Geographic Information Technology Training Alliance (GITTA) presents:

Das relationale Datenmodell

Responsable: Anca Dobre, Susanne Bleisch

Table des matières

1. Das relationale Datenmodell

Das Relationenmodell dient den meisten derzeitigen Datenbanken als Grundlage. In komerziellen Datenbanken wird das Relationenmodell seit etwa 1981 eingesetzt. Es wurde von E.F. Codd um 1970 vorgestellt mit dem Ziel die Datenunabhängigkeit zu gewährleisten und basiert auf einer Variante des mathematischen Konzepts der Relation in der Relationen auf einfache Weise als Tabellen interpretiert werden.
Der Fokus dieser Lektion liegt in der Umwandlung eines konzeptionellen Schemas, wie zum Beispiel dem Entity-Relationship-Schema (Gegenstands-Beziehungs-Schema), in das logische Schema, dem relationalen Datenbankmodell. Erklärungen zu den verschiedenen Schema-Typen finden sie in der Unit **Datenmodelle, Schemata und Instanzen**.

1.1. Konzepte des Relationenmodells

Im Gegensatz zum Entity-Relationship-Modell (oder Gegenstands-Beziehungs-Modell), welches ein konzeptuelles Modell darstellt, handelt es sich beim Relationenmodell um ein logisches Datenmodell. Dieses Modell liegt in gewisser Hinsicht eine Stufe "tiefer". Es werden hier keine abstrakten Gegenstände oder Gegenstandstypen mehr betrachtet, sondern nur noch deren konkrete, datenmässige Umsetzung. Das Ziel der logischen Datenmodellierung ist das Anordnen der zu speichernden Informationen mit Hilfe eines Modells, welches eine möglichst redundanzfreie Speicherung unterstützt und geeignete Operationen für die Datenmanipulation zur Verfügung stellt

1.1.1. Organisation der Daten im relationen Datenbankmodell

In der Regel basiert die logische Datenmodellierung auf einem fertiggestellten konzeptuellen Schema, das mit Hilfe bestimmter Richtlinien und Regeln in möglichst eindeutiger Weise auf ein relationales Schema abgebildet wird. Die grundlegende Organisationsform der Daten im relationalen Datenbankmodell ist die Relation. Eine Relation kann als Tabelle dargestellt werden, wobei zu beachten ist, dass sich die Definition einer Relation nicht mit der einer Tabelle deckt und umgekehrt
Es sind im wesentlichen zwei Gründe, die für das Relationenmodell als logisches Datenbankmodell sprechen:

- Einfachheit: die gesamte Information einer relationalen Datenbank wird einheitlich durch Werte repräsentiert, die mittels eines einzigen Konstrukts (nämlich der "Relation") strukturiert sind
- Systematik: das Modell besitzt eine fundierte mathematische Grundlage - die Mengentheorie

Im Folgenden werden die grundlegenden Begriffe des relationalen Datenbankmodells eingeführt.

1.1.2. Definitionen

Domäne:

Eine Domäne besteht aus einem Namen D und einer Menge atomarer Werte. Ein anderer Name für Domäne ist Wertebereich. Domänen definieren den Wertebereich von Attributen.

Beispiel Domäne

Tupel:

Ein Tupel **t** ist eine Liste mit n Werten **t = <d1, d2, ..., dn>**, wobei jeder Wert di ein Element der Domäne **Di**, oder NULL sein muss.

Beispiel Tupel

Attribut:

Ein Attribut **A** bezeichnet die Funktion, die eine Domäne D in einem Relationenschema R ausübt. Es kann auch als Abbildung der Tupel einer Relation auf den Wert des jeweiligen Tupels (für dieses Attribut) verstanden werden., wobei jeder Wert di ein Element der Domäne, oder NULL sein muss.

Beispiel Attribut

Relationenschema:

Ein Relationenschema **R**, Schreibweise: R(A1, A2, ..., An), bezeichnet eine Menge von Attributen {A1, A2, ..., An}.

Beispiel Relationenschema

Relation:

Eine Relation **r** ist eine Instanz (Ausprägung) des Relationenschemas R(A1, A2, ..., An). Sie ist eine Teilmenge des kartesischen Produkts (Kreuzprodukt) der beteiligten Domänen.

Beispiel Relation

Relationales Datenbankschema:

> Ein relationales Datenbankschema ist eine Menge von Relationenschemata S = {R1, ..., Rn} zusammen mit einer Menge von Integritätsbedingungen. Eine relationale Datenbankinstanz ist die Menge {r1, ..., rn} wobei ri Instanz von Ri ist und alle Integritätsbedingungen erfüllt sind. Eine relationale Datenbank ist ein relationales Datenbankschema mit einer entsprechenden Datenbankinstanz.

Begriffe des relationalen Datenschemas

Das relationale Schema eines Gegenstandes (Entität) kann als Tabelle (Relation) abgebildet werden. In diesem Beispiel sind die Entität die Studierendennoten. Diese Entität wird mit den Attributen Name, Fach und Note beschrieben. Die Domäne oder Wertebereich der Attribute Name und Fach ist alle Gross- und Kleinbuchstaben des Alphabets, die des Attributs Note sind Zahlen von 1 bis 6 mit einer Kommastelle. Diese Struktur der Entität mit ihren Attributen, ohne den eigentlich Inhalt, nennt man Relationenschema. Wenn nun Inhalte in die Relation eingefügt werden, dürfen nur Werte verwendet werden, die in der Domäne definiert sind. Ein Tupel sind die zusammengehörenden Werte verschiedener Attribute. Es entspricht in der Tabelle einer Zeile.
Zwischenfrage: In der obigen Tabelle hat sich ein kleiner Fehler eingeschlichen. Können Sie ihn entdecken?
Falls Sie die Antwort zu dieser Frage diskutieren möchten, steht Ihnen dazu das Diskussionsforumstopic "Relationales Datenmodell" zur Verfügung.

1.2. Abbilden eines ER-Schemas auf ein relationales Datenbankschema

In dieser Unit werden nun die Regeln eingeführt, mit deren Hilfe die Konstrukte des Entity-Relationship-Modells auf Konstrukte des relationalen Modells abgebildet werden können. Danach sind wir in der Lage, jedes beliebige Entity-Relationship-Modell in ein relationales Datenbankschema zu überführen.
Jeder der folgenden Abbildungsregeln ist einem Konstrukt des Entity-Relationship-Modells gewidmet. Um ein Entity-Relationship-Modell in ein relationlaes Datenbankschema umzuwandeln, müssen alle 8 Abbildungsregeln abgearbeitet werden. Jede Regel wird für alle im zu bearbeitenden Entity-Relationship-Modell gefundenen und der Regel entsprechenden Gegenstandstypen (Regel 1,2,7 und 8) oder Beziehungen (Regel 3,4,5 und 6) angewendet. Man beachte die korrekte Reihenfolge der Anwendung der Abbildungsregeln in der Praxis (1,7,8,2,3,4,5 und zum Schluss 6).
Zu jeder Abbildungsregel wird die Definition angegeben und ein Anwendungsbeispiel gezeigt.

1.2.1. Repetition: ER-Konzepte

Erweitertes Entity-Relationship Diagramm
Um die Abbildungsregeln zu verstehen, müssen Sie die Konzepte des ER-Modells kennen. Hier folgt ein Flash-Beispiel, anhand dem Sie ihre Kenntnisse auffrischen können.
Falls Sie die Konzepte des ER-Modells nicht kennen, schauen Sie im folgenden Buch nach:

- Fundamentals of Database Systems, Elmasri, Ramez; Navathe, Shamakant B. (1994)

Sie sollten folgende Konstrukte erkennen: starker Gegenstandstyp, schwacher Gegenstandstyp, Beziehungstyp, identifizierender Beziehungstyp, Eigenschaft, abgeleitete Eigenschaft, mehrwertige Eigenschaft, zusammengesetzte Eigenschaft, Partialschlüsseleigenschaft und Schlüsseleigenschaft.

Seulement les images peuvent être représentées dans cette version! Flash, animation, son etc. sont visibles uniquement dans la version en ligne. [link]

1.2.2. Regel 1

In diesem ersten Schritt werden alle starken Gegenstandstypen ins relationale Datenbankschema umgewandelt. Bei Subklassen benutzen sie Regel 8.

Definition Regel 1
Definiere ein Relationenschema R für jeden starken Gegenstandstypen G, wobei die Eigenschaften von G die Attribute von R bilden. Bei mehrwertigen Eigenschaften verfahre nach Regel 7. Wähle einen Primärschlüssel (Identifikationsschlüssel) aus.
In diesem Beispiel sehen Sie die Anwendung der Regel 1:

Seulement les images peuvent être représentées dans cette version! Flash, animation, son etc. sont visibles uniquement dans la version en ligne. [link]

Zuerst wird der starke Gegenstandstyp Kunde ausgewählt und eine Relation dafür erstellt.
Kunde
Dann werden alle Eigenschaften von Kunde, in diesem Fall KundNr, Vorname, etc zu Attributen dieser Relation.
Kunde(KundNr., Vorname, Nachname, Strasse, StrNr)
Um ein einzelnes Tupel in dieser Relation ansprechen zu können, wird ein Primärschlüssel (mehr Informationen siehe Unit Datenintegrität) ausgewählt. In diesem Fall ist das Attribut KundNr als Primärschlüssel verwendbar.
Kunde(KundNr, Vorname, Nachname, Strasse, StrNr)

1.2.3. Regel 2

In diesem Schritt werden alle schwachen Gegenstandstypen ins relationale Datenbankschema umgewandelt.

Definition Regel 2
Für jeden schwachen Gegenstandstypen S mit Eigentümer G erzeuge ein Relationenschema R, wobei die Eigenschaften von S die Attribute von R bilden. Bei mehrwertigen Eigenschaften verfahre nach Regle 7. Übernimm den Primärschlüssel des Relationenschemas, das dem Eigentümer G entspricht und füge ihn als Fremdschlüssel R hinzu. Wähle eine Attributkombination (Partialschlüssel) aus, die dann zusammen mit diesem Fremdschlüssel den Primärschlüssel der Relation bildet. Erst die Kombination von Fremdschlüssel- und Partialschlüsselattributen bildet den Primärschlüssel von R.
In diesem Beispiel sehen Sie die Anwendung der Regel 2:

Seulement les images peuvent être représentées dans cette version! Flash, animation, son etc. sont visibles uniquement dans la version en ligne. [link]

Zuerst wird der schwache Gegenstandstyp Teil ausgewählt und eine Relation dafür erstellt.
Teil
Dann werden alle Eigenschaften von Teil, in diesem Fall Name und Redakteur zu Attributen dieser Relation.
Teil(Name, Redakteur)
Dann wird der Primärschlüssel des Relationenschemas, das dem Eigentümer entspricht als Fremdschlüssel hinzugefügt (Eigentümer ist hier die Relation Zeitung(Name, Auflage, Preis)).
Teil(ZeitungName, Name, Redakteur)
Das Attribut Name bildet zusammen mit dem Fremdschlüssel den Primärschlüssel der Relation.
Teil(ZeitungName, Name, Redakteur)

1.2.4. Regel 3

Hier werden nun alle binäre Beziehungstypen der Art (1,1)(1,1), (0,1)(1,1) oder (0,1)(0,1) ins relationale Datenbankschema umgewandelt.

Definition Regel 3
Suche alle regulären, binären (1,1)(1,1)-, (0,1)(1,1)- und (0,1)(0,1)-Beziehungstypen B. Finde die Relationenschemata S und T für die beteiligten Gegenstandstypen. Wähle eines davon (zum Beispiel S) aus und füge dort den Primärschlüssel von T als Fremdschlüssel sowie die Eigenschaften von B als Attribute hinzu.
In diesem Beispiel sehen Sie die Anwendung der Regel 3:

Seulement les images peuvent être représentées dans cette version! Flash, animation, son etc. sont visibles uniquement dans la version en ligne. [link]

Zuerst wird ein Beziehungstyp der Art (1,1)(1,1)-, (0,1)(1,1)- und (0,1)(0,1)ausgewählt. Bei uns der Beziehungstyp "leitet".
Dann wählen wir einen an der Beziehung "leitet* beteiligten Gegenstandstypen aus.
Zeitung(Name, Auflage, Preis)
Den Primärschlüssel des zweiten Gegenstandstyps (Chefredaktor) fügen wir als Fremdschlüssel in die Relation Zeitung ein.
Zeitung(Name, Auflage, Preis, *Chefredaktor_PersNr)*
Am Ende fügen wir die Eigenschaften der Beziehung "leitet" als Attribute der Relation Zeitung hinzu.
Zeitung(Name, Auflage, Preis, *Chefredaktor_PersNr*, SeitDatum)

1.2.5. Regel 4

Nun werden alle binäre Beziehungstypen der Art (1,n)(1,1), (0,n) (1,1), (1,n)(0,1) oder (0,n)(0,1) ins relationale Datenbankschema umgewandelt.

Definition Regel 4
Suche alle regulären, binären (1,n)(1,1)-, (0,n)(1,1)-, (1,n)(0,1)- und (0,n)(0,1)-Beziehungstypen B, sowie die jeweiligen Relationenschemata S und T der beteiligten Gegenstandstypen. Wähle das Relationenschema auf der "(1,1)"/"(0,1)"-Seite (S) aus und füge dort den Primärschlüssel von T als Fremdschlüssel hinzu. Ausserdem werden allfällige Eigenschaften von B als Attribute zu S hinzugefügt.
In diesem Beispiel sehen Sie die Anwendung der Regel 4:

Seulement les images peuvent être représentées dans cette version! Flash, animation, son etc. sont visibles uniquement dans la version en ligne. [link]

Zuerst wird ein Beziehungstyp der Art (1,n)(1,1)-, (0,n)(1,1)-, (1,n)(0,1)- und (0,n)(0,1) ausgewählt. Bei uns der Beziehungstyp "gibt in Auftrag".
Dann wählen wir den auf der "(1,1) oder (0,1)"-Seite liegenden Gegenstandstypen aus. Bei uns ist dies die Relation Inserat.
Inserat(AuftragsNr, Grösse, Preis, ErscheinDat)
Den Primärschlüssel des zweiten Gegenstandstyps (Kunde) fügen wir als Fremdschlüssel in die Relation Inserat ein.
Inserat(AuftragsNr, Grösse, Preis, ErscheinDat, *AuftraggeberKundNr*)

1.2.6. Regel 5

Hier werden alle binäre Beziehungstypen der Art (0,n)(0,n), (1,n) (0,n) oder (1,n)(1,n) ins relationale Datenbankschema umgewandelt.

Definition Regel 5
Suche alle regulären, binären (0,n)(0,n)-, (0,n)(1,n)- und (1,n)(1,n)-Beziehungstypen B, sowie jeweils die Relationenschemata S und T der beteiligten Gegenstandstypen. Definiere für jeden Beziehungstyp B ein neues Relationenschema R. Die Primärschlüssel der Relationenschemata der beteiligten Gegenstandstypen S und T werden als Fremdschlüssel übernommen. Sie bilden zusammen den Primärschlüssel des neuen Relationenschemas R. Füge Attribute, die Eigenschaften von B entsprechen, zu R hinzu.
In diesem Beispiel sehen Sie die Anwendung der Regel 5:

Seulement les images peuvent être représentées dans cette version! Flash, animation, son etc. sont visibles uniquement dans la version en ligne. [link]

Zuerst wird ein Beziehungstyp der Art (0,n)(0,n)-, (0,n)(1,n)- und (1,n)(1,n) ausgewählt. Bei uns der Beziehungstyp "erscheint in".
Dann definieren wir für den Beziehungstyp "erscheint in" ein neues Relationenschema.
InseratInZeitung
Die Primärschlüssel der Relationenschemata der beteiligten Gegenstandstypen Inserat und Zeitung werden als Fremdschlüssel übernommen.
InseratInZeitung (Inserat_AuftragsNr, Zeitung_Name)
Zum Schluss fügen wir Attribute, die Eigenschaften der Beziehung "erscheint in" entsprechen, zur Relation InseratInZeitung hinzu.
InseratInZeitung (Inserat_AuftragsNr, Zeitung_Name, Position)

1.2.7. Regel 6

Hier werden alle n-stelligen Beziehungstypen ins relationale Datenbankschema umgewandelt. Dies erfolgt analog zu Regel 5.

Definition Regel 6
Verfahre für n-stellige Beziehungstypen (n>2) analog zu Regel 5, d.h. bilde sie auf eigenständige Relationenschemas ab und übernimm den Primärschlüssel der Relationenschemas aller beteiligten Gegenstandstypen als Fremdschlüssel.
Siehe Beispiel von Regel 5.

1.2.8. Regel 7

Wenn man in Regel 1 auf mehrwertige Eigenschaften gestossen ist, wandelt man diese gemäss dieser Regel ins relationale Datenbankschema um.

Definition Regel 7
Definiere für jede mehrwertige Eigenschaft E ein neues Relationenschema R'. R' enthält ein Attribut A, das der Eigenschaft E entspricht und den Primärschlüssel K jenes Relationenschemas (R), welches dem Gegenstandstyp entspricht, der E enthält. Der Primärschlüssel von R' ergibt sich aus der Kombination von A und K (man beachte, dass R keine Attributentsprechung für die mengenwertige Eigenschaft besitzt).
In diesem Beispiel sehen Sie die Anwendung der Regel 7:

Seulement les images peuvent être représentées dans cette version! Flash, animation, son etc. sont visibles uniquement dans la version en ligne. [link]

Zuerst definieren wir für die mehrwertige Eigenschaft TelNr eine neue Relation ChefTelNr mit dem Attribut TelNr)
ChefTelNr(TelNr)
Der Primärschlüssel der Relation, an der die mehrwertige Eigenschaft TelNr hängt (bei uns Chefredakteur), wird in die Relation ChefTelNr als Fremdschlüssel übertragen.
ChefTelNr(*Chefredakteur_PersNr*, TelNr)
Zusammen mit dem Attribut TelNr bildet das Attribut Chefredakteur_PersNr den Primärschlüssel.
ChefTelNr(Chefredakteur_PersNr, TerlNr)

1.2.9. Regel 8

Wenn man in Regel 1 auf Subklassen gestossen ist, wandelt man diese gemäss dieser Regel ins relationale Datenbankschema um.

Definition Regel 8
Definiere ein Relationenschema R für die Superklasse C mit den Attributen A(R) = (K, A1, ..., An). Bestimme K zum Primärschlüssel von R. Definiere weiter ein Relationenschema Ri für jede Subklasse Si, (1<=i<=m) mit den Attributen A(Ri) = (K) vereinigt (Attribute von Si). Setze den Primärschlüssel von Si gleich K.
In diesem Beispiel sehen Sie die Anwendung der Regel 8:

Seulement les images peuvent être représentées dans cette version! Flash, animation, son etc. sont visibles uniquement dans la version en ligne. [link]

Die Relation Angestellter(ANummer) ist die Superklasse, Techniker und Ingenieur(Ausbildung) sind die Subklassen in diesem Beispiel.
Angestellter(ANummer)
Techniker
Ingenieur(Ausbildung)
Den beiden Gegenstandstypen der Subklasse fügen wir den Primärschlüssel der Superklasse hinzu.
Angestellter(ANummer)
TechnikerANummer)
Ingenieur(ANummer, Ausbildung)

1.2.10. Anwendung der Abbildungsregeln

Nachstehende Interaktion ermöglicht das Üben der acht Abbildungsregeln. In einem ersten Schritt sollen jeweils die zur aktuellen Regel passenden Gegenstandstypen selektiert werden. Danach muss das entsprechende relationale Schema für diese Gegenstandstypen gewählt werden.
Um die Flash-Animation zu starten, klicken Sie bitte auf die Grafik.

Interaktion: Anwendung Abbildungsregeln

1.2.11. Umwandlung konzeptionelles Schema -> logisches Schema

Wandeln Sie das nachstehende konzeptionelle Schema in ein relationales Datenbankschema um, indem Sie die in dieser Unit eingeführten Regeln anwenden. Um die Darstellung zu vereinheitlichen unterstreichen Sie bitte Identifikationsschlüssel und stellen Sie Fremdschlüssel kursiv dar.
Dokumentieren Sie Ihre Lösung in einer Word- oder PDF-Datei und veröffentlichen Sie diese auf dem Diskussionsforum unter dem Thema 'Übung Umwandlung'. Schauen Sie sich die Lösungen Ihrer Mitstudierenden an und kommentieren Sie diese gegebenenfalls. Falls Sie Probleme oder Fragen haben, können Sie diese ebenfalls im Diskussionsforum stellen und diskutieren. Die veröffentlichten Lösungen werden von einem Tutor angeschaut und in einem allgemeinen Feedback kommentiert.

Konzeptionelles Datenmodell

1.3. Datenintegrität

Unter dem Begriff Integrität oder Konsistenz (engl. integrity, consistency) versteht man die Widerspruchsfreiheit von Datenbeständen. Eine Datenbank ist integer oder konsistent, falls die gespeicherten Daten fehlerfrei erfasst sind und den gewünschten Informationsgehalt korrekt wiedergeben. Die Datenintegrität ist dagegen verletzt, wenn Mehrdeutigkeiten oder widersprüchliche Sachverhalte zu Tage treten.

1.3.1. Schlüsselintegritätsbedingung

Relationen sind Mengen von Tupeln, die allein durch ihre Werte unterschieden werden. Der Begriff Menge impliziert Eindeutigkeit der Elemente, d.h. es kann in einer Menge keine zwei Elemente geben, die die gleichen Werte besitzen. Tupel müssen folglich eindeutig identifizierbar sein.

Schlüsselkandidat:

Jedes Attribut oder jede minimale Attributkombination, welche jedes Tupel einer Relation eindeutig identifiziert, ist ein Schlüsselkandidat. Dabei bedeutet "minimal", dass kein Attribut ohne Verlust der eindeutigen Identifizierbarkeit weggelassen werden kann.

Primärschlüssel:

Der unter den Schlüsselkandidaten ausgewählte Identifikationsschlüssel wird zum Primärschlüssel der Relation.

Primärschlüssel werden meist unterstrichen dargestellt.

1.3.2. Gegenstandsintegritätsbedingung

Die Gegenstandsintegritätsbedingung folgt direkt aus der Schlüsselintegritätsbedingung und besagt, dass kein Primärschlüsselwert NULL (=kein Wert) sein darf. Erlauben wir NULL-Werte für Schlüsselattribute, so könnten mehrere Tupel NULL als Schlüsselwert besitzen. Damit wären diese Tupel nicht mehr eindeutig identifizierbar und die Schlüsselbedingung verletzt.

Beispiel:

ID	Name	Vorname	Geburtsjahr
NULL	Meier	Hans	1955
NULL	Meier	Hans	1985

Da der Primärschlüsselwert (Attribut ID) bei beiden Tupeln leer (NULL) ist, können wir die beiden Tupel nicht direkt identifizieren.

1.3.3. Referentielle Integritätsbedingung

Im Gegensatz zum GBM gibt es im relationalen Modell kein Konstrukt, mit dem Beziehungen zwischen Tupeln explizit modelliert werden können. Beziehungen werden hier implizit mit Hilfe des Primär-Fremdschlüssel-Konzepts dargestellt.

Fremdschlüssel:

Ein Attribut in einem Relationenschema R1 ist ein Fremdschlüssel wenn es in Beziehung zu einem Primärschlüsselattribut aus R2 steht und es gilt:

- Die Domäne des Fremdschlüssels aus R1 ist identisch mit der Domäne des Primärschlüssels aus R2

- Die Menge der Fremdschlüsselattributwerte von R1 muss eine Teilmenge der vorhandenen Primärschlüsselattributwerte von R2 sein.

Fremdschlüssel werden meist gestrichelt unterstrichen dargestellt.

Eine Beziehung zwischen zwei Relationen wird nun so hergestellt, dass die Domänen des Primärschlüssels der einen Relation in die zweite Relation als Fremdschlüsseldomänen (mit entsprechenden Attributen) aufgenommen werden.

Referentielle Integritätsbedingungen verlangen, dass aktuelle Fremdschlüsselwerte sich immer nur auf Primärschlüsselwerte von existierenden Tupeln beziehen.

Formal ausgedrückt heisst dies:

Gegeben sind:

- Ein Tupel `t1` einer Relation `R1` mit dem Schema`R1=(A1, ..., Am, ...)`, wobei `Ai` die Primärschlüsselattribute sind.
- Ein Tupel `t2` einer Relation `R2` mit dem Schema `R2=(B1, B2, ..., Bn, ..., A1, ..., Am, ...)`, wobei `Ai` die Fremdschlüsselattribute sind.

Das Tupel `t2` ist genau dann referentiell integer, wenn ein Tupel `t1` existiert mit der Eigenschaft `t1.Ai=t2.Ai (i=1, ..., m)`, oder wenn `t2.Ai =NULL (i=1, ..., m)` gilt.

Bildlich betrachtet, muss jedes Tupel der Fremdschlüsselrelation ein Tupel der Primärschlüsselrelation referenzieren (oder auf NULL gesetzt sein). Daher spricht man im Zusammenhang mit referentieller Integrität auch davon, dass keine "hängenden Referenzen" existieren dürfen (also Verweise auf etwas, das nicht existiert).

Das Relationenschema mit dem Fremdschlüssel wird als referenzierendes, das mit dem entsprechenden Primärschlüssel als referenziertes Relationenschema bezeichnet. Ein Fremdschlüssel kann dabei auch das eigene Relationenschema referenzieren. Die meisten der heutigen relationalen Datenbanksysteme unterstützen die automatische Einhaltung der referentiellen Integrität, zum Teil sogar schon PC-basierte Systeme.

Beispiel Referentielle Integrität

Die Tabelle Abteilung hat die Abteilungsnummer als Primärschlüssel. Dieser wird in der Tabelle Mitarbeiter als Fremdschlüssel verwendet, um die Abteilungszugehörigkeit eines Mitarbeiters zu fixieren. Die Fremd-Primärschlüssel-Beziehung erfüllt die Regel der referentiellen Integrität, falls alle Abteilungsnummern des Fremdschlüssels aus der Tabelle Mitarbeiter in der Tabelle Abteilung als Primärschlüsselwerte aufgeführt sind. In unserem Beispiel ist also die Regel der referentiellen Integrität nicht verletzt.

Nehmen wir an, wir möchten in die Tabelle Mitarbeiter, ein neues Tupel "Id: 4, Weber, A5" einfügen. Unsere Einfügeoperation wird abgewiesen, falls das Datenbanksystem die referentielle Integrität unterstützt. Der Wert A5 wird nämlich als ungültig erklärt, da er in der referenzierten Tabelle Abteilung nicht vorkommt.

1.3.4. Integritätsgefährdende Operationen

Wir unterscheiden und erläutern drei Arten von Operationen, die die Integrität im besprochenen Sinne gefährden können:

- Einfügen von Tupeln
- Löschen von Tupeln
- Ändern von Attributwerten eines Tupels

Bei all diesen Operationen sind alle Arten von Integritätsbedingungen zu beachten und einzuhalten. Anfragen, also das Wiederfinden von Daten in einer Datenbank, stellen, als reine Leseoperationen natürlich keine integritätsgefährdende Operation dar.

Einfügen von Tupeln
Bei dieser Operation sind alle drei Integritätsbedingungen betroffen. Folgende Verletzungen von Integritätsbedingungen sind möglich:

- In einem existierenden Tupel gibt es bereits den zur Einfügung vorgesehenen Primärschlüsselwert.
- Der Primärschlüsselwert des neuen Tupels ist NULL.
- Zu einem neuen Fremdschlüsselwert existiert kein zugehöriger Primärschlüsselwert.

Integritätsverletzende Operationen müssen vom Datenbanksystem entweder zurückgewiesen oder nach einem vereinbarten Protokoll in zulässige Operationen umgewandelt werden.

Beispiel Einfügen von Tupeln

Löschen von Tupeln

In diesem Fall kann nur die referentielle Integrität verletzt werden. Referenzierende Tupel können durch das Löschen des referenzierten Tupels betroffen sein, indem ihr Fremdschlüsselwert ungültig wird. Geeignete Gegenmassnahmen bestehen entweder im Zurückweisen der Löschoperation, im kaskadierenden Löschen (das referenzierende Tupel wird automatisch mitgelöscht) oder im Setzen der Fremdschlüsselattribute auf NULL.

Beispiel Löschen von Tupeln

Durch das Löschen des Tupels mit der Abt_Nr "A1" in der Relation Abteilung würde in der Relation Mitarbeiter im Tupel mit der ID "1" die Abteilungsnummer ungültig werden.

Ändern von Attributwerten

Diese Operation betrifft wiederum alle drei Arten von Integritätsbedingungen. Da eine Änderung auch als Kombination aus dem Löschen des alten Tupels und dem Einfügen des geänderten Tupels aufgefasst werden kann, treten hier die gleichen Probleme wie beim Einfügen- und Löschen von Tupeln auf. Werteänderungen sind nur bei Primär- und Fremdschlüsseln kritisch. Alle anderen Attribute können problemlos geändert werden.

Beispiel Ändern von Tupeln

Durch das Ändern des Wertes Abt_Nr in der Relation Abteilung von "A1" zu "A4" würde in der Relation Mitarbeiter im Tupel mit der ID "1" die Abteilungsnummer ungültig werden.

1.4. Normalisierungsprozess

Unter Normalisierung versteht man die systematische Untersuchung einer Relation mit dem Zweck, qualitativ hochwertige Relationen zu erhalten. Eine Relation ist dann normalisiert, wenn sie folgende Eigenschaften aufweist:

- sie ist redundanzfrei
- sie verursacht keine Probleme bei der Datenpflege
- sie beschreibt einen Ausschnitt aus der Realität angemessen und richtig

Unnormalisierte Relationen enthalten nichtatomare Attribute, d.h. die Attribute selbst besitzen eine gewisse Struktur z.B. eine Menge von Werten. Dies kann aber bereits durch eine sorgfältige ER-Modellierung verhindert werden.
Normalerweise sollte der Normalisierungsprozess schon auf konzeptueller Ebene stattfinden. Sein Ziel besteht darin, eine gewisse Einheitlichkeit in den Entwurf zu bringen. Allerdings sind bisher die für diesen Prozess erforderlichen Schritte nur auf der Ebene des relationalen Datenmodells sauber definiert worden. Daher werden auch wir die Normalisierung auf logischer Ebene betrachten.Der Entwerfer eines konzeptuellen Schemas sollte sich über die Normalisierungsprinzipien im klaren sein, um mit einem korrekten konzeptuellen Schema die Implementierung eines korrekten logischen Schemas zu erleichtern.

1.4.1. Abhängigkeiten

Um die Umwandlung der Relationen in die drei Normalformen zu verstehen, müssen wir zuerst das Konzept der Abhängigkeiten zwischen Attributen dieser Relationen einführen.

Funktionale Abhängigkeit:

Attribut B eines Gegenstandstyps G ist von Attribut A funktional abhängig, wenn zu jedem Wert von A höchstens ein Wert von B auftreten kann.

G.A --> G.B

Beispiel:

ID	Name
S1	Meier
S2	Weber

Das Attribut Name ist funktional abhängig vom Attribut ID (ID --> Name).

Identifikationsschlüssel:

Attribut A für die gilt: Jedes Attribut von G ist von A funktional abhängig; kein Attribut von A ist von den übrigen A-Attributen funktional abhängig.

G.A --> G.B

Beispiel:

ID	Name	Vorname
S1	Meier	Hans
S2	Weber	Ueli

Das Attribut ID ist Identifikationssschlüssel.

Volle funktionale Abhängigkeit:

A Identifikationsschlüssel eines Gegenstandstyps G, B Attribut; B ist genau dann von A voll funktional abhängig, wenn B von A funktional abhängig ist, aber nicht bereits von Teilen von A.

G.A ==> G.B

Beispiel:

IDStudent	Name	IDProfessor	Note
S1	Meier	P2	5
S2	Weber	P1	6

Das Attribut Note ist voll funktional abhängig von den Attributen IDStudent und IDProffesor (IDSt, IDProf ==> Note).

Transitive Abhängigkeit:

A Identifikationsschlüssel eines Gegenstandstyps G, B und C weitere Attribute, alle untereinander verschieden/disjunkt; C ist transitiv abhängig von A falls gilt:

G.A --> G.B ; G.B --> G.C ; G.B -/-> G.A

Beispiel:

ID	Name	Konto_Nr	Bank_Clearing_Nr	Bank
L1	Meier	1234-5	836	UBS
L2	Weber	5432-1	835	CS

Die funktionale Abhängigkeit bezüglich Bank_Clearing_Nr --> Bank ist eine transitive Abhängigkeit, da Bank_Clearing_Nr nicht Primärschlüssel der Relation ist.

1.4.2. 1. Normalform

1. Normalform:

Ein Relationenschema befindet sich in 1. Normalform, wenn alle seine Attribute einfach und einwertig sind.

Zur Verwaltung der Studenten sei folgende Relation gegeben:

Student(Vorname, Nachname, Informatikkentnisse)

Im Attribut Informatikkentnisse können mehrere Werte stehen.

Die Attribute Vorname und Nachname sind einfach und einwertig.

Beispiel 1. Normalform

Um die 1. Normalform zu erreichen, muss für jeden Wert eines mehrwertigen Attributes ein separates Tupel erzeug werden.

1.4.3. 2. Normalform

2. Normalform:

Ein Relationenschema ist in 2. Normalform, wenn es in 1. Normalform ist und wenn jedes nicht zum Identifikationsschlüssel gehörige Attribut von diesem voll funktional abhängig ist.

Zur Verwaltung der Prüfungsnoten sei folgende Relation gegeben:
Student(IDSt, StudentNachname, IDProf, ProfessorNachname, Note)
Die Attribute IDSt und IDProf bilden den Identifikationsschlüssel.
Alle Attribute sind einfach und einwertig.
Zudem ist bekannt, dass folgende funktionale Abhängigkeiten existieren:
1. Das Attribut ProfessorNachname ist funktional abhängig vom Attribut IDProf (IDProf --> ProfessorNachname)
2. Das Attribut StudentNachname ist funktional abhängig vom Attribut IDSt (IDSt --> StudentNachname)
3. Das Attribut Note ist voll funktional abhängig von den Attributen IDSt und IDProf (IDSt, IDProf ==> Note)

Beispiel 2. Normalform

Die obere Tabelle ist in 1. Normalform, da alle Attribute einfach und einwertig sind. Wenn jedoch der Student 1 von der Schule abgeht und gelöscht wird, gehen auch alle Informationen über den Professor Schmid verloren. Das Attribut Professor ist nämlich nicht voll funktional abhängig vom Identifikationsschlüssel IDSt. Um dieses Problem zu lösen, wird eine neue Relation Professoren mit den Attributen ProfID und Professor geschaffen. Die Relation Noten dient dazu, die beiden Relationen Studenten und Professoren zu verbinden und die Noten zu verwalten. In die Relation Noten wird die ID des Professors und des Studenten eingefügt. Auf diese Weise können die drei Relationen miteinander verknüpft werden. Das Problem des Verlusts von Professoren-Informationen bei der Löschung von Studenten ist damit behoben.

1.4.4. 3. Normalform

3. Normalform:

Ein Relationenschema befindet sich in 3. Normalform, wenn es in 2. Normalform ist und kein Attribut, das nicht zum Identifikationsschlüssel gehört, von diesem transitiv abhängt.

Zur Verwaltung der Bankverbindung von Lieferanten sei folgende Relation gegeben:
Lieferant(ID, Name, Konto_Nr, Bank_Clearing_Nr, Bank)
Das Attribut ID ist Identifikationsschlüssel. Alle Attribute sind einfach und einwertig.
Zudem ist bekannt, dass folgende funktionale Abhängigkeiten existieren:
1. Name, Konto_Nr, Bank_Clearing_Nr sind funktional abhängig von ID (ID --> Name, Konto_Nr, Bank_Clearing_Nr)
2. Bank ist funktional abhängig von Bank_Clearing_Nr (Bank_Clearing_Nr --> Bank)

Beispiel 3. Normalform

Da alle Attribute einfach und einwertig sind, befindet sich die Ausgangsrelation in 1. Normalform. Ausserdem befindet sie sich ebenfalls in 2. Normalform, da alle Attribute vom Identifikationsschlüssel voll funktional abhängig sind. Die funktionale Abhängigkeit bezüglich Bank_Clearing_Nr --> Bank ist eine transitive Abhängigkeit, da Bank_Clearing_Nr nicht Primärschlüssel der Relation LIEFERANT ist. Um die 3. Normalform zu erreichen, wird eine neue Relation erstellt und die bestehende geändert.

1.4.5. Übung Normalisierung

Die nachstehende Tabelle ist bereits in der 1. Normalform. Sie enthält pro Zeile und Spalte nur einen Eintrag. Ihre Aufgabe ist es, diese Relation so zu normalisieren, dass sie am Ende in der 3. Normalform vorliegt.
Dokumentieren Sie Ihre Lösung in einer Word- oder PDF-Datei und veröffentlichen Sie diese auf dem Diskussionsforum unter dem Thema 'Normalisierungsübung'. Schauen Sie sich die Lösungen Ihrer Mitstudierenden an und kommentieren Sie diese gegebenenfalls. Falls Sie Probleme oder Fragen haben, können Sie diese ebenfalls im Diskussionsforum stellen und diskutieren.
Die veröffentlichten Lösungen werden von einem Tutor angeschaut und in einem allgemeinen Feedback kommentiert.
Relation zur Verwaltung der Studierenden-Noten in den diversen Fächern

UnitID	StudentID	Datum	TutorID	Fach	Raum	Note	Buch	TutEmail
U1	St1	23.02.03	Tut1	GMT	629	4.7	Deumlich	tut1@fhbb.ch
U2	St1	18.11.02	Tut3	GIn	631	5.1	Zehnder	tut3@fhbb.ch
U1	St4	23.02.03	Tut1	GMT	629	4.3	Deumlich	tut1@fhbb.ch
U5	St2	05.05.03	Tut3	PhF	632	4.9	Dümmlers	tut3@fhbb.ch
U4	St2	04.07.03	Tut5	AVQ	621	5.0	SwissTopo	tut5@fhbb.ch

1.4.6. Unit-Zusammenfassung

Diese Unit hat aufgezeigt, wie und warum Relationen stufenweise normalisiert werden sollten. Unnormalisierte Relationen verursachen Probleme und damit auch Kosten. Eine sorgfältige Planung der Entitäten (im konzeptionellen Modell) und die Normalisierung des logischen Modells kann helfen dies zu vermeiden.

Relationen in 3. Normalform werden oft als „normalisiert“ bezeichnet. Obwohl, es gäbe noch weitere Normalformen (die vierte und fünfte). Die Anomalien, die vermieden werden durch eine Normalisierung bis zum fünften Grad sind aber sehr selten und so genügt es meistens bis zur dritten Normalform normalisieren.

1.5. Sommaire

Objekte der realen Welt werden im Relationenmodell durch Tabellen dargestellt. Jede Tabelle setzt sich aus Zeilen und Spalten zusammen. Die Tabelle ist also eine Sammlung aller zugehörigen Zeilen. Jede einzelne Zeile einer Tabelle, die auch als Tupel bezeichnet wird, setzt sich aus Datenfeldern zusammen, den Attributen. Die Attribute repräsentieren bestimmte Merkmale des entsprechenden Objektes der realen Welt. Jedes Attribut setzt sich aus einen bestimmten Attributnamen und Attributwert zusammen.
Relationen zwischen den einzelnen Tupeln sollen bestehende Beziehungen oder einen bestimmten Sachverhalt zwischen zwei Tabellen ausdrücken. Zusätzlich werden sog. Schlüsselattribute vergeben, um zum einen die Zuordnung (Relation oder Beziehung) zwischen Objekten (Tabellen) darzustellen und zum anderen den Zugriff auf eine Tabelle eindeutig zu definieren. Üblicherweise werden die Schlüsselattribute der jeweiligen Relation unterstrichen.
Unter dem Begriff Integrität oder Konsistenz (engl. integrity, consistency) versteht man die Widerspruchsfreiheit von Datenbeständen. Eine Datenbank ist integer oder konsistent, falls die gespeicherten Daten fehlerfrei erfasst sind und den gewünschten Informationsgehalt korrekt wiedergeben. Die Datenintegrität ist dagegen verletzt, wenn Mehrdeutigkeiten oder widersprüchliche Sachverhalte zu Tage treten.
Ein Relationsschema sollte einen Entity-Typ oder einen Beziehungstyp beschreiben. Das heisst eine Relation sollte Informationen beeinhalten, die tatsächlich auch logisch zusammengehören. Wird dies nicht beachtet, kann es zu Anomalien kommen. Um die Anomalien mehr oder weniger vollständig zu vermeiden, werden verschiedene Normalformen für relationale Datenbankschema vorgeschlagen.

1.6. Littérature recommandée

- **ELMASRI, R., NAVATHE, S.B.**, 1994. *Fundamentals of Database Systems*. 2nd. Redwood City, California: Addison-Wesley.
  Einführung ins Thema Datenbanken und SQL, auf Englisch

1.7. Glossaire

1. Normalform:

Ein Relationenschema befindet sich in 1. Normalform, wenn alle seine Attribute einfach und einwertig sind.

2. Normalform:

Ein Relationenschema ist in 2. Normalform, wenn es in 1. Normalform ist und wenn jedes nicht zum Identifikationsschlüssel gehörige Attribut von diesem voll funktional abhängig ist.

3. Normalform:

Ein Relationenschema befindet sich in 3. Normalform, wenn es in 2. Normalform ist und kein Attribut, das nicht zum Identifikationsschlüssel gehört, von diesem transitiv abhängt.

Attribut:

Ein Attribut **A** bezeichnet die Funktion, die eine Domäne D in einem Relationenschema R ausübt. Es kann auch als Abbildung der Tupel einer Relation auf den Wert des jeweiligen Tupels (für dieses Attribut) verstanden werden., wobei jeder Wert di ein Element der Domäne, oder NULL sein muss.

Domäne:

Eine Domäne besteht aus einem Namen D und einer Menge atomarer Werte. Ein anderer Name für Domäne ist Wertebereich. Domänen definieren den Wertebereich von Attributen.

Fremdschlüssel:

Ein Attribut in einem Relationenschema R1 ist ein Fremdschlüssel wenn es in Beziehung zu einem Primärschlüsselattribut aus R2 steht und es gilt:

- Die Domäne des Fremdschlüssels aus R1 ist identisch mit der Domäne des Primärschlüssels aus R2
- Die Menge der Fremdschlüsselattributwerte von R1 muss eine Teilmenge der vorhandenen Primärschlüsselattributwerte von R2 sein.

Fremdschlüssel werden meist gestrichelt unterstrichen dargestellt.

Funktionale Abhängigkeit:

Attribut B eines Gegenstandstyps G ist von Attribut A funktional abhängig, wenn zu jedem Wert von A höchstens ein Wert von B auftreten kann.

G.A --> G.B

Identifikationsschlüssel:

Attribut A für die gilt: Jedes Attribut von G ist von A funktional abhängig; kein Attribut von A ist von den übrigen A-Attributen funktional abhängig.

G.A --> G.B

Primärschlüssel:

Der unter den Schlüsselkandidaten ausgewählte Identifikationsschlüssel wird zum Primärschlüssel der Relation.

Primärschlüssel werden meist unterstrichen dargestellt.

Relation:

Eine Relation **r** ist eine Instanz (Ausprägung) des Relationenschemas R(A1, A2, ..., An). Sie ist eine Teilmenge des kartesischen Produkts (Kreuzprodukt) der beteiligten Domänen.

Relationales Datenbankschema:

Ein relationales Datenbankschema ist eine Menge von Relationenschemata S = {R1, ..., Rn} zusammen mit einer Menge von Integritätsbedingungen. Eine relationale Datenbankinstanz ist die Menge {r1, ..., rn} wobei ri Instanz von Ri ist und alle Integritätsbedingungen erfüllt sind. Eine relationale Datenbank ist ein relationales Datenbankschema mit einer entsprechenden Datenbankinstanz.

Relationenschema:

Ein Relationenschema **R**, Schreibweise: R(A1, A2, ..., An), bezeichnet eine Menge von Attributen {A1, A2, ..., An}.

Schlüsselkandidat:

Jedes Attribut oder jede minimale Attributkombination, welche jedes Tupel einer Relation eindeutig identifiziert, ist ein Schlüsselkandidat. Dabei bedeutet "minimal", dass kein Attribut ohne Verlust der eindeutigen Identifizierbarkeit weggelassen werden kann.

Transitive Abhängigkeit:

A Identifikationsschlüssel eines Gegenstandstyps G, B und C weitere Attribute, alle untereinander verschieden/disjunkt; C ist transitiv abhängig von A falls gilt:

G.A --> G.B ; G.B --> G.C ; G.B -/-> G.A

Tupel:

Ein Tupel **t** ist eine Liste mit n Werten **t = <d1, d2, ..., dn>**, wobei jeder Wert di ein Element der Domäne **Di**, oder NULL sein muss.

Volle funktionale Abhängigkeit:

A Identifikationsschlüssel eines Gegenstandstyps G, B Attribut; B ist genau dann von A voll funktional abhängig, wenn B von A funktional abhängig ist, aber nicht bereits von Teilen von A.

G.A ==> G.B

1.8. Bibliographie

- **ELMASRI, R., NAVATHE, S.B.**, 1994. *Fundamentals of Database Systems*. 2nd. Redwood City, California: Addison-Wesley.
- **ZEHNDER, C.A.**, 1998. *Informationssysteme und Datenbanken*. Zürich: vdf Hochschulverlag AG.