DAB2 – Datenbanken 2

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

Lektion 1: Einführung

Zürcher Fachhochschule

Inhalt der Vorlesung bis Ende Semester

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Einleitung und Anknüpfung an DAB1
- Physische Datenspeicherung und ungenutzer Speicherplatz
- Implementierungen der logischen Tabellen
- Implementierungen von Indizes
- Der Optimizer
- Grundlegende Zugriffspfade und Join-Arten und Subqueries
- Anfrageoptimierung durch unterschiedliche Formulierung der Abfragen
- Transaktionskonzept, Concurrency
- Nicht-Standard-Datenbanken, DWH, NoSQL, Big Data

Organisatorisches

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Sämtliche Unterlagen auf OLAT
- Modulvereinbarung und Leistungsnachweise siehe OLAT
- Benotung: 20% bewertete Praktika (2 Stück) und 80% Semesterendprüfung
- Bewertete Praktika:
  - Es gibt 2 Praktika mit Punkten, keine Note
- Prüfungsstoff:
  - Der Umfang des Prüfungsstoffs wird durch die Inhalte des Referenzfoliensatzes und der Praktika definiert (alle diese Inhalte sind Prüfungsstoff).
  - Zugelassene Hilfsmittel: Bei der Semesterendprüfung sind alle Unterlagen in Papierform zugelassen („open book“), aber keine sonstigen (wie z.B. elektronische Geräte).

3

Zürcher Fachhochschule

Ergänzende Literatur

Vorlesungsgrundlage (Ergänzung):

- Datenbanken: Implementierungstechniken.
  Gunter Saake, Kai-Uwe Sattler, Andreas Heuer.
  Die Vorlesung basiert auf diesem Buch (3. Auflage).
  Falls Sie Probleme mit den Folien haben, können Sie hier weitere Informationen finden.
- https://www.amazon.de/Datenbanken-Implementierungstechniken-Professional-Gunter-Saake/dp/3826691563

Weitere Bücher:

- Datenbanksysteme. Alfons Kemper, André Eickler.
  ISBN 978-3-486-72139-3. 9. Auflage, 2013
- … (es gibt zahllose Fachbücher und Artikel zum Thema Datenbanken)

4

Zürcher Fachhochschule

Gliederung

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

1 L	**Einführung**
4 L	**Datenorganisation Speicherung**
4 L	**Optimierung**
2 L	**Transaktionen, Recovery**
2 L	**Non-Standard Datenbanken**
1 L	**Repetition, Abschluss**

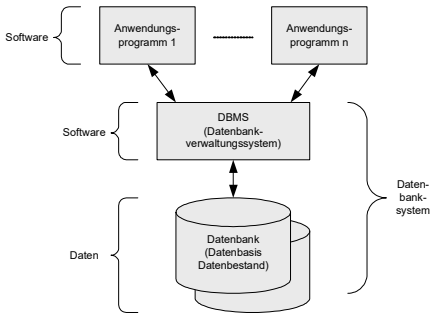
← "You are here"

5

Zürcher Fachhochschule

Rückblick: Begriffe

- Datenbanksystem (DBS)
  - DBMS (Datenbankverwaltungssystem)
  - Datenbank
    (Datenbasis, Datenbestand)
- Anwendungsprogramme (AP)
- DBS + AP = Informationssystem (IS)



6

Zürcher Fachhochschule

3-Ebenen-Architektur

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- 1975 von ANSI/SPARC (American National Standards Institute, Standards Planning And Requirements Committee) entwickelt
- Beschreibung einer Datenbank auf drei verschiedenen Ebenen (unterschiedliche Sichtweise):
  - Konzeptionell – Logische Gesamtsicht
  - Extern – Sicht einer einzelnen Anwendung / Benutzergruppe
  - Intern – Speicherung, Datenorganisation, Zugriffsstrukturen
- Realisiert durch ein DBMS
- Datenbeschreibungen ebenfalls in der DB gespeichert („Data Dictionary“)

7

Zürcher Fachhochschule

Drei verschiedenen Ebenen:

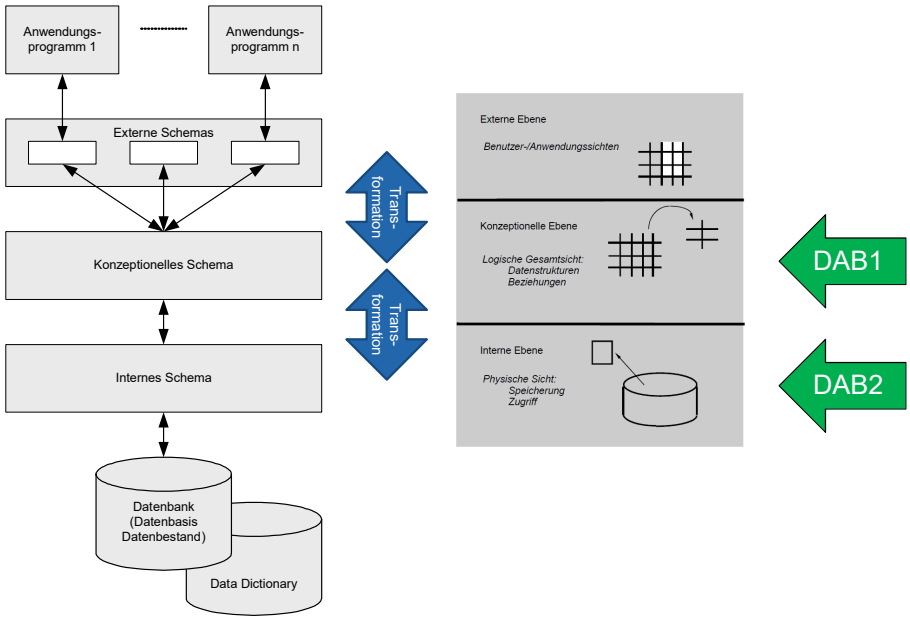
1. Konzeptionelle oder auch logische Ebene: wie sind die Daten strukturiert
2. Externe Ebene: wie sieht ein Benutzer / Programm die Daten
3. Interne oder auch physische Ebene: wie sind die Daten physisch gespeichert

DBMS realisieren / ermöglichen diese drei Ebenen. Dabei werden diese Datenbeschreibungen (Metadaten) der drei Ebenen ebenfalls in der DB gespeichert (im sogenannten „Data Dictionary“).

Die Abbildung zwischen den verschiedenen Ebenen (Schemas) erfolgt mittels Transformationen durch das DBMS.

3-Ebenen-Architektur

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie



8

Zürcher Fachhochschule

3-Ebenen-Architektur

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Zwischen den verschiedenen Ebenen (Schemas) erfolgen Transformationen

- Logische Datenunabhängigkeit:
  - Anwendungsprogramme sind (soweit möglich) immun gegen Änderungen in der logischen, konzeptionellen Ebene, sprich gegen strukturelle Änderungen am Datenbankschema (z.B. das Einfügen eines Attributs, das Löschen einer Tabelle).
- Physische Datenunabhängigkeit:
  - Anwendungsprogramme sind (soweit möglich) immun gegen Änderungen in der physischen, internen Ebene, sprich gegen Änderungen der Speicher- oder der Zugriffsstruktur (z.B. das Entfernen eines Indexes).

- Achtung: Nicht verwechseln mit 3-Phasen-DB-Entwurf!

3-Phasen-DB-Entwurf

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Das konzeptionelle Datenmodell (konzeptioneller Entwurf) ist eine weitgehend technologieunabhängige Spezifikation der Daten, die in der Datenbank gespeichert werden sollen → ERM, korrektes Diagramm (DAB1)
- Das logische Datenmodell (logischer Entwurf) ist eine Übersetzung des konzeptionellen Schemas in Strukturen, die mit einem konkreten DBMS implementiert werden können → RM, SQL (DAB1)
- Das physische Datenmodell (physischer Entwurf) umfasst alle Anpassungen, die nötig sind um eine befriedigende Leistung im Betrieb zu erreichen (Datenverteilung, Indexierung, …) → SQL, Tools (DAB2)

10

Zürcher Fachhochschule

In den beiden vorherigen Folien wurde in der 3-Schemen-Architektur nicht zwischen konzeptionellem und logischem Schema differenziert. Dies wird aber da und dort gemacht. Das konzeptionelle Schema meint dann das DBMS-unabhängige Schema, das logische Schema das DBMS-abhängige Schema (beide Schemata sind Teil der konzeptionellen, logischen Ebene).

In der Praxis sind auch andere Vorgehensmodelle mit anderen oder weiteren Vorgehensschritten verbreitet.

Physischer Entwurf

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Von grosser Bedeutung für die Praxis:
  - Sicherstellen der physischen Datenunabhängigkeit
  - Etablieren der geforderten Sicherheit
  - Gewährleisten einer genügenden Performanz
- In der Regel stark abhängig vom gewählten Produkt
- Typischerweise Aufgabe von Datenbankadministratoren und Datenbankentwicklern
- In der Lehre (auch in Lehrbüchern) oft wenig beachtet (warum?)

11

Zürcher Fachhochschule

Der physische Entwurf wird häufig nur kurz besprochen, da dieser komplex und abhängig vom jeweiligen Produkt ist. Die in dieser Vorlesung vermittelten Grundkenntnisse sind aber extrem wichtig, sobald Sie selbst eine Datenbank für eine Anwendung einsetzen möchten (z.B. wenn Sie mySQL in einem Web-Projekt einsetzen möchten). Mangelhafte Kenntnisse führen potentiell zu Datenverlusten, Performanceproblemen und hohen Aufwänden in der Wartung!

Die Abhängigkeit von bestimmten Produkten mag zunächst ärgerlich erscheinen (wir verwenden den SQL-Server). Die Konzepte der verschiedenen Produkte sind aber sehr ähnlich, so dass man, wenn man ein DBMS-Produkt gut kennt, sehr gut auf die Möglichkeiten und Rahmenbedingungen eines anderen Produktes schliessen kann.

Logischer Entwurf: Ergänzung

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Was bisher geschah (DAB1):
  - Logischer Entwurf mittels ERM/SQL
  - Definition von Tabellen
  - Festlegen einfacher Integritätsbedingungen
    - Schlüssel
    - Schlüssel ↔ Fremdschlüsselbeziehungen (referentielle Integrität)
    - Datentypen, NOT NULL
    - CHECK-Klausel
    - …
  - Einfache Abfragen
  - Was noch fehlt sind Mechanismen zur Formulierung von komplexeren Integritätsbedingungen, umfangreichen Geschäftsregeln, …
    → Datenbankprogrammierung (Trigger, Stored Procedures)

Trigger

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Programmcode, der vom DBMS ausgeführt wird.
- Auslösung der Abarbeitung dieses Programmcodes durch das DBMS aufgrund von sich ändernden Datenbankzuständen:
  - INSERT, UPDATE, DELETE
- Gut geeignet für:
  - Realisierung von Integritätsbedingungen
  - Berechnungen (z. B. das Nachführen von Summen)
  - Protokollieren von Datenänderungen
  - ... (vieles andere mehr!)
- Vorteile:
  - Entlastung der Anwendungsprogramme, effiziente Ausführung möglich, einfacher „rollout“.

13

Zürcher Fachhochschule

Trigger

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Prinzip: ECA
  - Event – Condition – Action
  - ON Ereignis IF Bedingung DO Aktion
- Können geschachtelt (auch kaskadierend) aufgerufen werden.
- Werden in SQL(-Dialekt) formuliert, haben einen Namen:
  - CREATE Trigger ...
  - DROP Trigger ...
  - ALTER Trigger ... {DISABLE ¦ ENABLE}
  - …
- Syntax proprietär (trotz SQL:1999-Standardisierung)!

Trigger

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Auslösung:
  - BEFORE vor der DML-Operation
  - AFTER nach der DML-Operation
  - INSTEAD OF anstelle der DML-Operation
- Wirkung:
  - for each statement – genau einmal → Statement Trigger
  - for each row – für jedes modifizierte Tupel → Row Trigger
- Geänderte Werte werden in Tabellen gespeichert:
  - `SELECT * FROM inserted` (liefert eingefügte und geändert-neu Datensätze)
  - `SELECT * FROM deleted` (liefert gelöschte und geändert-alt Datensätze)

15

Zürcher Fachhochschule

SQL Server unterstützt keine BEFORE Trigger, dies muss mittels INSTEAD OF Triggern ‘nachgebildet’ werden. Und der SQL-Server kennt nur "each statement"-Trigger.

Trigger – Beispiel

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

```
CREATE TRIGGER NewPODetail ON Purchasing.PurchaseOrderDetail
AFTER INSERT
AS
IF @@ROWCOUNT = 1
BEGIN
    UPDATE Purchasing.PurchaseOrderHeader
    SET SubTotal = SubTotal + LineTotal
    FROM inserted
    WHERE PurchaseOrderHeader.PurchaseOrderID = inserted.PurchaseOrderID
END
ELSE
BEGIN
    UPDATE Purchasing.PurchaseOrderHeader SET SubTotal = SubTotal +
    (SELECT SUM(LineTotal) FROM inserted
        WHERE PurchaseOrderHeader.PurchaseOrderID = inserted.PurchaseOrderID)
    WHERE PurchaseOrderHeader.PurchaseOrderID IN (SELECT PurchaseOrderID FROM inserted)
END;
```

Der Trigger wird ausgelöst, nachdem in der Tabelle PurchaseOrderDetail (KaufAuftragsPosition) ein Datensatz oder mehrere Datensätze eingefügt wurden. Es wird das Attribut SubTotal (Gesamtbetrag des Kaufauftrags) der Tabelle PurchaseOrderHeader (KaufAuftragKopf) um den, resp. die hinzugefügten Beträge (LineTotal) erhöht.

Trigger

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Einsatz von INSTEAD OF Triggern
  - Verhindern unerlaubter DML-Operationen
  - SQL-Constraints nicht ausreichend
  - Nicht änderbare Sichten ‚änderbar' machen
- Einsatz von AFTER Triggern
  - weitere Operationen sollen ausgelöst werden
  - Berechnungen ausführen und speichern
  - Historisierung

17

Zürcher Fachhochschule

Einsatz von INSTEAD OF Triggern

- Verhindern unerlaubter DML-Operationen: Verhindern von nicht erlaubten DML-Operationen und damit vermeiden von unnötiger Datenbank-Arbeit
- SQL-Constraints nicht ausreichend: wenn die Möglichkeiten SQL-Constraints zu definieren nicht ausreichen können mittels Triggern höchstkomplexe Überprüfungen ausgeführt werden.
- Nicht änderbaren Sichten ‚änderbar' machen: Änderungsoperationen auf Views sind bestimmten Einschränkungen unterlegen. Falls diese Einschränkungen greifen, d.h. die View nicht änderbar ist, ist es möglich, diese Änderungen durch Trigger an die Basistabellen weiterzugeben – und damit die Views 'änderbar' zu machen.

Einsatz von AFTER Triggern

- weitere Operationen sollen ausgelöst werden: auslösende DML-Operationen soll weitere Operationen auslösen (z.B. bestimmte andere Daten löschen)
- Berechnungen ausführen und speichern: z. B. das Nachführen von Summen oder von komplexen Berechnungs-Regeln
- Historisierung: Bestimmte oder alle Operationen an den Daten protokollieren (Logging, Auditing). Diese Historisierung kann nicht durch ein Anwendungsprogramm umgangen werden.
- Entlastung der Anwendungsprogramme: Datenbankintensive Operationen können direkt in der DB ausgeführt werden. Die Logik wird dann von der DB und nicht von den Anwendungsprogrammen durchgeführt
- Einfacher „rollout": Die Änderung wird sofort für alle Anwendungsprogramme aktiv

Trigger

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Einsatz sinnvoll, wenn:
  - Überprüfung/Funktion oft ausgeführt wird
  - SQL-Constraints nicht ausreichen
  - Logik von der DB und nicht von den Anwendungsprogrammen durchgeführt werden soll
  - Lösung stark vereinfacht wird
- Probleme:
  - Fehlerbehandlung
  - Testen, Debuggen
  - Gefahr der Unübersichtlichkeit
  - Ergebnis von Triggeroperationen evtl. abhängig von der Aufrufreihenfolge
  - Terminierung von geschachtelten Triggeraufrufen

18

Zürcher Fachhochschule

Insbesondere Trigger die Trigger auslösen (eventuell über mehrere Ebenen) sind sehr schwierig zu verstehen und zu warten. Es sollte daher darauf geachtet werden, Trigger möglichst nicht durch Trigger auszulösen.

Dieses Problem entspricht etwas der Problematik, wenn in Programmen Events andere Events auslösen. Auch dort ist dann die Fehlersuche sehr komplex.

Stored Procedures

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Menge von SQL-Anweisungen die unter einem gemeinsamen Namen gespeichert und vom DBMS als Einheit ausgeführt werden.
- Stored procedures werden in der Datenbank abgelegt und können von Anwendungsprogrammen und/oder Benutzern aufgerufen werden.
- Stored procedures bieten Programmierkonstrukte wie benutzerdefinierte Variablen, Rückgabewerte und Kontrollflussanweisungen. Die Programmiersprache ist der „prozedurale Kitt“ zwischen den nicht-prozeduralen (mengenorientierten) SQL-Anweisungen.
- Das Konzept entspricht in mancherlei Hinsicht den klassischen Prozeduren in höheren Programmiersprachen.

19

Zürcher Fachhochschule

- Kann Input-Parameter haben: Kann beliebig viele skalare Input-Parameter haben. Tabellen als Parameter müssen speziell definiert werden.
- Kann mehrere Tabellen als Rückgabewert haben: Gibt beliebig viele Tabellen als Resultat zurück.
- Die Function ist ein Spezialfall der Stored Procedure und hat nur einen skalaren Wert oder eine Tabelle als Rückgabewert.

Stored Procedure – Beispiel

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

```
CREATE PROCEDURE HumanResources.uspGetEmployees
    @LastName nvarchar(50),
    @FirstName nvarchar(50)
AS
    SELECT FirstName, LastName, Department
    FROM HumanResources.vEmployeeDepartmentHistory
    WHERE FirstName = @FirstName AND LastName = @LastName AND EndDate IS NULL;
```

- Erzeugt eine Stored Procedure (SP) Namens uspGetEmployees im Schema HumanRecources.
- @LastName und @FirstName sind Input-Parameter der SP.
- Die SP sucht den entsprechenden Mitarbeiter und gibt zusätzlich dessen Departement zurück. Der Mitarbeiter wird nur zurückgegeben, falls dieser in ungekündigtem Zustand ist (EndDate IS NULL).

20

Zürcher Fachhochschule

Stored Procedures

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Vorteile
  - Reduktion des Datenverkehrs.
  - Sehr komplexe Abfragen
  - Verwendbar für verschiedene Anwendungen (Kapselung von „business rules“).
  - Zugriffsoptimierung zur Erstellungszeit.
  - Sicherheit: nur Ausführungsrechte für stored procedure -> keine SQL-Injection möglich
  - Ermöglicht Parametrisierung von Abfragen.
  - Änder-/erweiterbar ohne die Anwendung anzupassen.
  - ...

21

Zürcher Fachhochschule

- Reduktion des Datenverkehrs zwischen Client und DBMS: Es können grosse Datenmengen direkt auf dem Server verarbeitet werden, ohne dass die Daten hin und her geschickt werden. Der SQL-Server ist unglaublich mächtig und kann gewaltige Mengen in kurzer Zeit verarbeiten.
- Realisierung sehr komplexer Abfragen möglich: SQL bietet eine breite Palette von Operationen, so dass fast alle gewünschten Aufgaben ausgeführt werden können.
- Zugriffsoptimierung zur Erstellungszeit: Das DBMS optimiert die Anfrage zum Zeitpunkt der Speicherung. Wird dynamisches SQL verwendet (d.h. das SQL-Statement wird zum Anfragezeitpunkt an den Server geschickt), muss vor der Ausführung der Operation die Optimierung ausgeführt werden (das Kostet Zeit).
- Sicherheit (nur Ausführungsrechte für Stored Procedure nötigt anstatt Schreib-/Leserechte auf Tabellen): Böswillige Datenbankzugriffe werden dadurch massiv erschwert.
- Parametrisierung möglich: Erhöht ebenfalls die Sicherheit und Performance und verhindert SQL-Injections.
- Änderbar und erweiterbar ohne Anwendungen anzupassen, da diese Methoden (Business Rules) zentral abgelegt sind (bei mehreren Umgebungen muss natürlich trotzdem das Deployment sichergestellt werden).
- Verwendbar für verschiedene Anwendungen: Unterschiedliche Anwendungen können die selbe Businesslogik verwenden.

Stored Procedures

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Nachteile
  - Syntax und Semantik sind nicht standardisiert, d.h. jeder DBMS-Hersteller hat seine eigene Sprache, z.B.:
    - Transact-SQL / T-SQL (Microsoft)
    - PL/SQL (Oracle)
    - ...
    - PSM/SQL (SQL3) hat sich nicht durchgesetzt

    -> Keine Portabilität
  - Verwaltungsaufwand (müssen analog wie Softwareobjekte – aber in Betriebsumgebung – verwaltet werden).
  - Fehlerbehandlung oft umständlich.
  - Oft keine „höheren“ Strukturierungsmöglichkeiten (z.B. interne Prozeduren, Module, «Klassen», ...).

22

Zürcher Fachhochschule

PSM/SQL (SQL / Persistant Stored Modules) ist eine ISO-Standard-Erweitererung für SQL, welche eine prozedurale Sprache für Stored Procedures definiert.

Stored Procedures vs. Triggers

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Stored procedures
  - Aufruf (explizit):
    - Durch Benutzer oder Anwendungsprogramm
    - Transaktionskontrolle
  - Einsatzbereiche:
    - Kapselung von „business rules“
    - Optimierung von Abfragen, Reduktion des Netzwerkverkehrs -> Batch Processing
    - Erhöhte Sicherheit benötigt
    - ...
  - Probleme:
    - Fehlerbehandlung
- Trigger
  - Aufruf (implizit):
    - Durch DBMS, in Abhängigkeit von Datenänderungen
    - Keine Transaktionskontrolle
  - Einsatzbereiche:
    - Konsistenzsicherung (referentielle Integrität)
    - Logging
    - Nachführen von Tabellen
    - ...
  - Probleme:
    - Kompliziert zu testen
    - Aufrufreihenfolge nicht determiniert

Stored Procedures, Triggers: DDL

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Trigger und Stored Procedures werden wie sonstige Datenbankobjekte mittels spezieller SQL-DDL-Anweisungen bearbeitet:
  - CREATE {TRIGGER|PROCEDURE|FUNCTION} Test ...
  - ALTER {TRIGGER|PROCEDURE|FUNCTION} Test ...
  - DROP {TRIGGER|PROCEDURE|FUNCTION} Test
- Trigger und Stored Procedures werden im Systemkatalog abgelegt.
- Syntax und Semantik hängen sehr stark vom verwendeten DBMS ab und sind somit nicht portabel!

Ausblick

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Das nächste Mal: Architektur von RDBMS
- Vorbereiten: Installation des Übungssystems gemäss separater Anweisung