Unit 0x02
Grundlagen Datenbanksysteme

Datenbanksysteme

Q&A

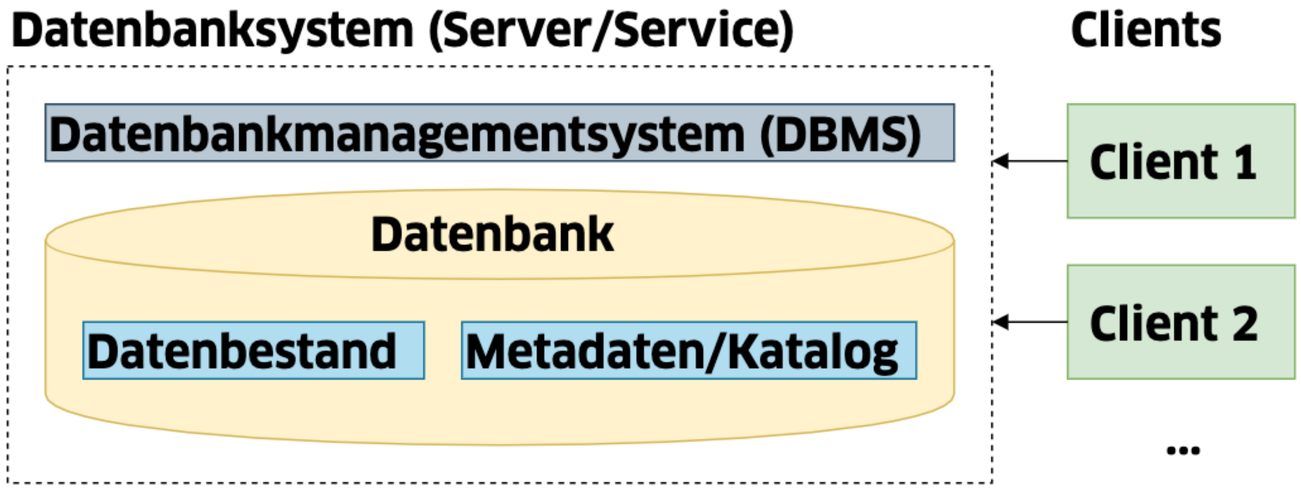
Was ist eine Datenbank, ein Datenbanksystem oder ein Datenbankmanagementsystem genau?

- Definition?
- Typen?
- Content?
- Anforderungen?
- Aufbau/Komponenten?

Datenbanksysteme – Definitionen

Begriffsbestimmung DBS, DBMS, Datenbank

- Ein Datenbanksystem (DBS) besteht aus dem Datenbankmanagementsystem (DBMS), d.h. der Software, und den Daten, also der Datenbank.
- Die Clients verbinden sich mit dem DBMS und kommunizieren über eine spezielle Datenbank-sprache, z.B. SQL, um Daten der Datenbank abzufragen, zu speichern oder zu administrieren.



- Im Sprachgebrauch wird zwischen DBS, DBMS, 'Datenbankverwaltungssystem' oder auch nur 'Datenbank' oft nicht unterschieden.
- 'Datenbank' meint in unserem Kontext die Menge der Daten.

Datenbanksysteme – Definitionen

Charakterisierung DBMS

- Ein Datenbankmanagementsystem (DBMS) ist eine Software zur *sicheren*, *konsistenten* und *persistenten* Speicherung großer Datenmengen, mit dem Ziel *mehreren Benutzern* (gleichzeitig) *effizienten*, *zuverlässigen*, *sicheren* und *bequemen* Zugriff auf diese Daten zu ermöglichen. ➜ **Anforderungen an ein DBS bzw. DBMS**
- Es besteht aus Komponenten zur Abfrage, Verwaltung und Administration der Daten bzw. Datenbank. ➜ **Aufbau DBMS**

Datenbanksysteme – Typen

DBS bzw. DBMS Typen

Je nach Art der Daten und des Anwendungsfalls eignen sich unterschiedliche Typen von DBS, genauer DBMS, unterschiedlich gut. Da sind beispielsweise

- **relationale DBMS,** die Objekten gleicher Struktur (Attribute) in Tabellen verwalten.

Oder aber

- **hierarchische oder objektorientierte DBMS,** die Objekten mit Eltern-Kind- oder Vererbungsbeziehungen besitzen und einzelne Strukturen gemeinsam haben.

Und auch

- **dokumentenorientiert DBMS,** die Objekte ohne vergleichbare Strukturen verwalten.

Es gibt noch mehr Typen und in der Praxis sind es häufig Mischformen…

Datenbanksysteme – Typen

RDBMS

- Am verbreitetsten sind **relationale DBMS (RDBMS)**, weswegen das auch ein großer Schwerpunkt der Vorlesung ist. Damit einher geht SQL als Abfragesprache.
- Der Erfolg und die lange Dominanz der RDBMS begründet sich sicherlich in einer "normierten" Abfragesprache und der vergleichsweise einfachen und effizienten Behandlung von Objekten gleichen Aufbaus in Tabellen. Und in der Praxis sind viele Daten strukturiert und typisiert.
- Hinzu kommt, dass andere Anwendungsfälle, etwa hierarchische Daten, mit entsprechendem Overhead, dennoch in RDBMS abgebildet werden können. Dies und weitere nicht-relationale DBMS behandeln wir am Ende der Vorlesung.

Fragestellungen rund um Datenbanken, etwa Definitionen, Anforderungen oder Aufbau sind im Kern unabhängig vom konkreten DBS-Typ.

Datenbanksysteme – Content

’Daten’ bzw. ’Datum’

- Daten sind zunächst Folgen von Zeichen (Zahlen, Buchstaben, Symbole) oder auch strukturierte Objekte.
- Sie folgen einer bestimmten Syntax.
- Beispiel: 112

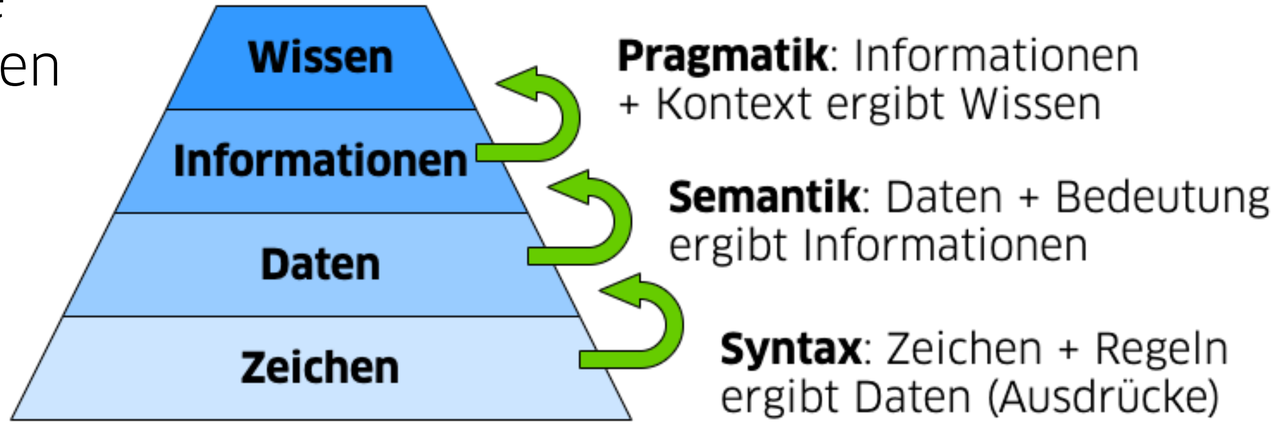
’Informationen’

- Aus Daten entstehen Informationen, wenn die Bedeutung bekannt ist.
- Durch die Hinzunahme von Informationen, ggf. weiteren Daten, kann das ursprüngliche Datum interpretiert bzw. verstanden werden.
- Beispiel: ’112’ öffnet den Haussafe

Datenbanksysteme – Content

’Wissen’

- Wissen entsteht durch systematische Verknüpfung von Daten, Informationen und (subjektiver) Erfahrung.
- Beispiel: ’112’ öffnet den Haussafe, aber wir wissen, dass das kein sicheres Passwort ist, da der Safe bei der Feuerwehr steht…



Achtung: Hier gibt es im Detail durchaus unterschiedliche Definitionen.

Q&A

Was enthalten Datenbanken nun?

- Daten, Informationen oder Wissen?

Datenbanksysteme – Anforderungen an ein DBMS

Redundanz und Konsistenz

- Beispiel redundanter Daten: n-mal ’Heise’ in der Datenbank.
- Beispiel inkonsistenter Daten: Thema ’C#’ und (zweites?) Thema ’CSharp’.
- Redundante Daten ermöglichen inkonsistente Daten und belegen Speicher.

Integrität und Transaktionen

- Beispiel korrumpierter Daten: ’Bild’ im Heise-Verlag, Gründung ’2089’.
- Transaktionen sichern mehrere Datenbewegungen als eine Aktion ab, die ganz oder gar nicht ausgeführt wird, Beispiel: 100€ von einem Konto auf ein anderes buchen.
- Fehlende Transaktionen ermöglichen korrupte Daten.

Datenbanksysteme – Anforderungen an ein DBMS

Relationen/Verknüpfungsmöglichkeiten

- Beispiel von Daten mit Beziehung: Magazin *erscheint im* Verlag.
- In jeder ernsthaften Datenmenge gibt es abzubildende Beziehungen.

Zugriffsberechtigungen

- Beispiel fehlender Rechte: Student trägt Note selber ein (wie praktisch…).

Mehrbenutzerbetrieb

- Beispiel Single-User-Betrieb: Bitte noch warten bis zum Aufgeben Ihrer Bestellung, ein Benutzer aus Brasilien ist eingeloggt.
- Gerade mit Blick auf Transaktionen gibt es hier ein großes Fehlerpotenzial.

Datenbanksysteme – Anforderungen an ein DBMS

Persistenz und Datensicherung

- Beispiel nicht-persistenter Datensicherung (aus MySQL Dokumentation):
  "The BLACKHOLE storage engine acts as a “black hole” that accepts data but throws it away and does not store it. Retrievals always return an empty result."
- Beispiel fragwürdiger Datensicherung:
  Bei Microsofts Tochterfirma 'Danger' ist der Name offenbar Programm […]
  (Spiegel 10/2009)

”Bedauerlicherweise müssen wir Sie nun darüber informieren, dass, basierend auf der letzten Einschätzung zur Datenwiederherstellung von Microsoft/Danger, persönliche Daten, die Sie auf Ihrem Sidekick gespeichert haben, mit höchster Wahrscheinlichkeit auf Grund eines Server-Fehlers bei Microsoft/Danger verlorengegangen sind.”

Datenbanksysteme – Anforderungen an ein DBMS

Kennzahlen DBMS, Beispiele

- Status
- User
- Performanceanalyse
- Abfrageanalyse
- ..

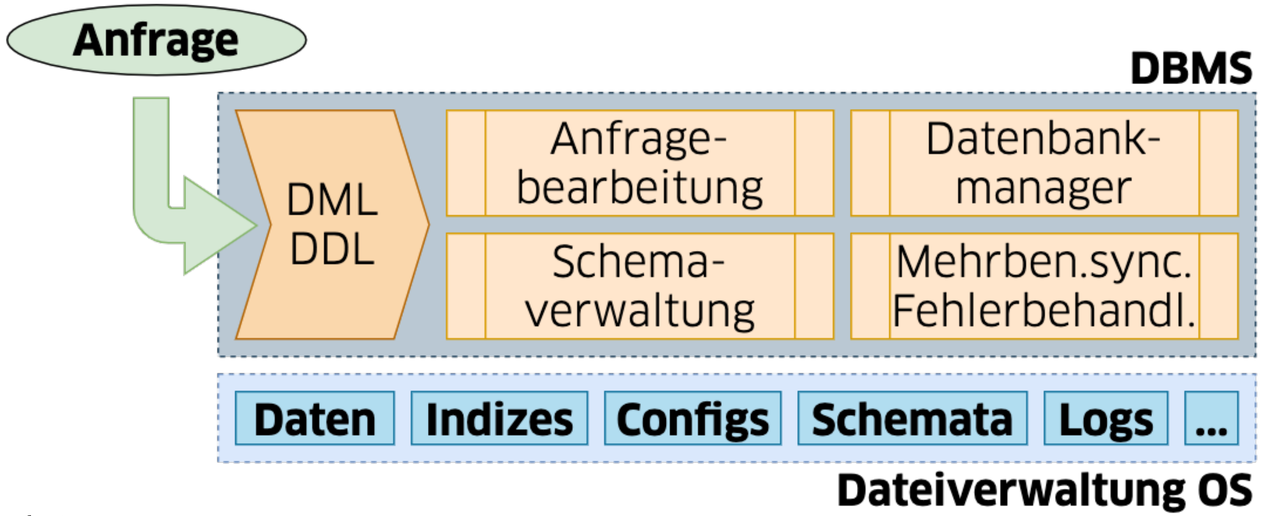
Datenbanksysteme – Anforderungen an ein DBMS

Aus den Anforderungen abgeleitete Themen der Vorlesung

- Modellierung eines realen Aspekts (Mini-Welt) in einem DBMS führt zu Abstraktions-ebenen und in der Folge zu ER-Diagrammen, Entitäten, Relationen und DBMS-Strukturen.
- Redundanzen, bzw. mögliche Anomalien, führen zum Konzept der Normalisierung, also einem Qualitätsmass für eine Modellierung bzw. für DBMS-Strukturen.
- Zugriffsberechtigungen und Mehrbenutzerbetrieb erfordern Authentisierung, Transaktionen und Abstraktionen wie Sichten.
- Persistenz führt zu unterschiedlichen Storage-Engines und Indizes, also physische Abbildungen der Daten mit Blick auf Speicher und Performance, d.h. der Diskussion um den internen Aufbau eines DBMS.

Datenbanksysteme – Aufbau

Komponenten DBMS

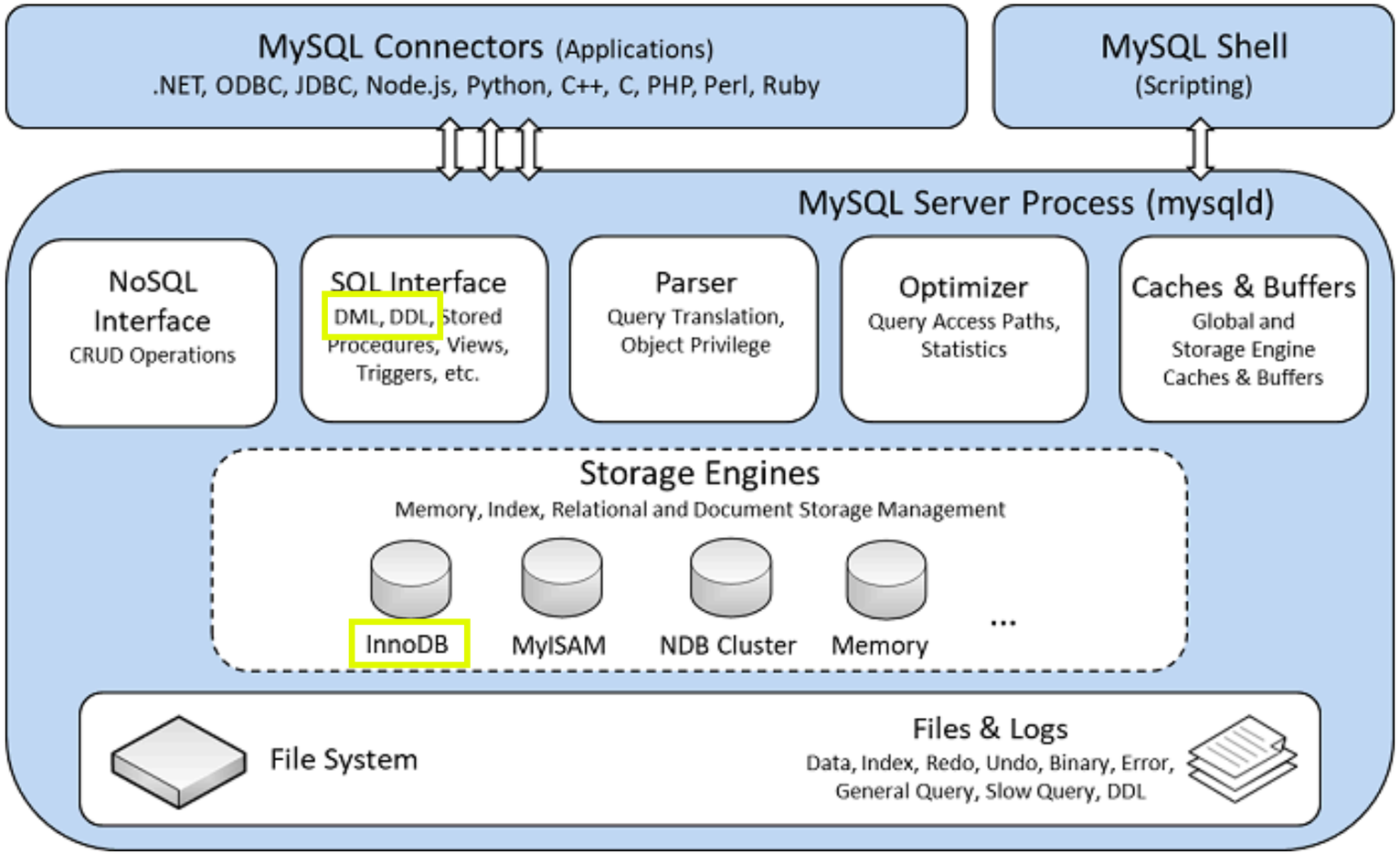


- **Data Manipulation Language (DML)**
  Compiler: Überführt Anfrage in ausführbare Form, primär zur Manipulation von Daten.
- **Data Definition Language (DDL)**
  Compiler: Wie DML, nur für Datenstrukturen.
- **Anfragebearbeitung:** Erstellen eines Ablaufplans für eine Abfrage inkl. logischer und physischer Optimierung.
- **Datenbankmanager:** Kern des DBMS, Ausführung der Anfragen.
- **Schemaverwaltung:** Verwaltung der Metadaten, 'Typsystem' des DBMS.
- **Mehrbenutzersynchronisation, Fehlerbehandlung:** Transaktionsverwaltung.

Theoretischer Aufbau

Datenbanksysteme – Aufbau

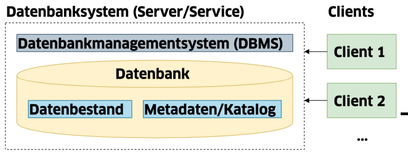
Beispiel MySQL Architektur



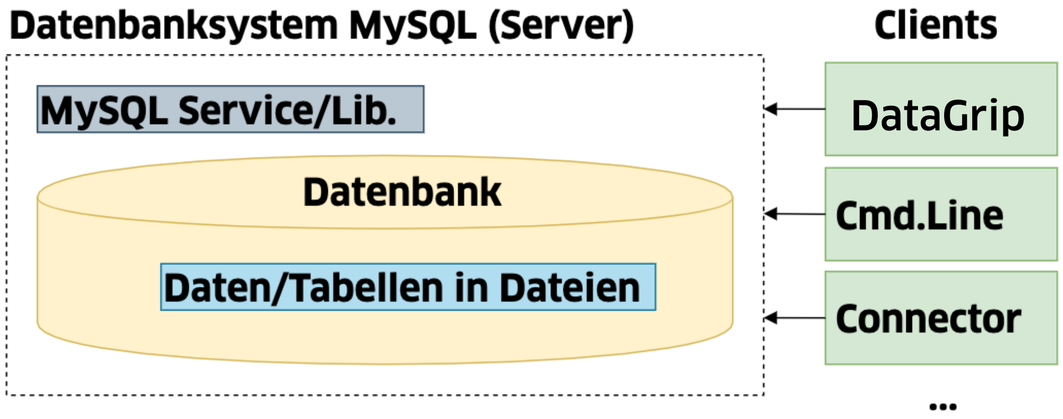
MySQL Dokumentation 'MySQL Architecture with Pluggable Storage Engines'

Realer Aufbau

Datenbanksysteme – Aufbau



Beispiel Storage Engine MySQL-InnoDB



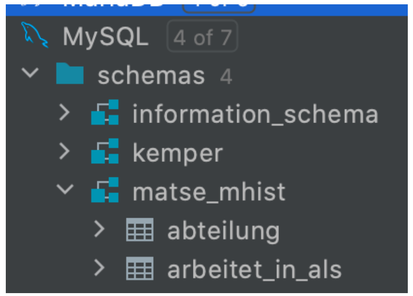
Metadaten

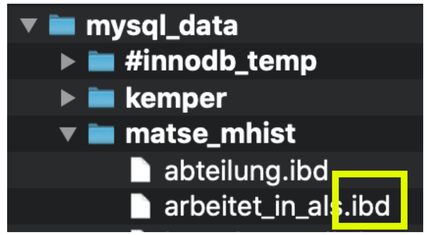
```
SELECT engine,TABLE_COLLATION,CREATE_TIME
FROM information_schema.tables
WHERE table_schema = 'matse_mhist'
    AND table_name = 'produkt';
```

Output | information_schema.TABLES

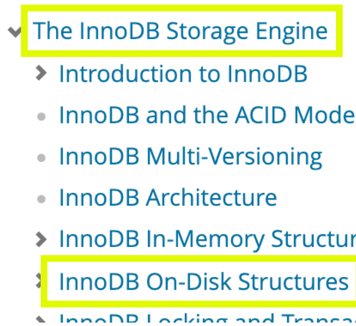
1 row | Tx: Auto | DDL | DML

ENGINE	TABLE_COLLATION	CREATE_TIME
InnoDB	utf8_general_ci	2020-10-12 21:36:21





Persistenz Tabellen
DataGrip vs. Dateisystem



MySQL Dok. InnoDB
Formate und *.idb

Alternative
Engines

Datenbanksysteme – Aufbau

Beispiel MySQL-InnoDB Type Storage Requirements

Numeric Type Storage Requirements

Data Type	Storage Required
TINYINT	1 byte
SMALLINT	2 bytes
MEDIUMINT	3 bytes
INT, INTEGER	4 bytes
BIGINT	8 bytes
FLOAT(*p*)	4 bytes if 0 <= *p* <= 24, 8 bytes if 25 <= *p* <= 53
FLOAT	4 bytes
DOUBLE [PRECISION], REAL	8 bytes
DECIMAL(*M*, *D*), NUMERIC(*M*, *D*)	Varies; see following discussion
BIT(*M*)	approximately (*M*+7)/8 bytes

String Type Storage Requirements

In the following table, *M* represents the declared column length in characters for nonbinary string types and bytes for binary string types. *L* represents the actual length in bytes of a given string value.

Data Type	Storage Required
CHAR(*M*)	The compact family of InnoDB row formats optimize storage for variable-length character sets. See COMPACT Row Format Storage Characteristics. Otherwise, *M* × *w* bytes, <= *M* <= 255, where *w* is the number of bytes required for the maximum-length character in the character set.
BINARY(*M*)	*M* bytes, 0 <= *M* <= 255
VARCHAR(*M*), VARBINARY(*M*)	*L* + 1 bytes if column values require 0 − 255 bytes, *L* + 2 bytes if values may require more than 255 bytes
TINYBLOB, TINYTEXT	*L* + 1 bytes, where *L* < 2^8
BLOB, TEXT	*L* + 2 bytes, where *L* < 2^{16}
MEDIUMBLOB, MEDIUMTEXT	*L* + 3 bytes, where *L* < 2^{24}

MySQL Dokumentation