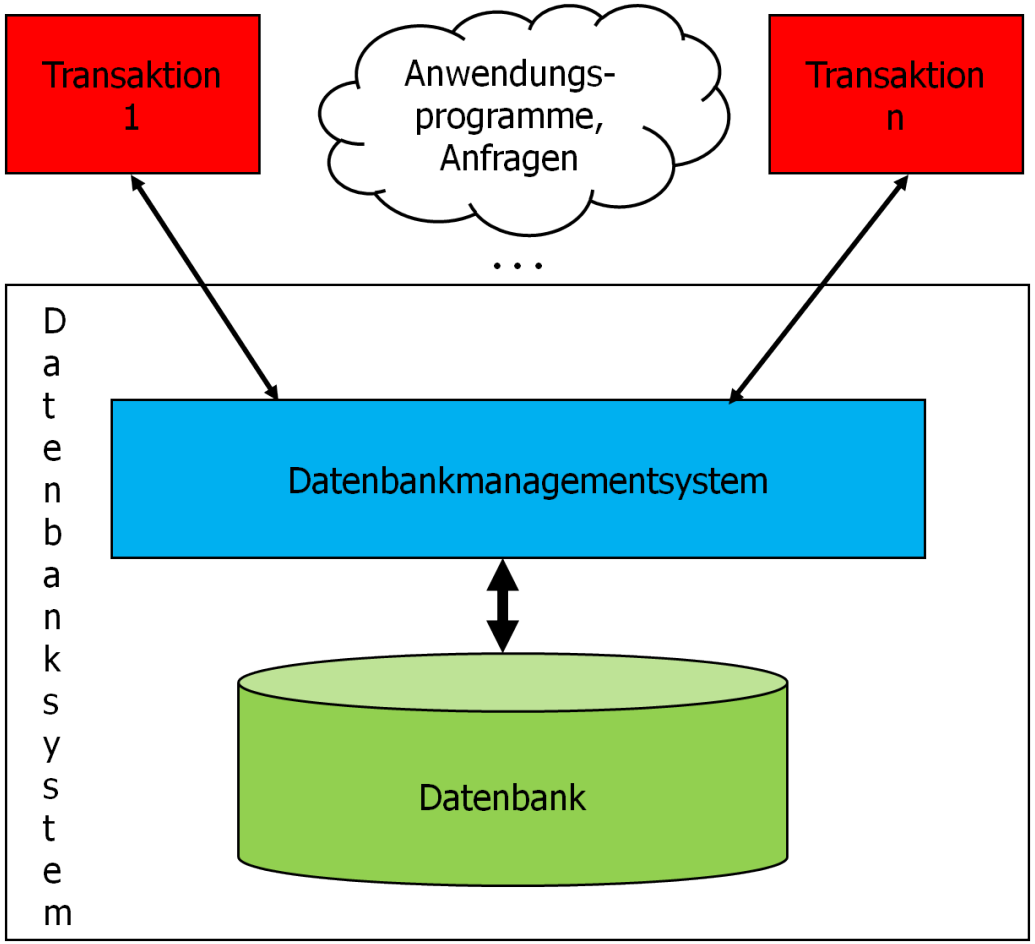
Literatur:

- *Datenbanken: Grundlagen und XML-Technologien.* Georg Lausen. Elsevier Spektrum Akademischer Verlag, 2005.
- *Datenbanken - Konzepte und Sprachen.* Andreas Heuer, Gunter Saake. International Thomson Publishing, 2. Auflage, 2000.
- *Datenbanksysteme - Eine Einführung.* Alfons Kemper, Andreas Eickler. Oldenbourg, 7. Auflage, 2009.
- *Datenmodelle, Datenbanksprachen und Datenbank-Management-Systeme.* Gottfried Vossen. Oldenbourg, 5. Auflage, 2008.

- *Database Management Systems.* Raghu Ramakrishnan, Johannes Gehrke. McGraw-Hill Higher Education, 3rd. Edition.

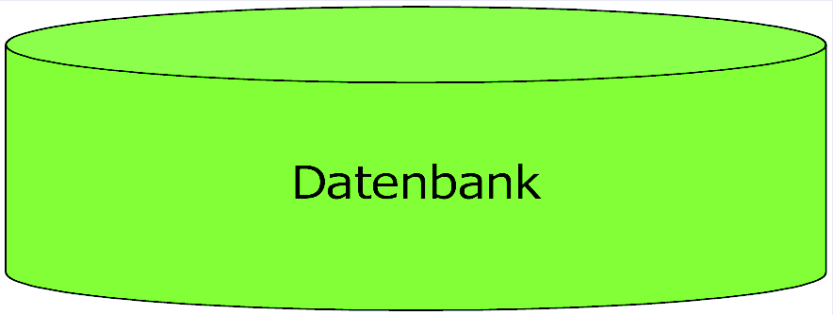
Kapitel 1: Einführung

1.1 Datenbanken?



1.2 Grundbegriffe relationaler Datenbanken

Repräsentieren von Daten einer Miniwelt



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.

Repräsentation und Strukturierung von Daten in Relationen

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 Relation ist eine minimale Teilmenge der Attribute der Relation, mittels der die einzelnen Tupel der Relation 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

- Die identifizierende Eigenschaft des Primärschlüssels muss für jede Instanz der Relation gewährleistet sein.

Beispiel Student, Kurs und Belegung

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 Arbeiten mit einer Datenbanken

- Anwendungsprogramme (Transaktionen) 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.
  - Es existiert eine standardisierte Anfragesprache: SQL.
- Anfrageoptimierer.

Anfragen in SQL, der Anfragesprache für Datenbanken:

Was wissen wir über die Studierenden?

```
SELECT *
FROM Student
```

Welche Matrikelnummern haben die Studierenden, die den Kurs 'K010' belegt haben?

```
SELECT MatrNr
FROM Belegung
WHERE KursNr = 'K010'
```

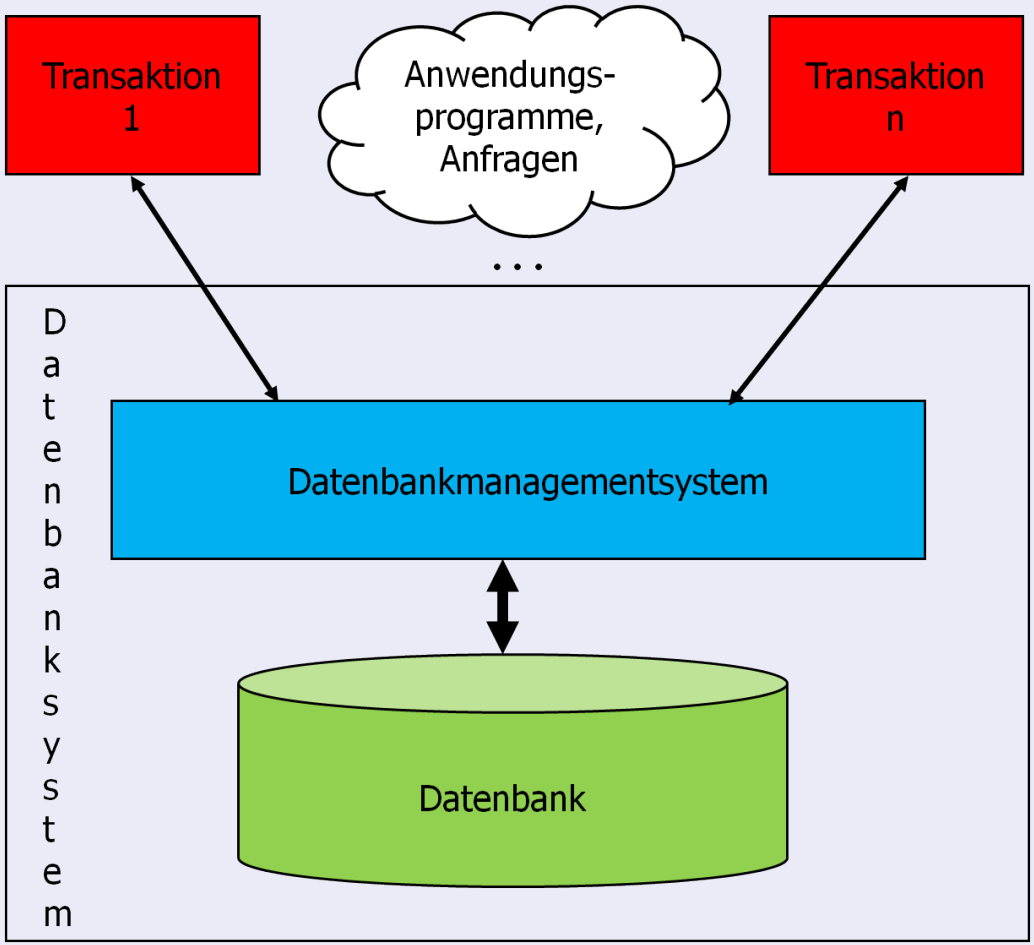
Wie heißen die Studierenden, die den Kurs 'K010' belegt haben?

```
SELECT S.Name
FROM Student, Belegung
WHERE Student.MatrNr = Belegung.MatrNr AND Belegung.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
```

1.4 Basisarchitektur



1.5 Diskussion

Abkürzungen

- - Datenbank: DB
- - Datenbanksystem: DBS
- - Datenbankmanagementsystem: DBMS

Vorteile eines DBS

- *Mehrfachnutzung der Daten.* Dateninseln, in denen einzelne Benutzer ihre Daten isoliert verwalten werden vermieden. Somit kann Redundanz der Daten ausgeschlossen werden und eine Standardisierung der Datenformate erreicht werden.
- *Unabhängigkeit von Programmen und Daten.* Programme kommunizieren mit der DB über vom DBMS angebotene Interfaces; Änderungen in den Strukturen der DB können vor den Programmen weitgehend verborgen werden.
- *Gewährleistete Datenintegrität.* Das DBMS hat das Wissen über die zulässigen Zustände der DB und kann Verletzungen der Integrität ausschließen.

weitere Vorteile eines DBS

- *Kontrollierter Mehrbenutzerbetrieb.* Zeitlich überlappender Zugriff unterschiedlicher Programme zu gemeinsamen Daten kann durch das DBMS kontrolliert werden.
- *Datenpersistenz.* Datenverlust durch Hardware- und Softwarefehler kann im laufenden Betrieb repariert werden; das DBMS verwaltet selbst die hierzu erforderlichen Informationen.
- *Datensichten.* Unterschiedliche Benutzergruppen benötigen auf ihre Bedürfnisse abgestimmte Sichten auf das zentrale konzeptuelle Schema. Das DBMS kann die erforderlichen Abbildungen zur Verfügung stellen.
- *Autorisierter Datenzugriff.* Zugriff zu den Daten ist nur über das DBMS möglich. Damit kann nicht autorisierter Zugriff vermieden werden.

Nachteile eines DBS

- *Komplexität.* Die effiziente und sichere Nutzung erfordert speziell ausgebildetes Personal und verursacht erhebliche Hardware- und Software-Kosten.
- *eingeschränkte Effizienz.* DBS sind universelle Softwaresysteme, die für unterschiedlichste Arten von Anwendungen die geforderte Leistung zur Verfügung stellen wollen. Für spezielle Anwendungen kann ihre Leistungsfähigkeit deutlich geringer sein im Vergleich zu für diese Anwendungen spezialisierte Systeme. Die Diskussionen hierzu hierzu laufen unter dem Stichwort *NoSQL*.

  NoSQL-Systeme erreichen die Effizienzsteigerung typischerweise durch Aufgabe von für Datenbanken existentieller Eigenschaften wie die Gewährleistung der Datenintegrität oder der kontrollierte Mehrbenutzerbetrieb.

Die Lösung: NewSQL

Basierend auf modernen Multicore-Systemen mit sehr großem Interspeichern (> 1 TB) oder Computer-Clustern holen aktuell sogenannte *NewSQL*-Systeme wieder auf, wobei sie im Unterschied zu NoSQL die volle Funktionalität bieten.

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

DOI:10.1145/2366316.2366319 http://cacm.acm.org/blogs/blog-cacm

New Opportunities for New SQL

Michael Stonebraker expects a substantial increase in the number of New SQL engines using a variety of architectures in the near future.

Michael Stonebraker "New SQL: An Alternative to NoSQL and Old SQL for New OLTP Apps"
http://cacm.acm.org/blogs/blog-cacm/109710
June 16, 2011

Historically, Online Transaction Processing (OLTP) was performed by customers submitting traditional transactions (order something, withdraw money, cash a check, etc.) to a relational DBMS. Large enterprises might have dozens to hundreds of these systems. Invariably, enterprises wanted to consolidate the information in these OLTP systems for business analysis, cross selling, or some other purpose. Hence, Extract-Transform-and-Load (ETL) products were used to convert OLTP data to a common format and load it into a data warehouse. Data warehouse activity rarely shared machine resources with OLTP because of lock contention in the DBMS and because business intelligence (BI) queries were so resource-heavy that they got in the way of timely responses to transactions.

This combination of a collection of OLTP systems, connected to ETL, and connected to one or more data warehouses is the gold standard in enterprise computing. I will term it "Old OLTP." By and large, this activity was supported by the traditional RDBMS vendors. In the past I have affectionately called them "the elephants"; in this posting I refer to them as "Old SQL."

As noted by most pundits, "the Web changes everything," and I have noticed a very different collection of OLTP requirements that are emerging for Web properties, which I will term "New OLTP." These sites seem to be driven by two customer requirements:

The need for far more OLTP throughput. Consider new Web-based applications such as multiplayer games, social networking sites, and online gambling networks. The aggregate number of interactions per second is skyrocketing for the successful Web properties in this category. In addition, the explosive growth of smartphones has created a market for applications that use the phone as a geographic sensor and provide location-based services. Again, successful applications are seeing explosive growth in transaction requirements. Hence, the Web and smartphones are driving the volume of interactions with a DBMS through the roof, and New OLTP developers need vastly better DBMS performance and enhanced scalability.

The need for real-time analytics. Intermixed with a tidal wave of updates is the need for a query capability. For example, a Web property wants to know the number of current users playing its game, or a smartphone user wants to know "What is around me?" These are not the typical BI requests to consolidated data, but rather real-time inquiries to current data. Hence, New OLTP requires a real-time query capability.

In my opinion, these two characteristics are shared by quite a number of enterprise non-Web applications. For example, electronic trading firms often trade securities in several locations around the world. The enterprise wants to keep track of the global position for each security. To do so, all trading actions must be recorded, creating a fire hose of updates. Furthermore, there are occasional real-time queries. Some of these are triggered by risk exposure—i.e., alert the CEO if the aggregate risk for or against a particular security exceeds a certain monetary threshold. Others come from humans, e.g., "What is the current position of the firm with respect to security X?"

Hence, we expect New OLTP to be a substantial application area, driven by Web applications as the early adopters. These applications will be followed by more traditional enterprise systems. Let's look at the deployment options.

1. Traditional OLTP. This architec-

2

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

[2] In: Communications of the ACM, Volume 55 , Issue 11 (May 2012). Kann aus dem Institutsnetz heraus vom ACM-Portal heruntergeladen werden.