

Vorlesung 2

UML Klassendiagramme

→ Entspricht Beziehungen

→ Optional:

- Assoziationsnamen
- Liniendichtung (\leftarrow oder \rightarrow), sonst bidirectional
- Rollennamen
- Kardinalitätsrestriktionen

Kardinalitätsrestriktionen:

$*..*$ \rightarrow mindestens \mathbb{N} , maximal \mathbb{N}

$0..*$ \rightarrow 0 oder mehrere

$1..*$ \rightarrow 1 oder mehrere

$0..1$ \rightarrow 0 kann keine oder eine geben

1 \rightarrow genau 1

UML Assoziationsklassen

• Für Beziehungen mit eigener Attributen ist eine Assoziationsklasse notwendig

→ wird mit einer gestrichelten Linie zu einer Klasse dargestellt

→ Name der Assoziationsklasse entspricht der Namen der Assoziation

UML Teil-vor Beziehungen

- Aggregation
 - zwischen Subkomponente und Superkomponente
- Komposition
 - Teil-Objekt gehört genau zu einem Aggregatobjekt
- Reflexive Assoziation

UML Vererbung

- alle Instanzen der Subklasse sind auch Instanzen der Superklasse
- Vererbung vor Eigenschaften der Superklasse an alle Subklassen
- Wiederverwendbarkeit, Erweiterbarkeit
- Keine Wiederholung

Konzeptueller Schema

⇒ Relationales Datenmodell

- Probleme für 1:1 Transformation der Klassen in Tabellen
 - Zu viele Tabellen
 - Zu viele Joins
 - Fehlende Tabellen
 - Falsche Modellierung der Vererbung
 - Denormalisierung der Daten

Transformation der Klassen in Tabellen

- Name der Tabelle = Plural vor dem Klassennamen
- Alle zusammengesetzte Attribute werden als neue Tabelle modelliert
- Abgeleitete Attribute werden nicht in Tabellen gespeichert
- Ersatzschlüssel - Schlüssel, die nicht aus der Domäne des Problems kommen (z.B. ID1)
- Wenn möglich kann man automatisch generierte Schlüssel benutzen

Transformation der Assoziations Beziehungen

- 1: 0, 1
 - man erstellt eine Tabelle für jede Klasse aus der Assoziation
 - der Primärschlüssel der „0..1“ Tabelle verweist auf der Schlüssel der „1“ Tabelle
- 1: 1
 - Meistens erstellt man eine einzige Tabelle welche die Attribute beider Klassen enthält
- 1: M (1:1..*)
 - man erstellt eine Tabelle für jede Klasse aus der Assoziation
 - der Schlüssel der „1“ ist ein Fremdschlüssel in der „M“ Tabelle

- M:N (1..*:1..*)
 - man erstellt eine Tabelle für jede Klasse der Assoziation
 - man erstellt eine zusätzliche Tabelle
 - die Primärschlüssel der ursprünglicher Tabellen werden als Fremdschlüssel in der Cross Table definiert
 - der Primärschlüssel in der Cross Table wird meistens als Zusammensetzung des Fremdschlüssel definiert
 - Wenn die Assoziation, eine Assoziationsklasse hat, werden alle Attribute der Assoziationsklasse in der Cross Table eingefügt
 - meistens besteht der Name der Cross Table aus den Namen der Tabellen deren Beziehungen diese modelliert

Transformation der Vererbung

I Methode

- man erstellt eine Tabelle für jede Klasse, ein Sicht (View) für jedes Paar Superklasse - Subklasse
- flexibel
- diese Methode erstellt die meiste Tabellen und leichter
- Effizienzprobleme → jede Abfrage braucht ein Join

II Methode

- Man erstellt eine einzige Tabelle (für die Superklasse) und man erstellt zusätzliche Attribute für die Subklassen
- Optional kann man eine Tabelle von Subklassen definieren und leichter für jede Subklasse
- erstellt am wenigsten Tabellen
- gute Effizienz
- eine neue Subklasse ⇒ man muss die Struktur der Datenbank ändern
- ein Tupel hat eine größere Länge

III Method

- man erstellt eine Tabelle für jede Subklasse und die Attribute der Superklasse in jeder Subklasse-Tabelle eingefügt
- ziemlich gute Effizienz
- eine neue Subklasse verursacht keine Strukturänderungen
- Änderungen in der Struktur der Superklasse verursachen Änderungen in der Struktur aller Tabellen

Welche Methode ist besser?

- wenn die Anzahl der Tupel in der Tabellen klein ist, dann kann man die 1. Methode wählen
- wenn die Superklasse wenige Attribute hat im Vergleich zu den Subklassen ⇒ 3. Methode
- wenn die Subklassen wenige Tupel haben, dann wählt man die 2. Methode

Transformation der Aggregation und Komposition

- ähnlich wie Assoziation
- Komposition wird oft in derselber Tabelle modelliert, da diese 1:1 Beziehungen haben
- Wenn Komposition in mehreren Tabellen modelliert wird, dann muss unbedingt ON DELETE CASCADE implementiert werden
- eine feste Anzahl von „Teile“ aus einem „Ganzen“ => genau dieselbe Anzahl von Fremdschlüssel in der Tabelle „Ganzen“

Transformation der Reflexiven Assoziation

- man fügt ein Fremdschlüssel ein, der auf dieselbe Tabelle verweist
- man kann nicht ON DELETE CASCADE benutzen