Zusammenfassung Objekt- und dokumentzentrierte Informationssysteme

Philipp Jäcks

20. Januar 2016

Inhaltsverzeichnis

1 Einführung

- Einsatz hierarchischer DB ab 1970
- Relationale DB Einsatz ab 1980; hierarchische DB bleiben bestehen
- OO DB Einsatz ab Ende 80er; relationale DB bleiben bestehen
- XML DB Einsatz ab 2000; alle alten DB bleiben
- heute: hierarchische DB speichern meisten unternehmenskritischen Daten; gefolgt von Relationalen
- OO und XML DB aber in jedem guten relationalen System enthalten
- OO DBMS: ab Ende 80er; Startup Unternehmen; komplexes DB-Modell; später unvollkommener, inkonsistenter Standard
- Trend OO DBMS: instabil, bei read-write-Transaktionen wenig performant; unvollständig (Zugriffsrechte, Recovery, Sichten, Integritätsbedingungen); verschwinden wieder vom Markt
- Objektrelationale DB: Relationale DB integrieren OO Konzepte -> überholen OO DBMS

1.1 Entwicklung der Datenbankmodelle

Relationenmodell, RDBSs: Objekte dargestellt durch Zeilen in Tabellen

SET OF RECORD
A_1: Standard−Datentyp$_1$
...
A_n: Standard−Datentyp$_n$
END;

Objekttypen	Eigenschaft
Personen	*Name* (bestehend aus *Vor-* und *Nachname*) *Adresse* (bestehend aus *PLZ*, *Ort*, *Straße* und *Hausnummer*) *Hobbies* (bestehend aus einer Menge von *Hobbies*) *Geburtsdatum*

Tabelle 1: Beispiel: Modellierung von Personen

In RDBS:

- Eigenschaften *Name, Adresse* in Komponenten zerlegen
- Eigenschaft *Hobbies* auslagern (1NF)

Vorname	Nachname	PLZ	Ort	Straße	Hausnr	Gebdat

Vorname	Nachname	PLZ	Gebdat	Hobby

- Eigenschaften *Vorname, Nachname, PLZ* und *Geburtsdatum* sollen Schlüssel sein

Eigenschaften relationaler DB

- Starre Strukturen (Relationenschemata)
- Einfache Strukturen (nur Tabelle, 1NF)
- Für einfache Attributwerte (Zahlen, Zeichenketten, also Standard-Datentypen)
- Mit fester Semantik (Built-in Funktionen für Std-Typen)
- Austausch von Daten (etwa im Web) erschwert: Trennung Schema (Relationenschema) und Instanz (Relation), zum Versand muss beides zusammengefasst werden

Entwicklung: OO DB

- Forschung Ende 80er, Hype 90er => Nischenprodukt für neue Anwendungen; Ende 90er in RDBS
- Konzepte: komplexe Strukturen (statt nur Tabellen nun auch beliebig strukturierte Objekte)
  Komplexe Attributwerte (Typen können konstruiert werden)
  mit variabler Semantik (Methoden können def. werden)

Entwicklung: XML DB

- Forschung 90er, Hype Anfang 00er => Nischenprodukt für bestimmte Anwendungen; XML-Konzepte im objekrelationalen SQL-Standard
- Konzepte: Ausgangspunkt Dokumentbeschreibungssprache (Markup-Sprache) statt starrer Datenstruktur
  Variable Strukturen zur Beschreibung von Daten und Dokumente
  Komplexe Strukturen mit variabler Semantik mgl
  Austauschformat im Web zum Dokument- und Datenaustausch

Entwicklung: Digitale Bibliotheken (ab 2000)

- XML-Dokumente (oder andere Dokumentformate, pdf, doc... oft textlastig) langfristig speichern

- Auffindbar machen über strukturierte Metadaten (etwa in XML oder im relationalen DB-Modell)
- Spezielle Anforderungen: Dokumente haben Wert, können gekauft werden (E-Commerce)
  Dokumenten weltweit eindeutig identifiziert
  Identifier sind persistent, nicht flüchtig, obwohl Dokument vergriffen oder gesperrt sein kann
  Versionierung der Dokumente
- Suche nach Features in Texten (Stichworte,..) oder nach strukturierten Metadaten

Entwicklung: Multimedia DB (ab 95)

- Dokumente nicht nur textlastig, sondern Bild, Audio, Video, 2D/3D-Geoobjekt,...
- Problem: Features sind nicht nu Stichworte, sondern
  Bei Bildern: Farben und Farbverteilungen, erkannte Objekte wie Gesichter, Muster, Strukturen,..
  Bei Videos: Schnitte, Szenen, Bewegungen,...
  Bei Audio: speziell Musik, Melodien, Dynamik, Rhythmus,...
  Bei Geoobjekten: Enthaltensein, Schnittflächen,...

Darstellung komplexer Objekte: OODBS
In OO DBS sind nicht nur Std-Typen für Eigenschaften von Objekten erlaubt, sondern auch wdh Anwendung von Typkonstruktoren.

```
CLASS Personen
        TYPE TUPLE
                (Name: TUPLE
                        (Vorname: STRING,
                        Nachname: STRING),
                Adresse: TUPLE
                        (PLZ: INTEGER,
                        Ort: STRING,
                        Strasse: STRING,
                        Hausnummer: INTEGER),
                Hobbies: SET (Hobby: STRING),
                Geburtsdatum: DATE)
```

Objektidentität

- RDBS: Schlüsselwerte können sich ändern, Identität eines Objektes geht evtl verloren
- OODBS:
  - Objekte ex unabhängig von Werten ihrer Eigenschaften, d.h. Identität bleibt gleich, während sich Eigenschaften ändern

- in technischen Anwendungen: teilweise Objekte nicht durch äußere Eigenschaften unterscheidbar (Bsp: Menge von Schrauben gleicher Art)
- Entscheidung evtl durch Position, aber nicht durch Namen

Ist-Hierarchie (Vererbung)

- Fehlt in RDBS; quasi nur über viele Fremdschlüssel simulierbar
- OODBS:
  - Objekttypen in Vererbungshierarchie mgl

```
CLASS Studenten INHERITS Personen
    TYPE TUPLE
        (Matrikelnummer: INTEGER,
        Studienfach: STRING,
        Vater: Personen,
        Mutter: Personen,
        ...)
```

  - Vererbung (Studenten sind spezielle Personen) und Komponentenobjekte (Vater und Mutter sind Personen)

Methoden statt Host-Prozeduren

- RDBS: spezielle Prozeduren und Funktionen von *außen* aufgesetzt
  SQL Ausdruck oder sogar Programm in höheren Programmierspraceh
  Bsp: Alter einer Person aus Geburtsdatum: Sicht in SQL auf Basistabelle oder C-Programm mit eingebetteter SQL-Anfrage
- OODBS: neben Eigenschaften auch die mit ihnen durchführbaren Methoden in die Objekttyp-Definition einkapseln und vererben
  Bsp: Alter ist in Definition erklärt (Interface, getrennt davon Impl)

Dokumente

- Text- oder große Multimediadokumente: groß, unstrukturiert/maximal semistrukturiert, variabel strukturiert (Text nicht immer starr in Kapitel/Abschnitt/...)
- in RDBS:
  - als CLOB oder BLOB (völlig unstrukturiert), Metadaten extrahieren in relationale Tabelle (Autor, Titel, Format, Länge...)
  - oder *schreddern*: Dokument in kleinste Anteile zerlegen und in relationaler Tabelle speichern mit folgenden Problemen
    Relation sieht starre Struktur vor
    Relation muss Ordnung der kleinste Anteile bewahren bei der Rekonstruktion des gesamten Dokuments

- in XML DB:
  - von unstrukturiert bis voll strukturiert (auch variabel)
  - Markup-Sprache für Dokumente geeignet
  - evtl stark strukturierte Anteile in Relationen gespeichert -> Side Tables

1.2 Nachteile relationaler DB

Komponenten DB-Modell

- **Strukturell:** Datenstrukturen für Anwendungsobjekte, Konzepte, Modellierung von Beziehungen zw Anwendungsobjekten, Integritätsbedingungen
  im Relationenmodell: Relationen (Tabellen) für alles, Schlüssel, Fremdschlüssel
- **Operationenteil:** Generische Operationen auf Datenstrukturen und Beziehungen
  im Relationenmodell: Relationenalgebra, SQL-Anfragen und SQL-Updates
- **Höhere Konzepte:** Metainformationen und objektspezifische Operationen,..
  im Relationenmodell: höchstens Data Dictionary, sonst nichts

Vorteile im Relationenmodell

- **Strukturteil:** einfache, einheitliche Beschreibung der Anwendungsdaten
  Exaktes, mathematisches Fundament
- **Operationeteil:**
  - Deskriptivität: was, nicht wie; mengenorientiert
  - Abgeschlossenheit: Ergebnis ist wieder eine Relation
  - Adäquatheit: alle Konzepte des Strukturteils unterstützt
  - Optimierbarkeit: System kann selbst schnellere Auswertungsreihenfolge finden
  - effiziente Impl: jede Operation der Relationenalgebra effizient implementierbar
  - Sicherheit: jede syntaktisch richtige Anfrage liefert Ergebnis
  - Orthogonalität: Alle Operationen beliebig miteinander kombinierbar

Nachteile Datenmodellierung

- Komplexe Attribute: Wertemengen oder mehrere Komponenten nur über Fremdschlüssel simulierbar
- Beziehungen: immer über Fremdschlüssel dargestellt

Nachteile Datenbankentwurf

- Methoden:

  - Informale Methoden: Entity-Relationship Modell abbilden in Relationenschemata
    Schwach bei komplexen Attributen (simuliert über n:m-Beziehungen)
  - Formale Algorithmen: Attribute und Abhängigkeiten bestimmen Relationenschemata
    Normalformen, Abhängigkeitstreue, ...
    Entwurf ohne Semantik, Abhängigkeiten reichen nicht zur Anwendungsbeschreibung
  - Allgemeine Schwächen: Ergebnis verliert mühsam erfasste Semantik

- mangelnde Semantik: Beschreibung der Semantik mit Abhängigkeiten zw Attributen (funktional[1], mehrwertig)
  Reale Bsp komplizierter: CIM-Datenbank, Ventilfeder, Anlasserzahnkranz,.. (was bestimmt sich funktional oder mehrwertig)
  vernachlässigt: Verbund- und Inklusionsabhängigkeiten

Nachteile Anfragesprache
Strukturmangel im Ergebnis
Beispiel: Dreifacher Verbund zur Rekonstruktion EINES Buches

Nachteile Anfrageoperationen
Anfragen an komplexe Attribute

- keine Unterstützung komplexer Strukturen in Anfrageformulierung
- Notwendigkeit expliziter Verbundoperationen

Nachteile Update-Operationen
Identifikation der Objekte über sichtbare Schlüssel -> Kein Unterschied zw Umzug, Kauf neues Auto
OODBS: Objekte eindeutig identifizierbar -> Unterschied zw Umzug und Autokauf

[1] Attribute bestimmen eindeutig den Wert anderer Attribute, dann spricht man von funktionaler Abhängigkeit

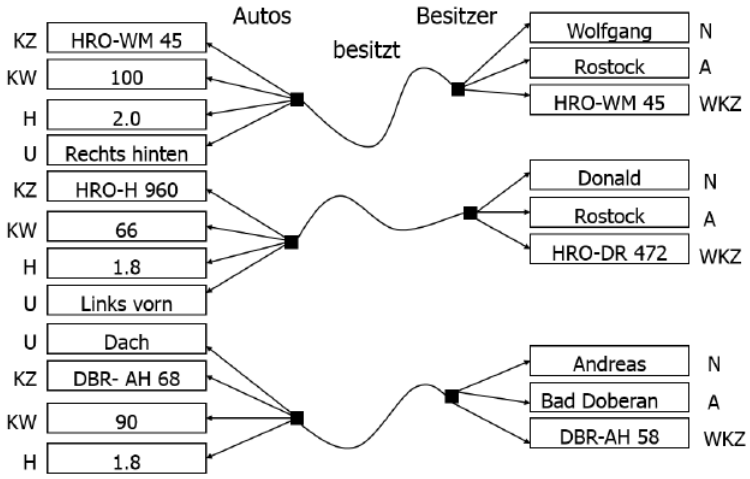

Abbildung 1: Update Operation in OODBS