

Datenbanken



Das Relationenmodell

Thomas Studer

Institut für Informatik
Universität Bern

Repetition: Relation und kartesisches Produkt

- *n-stellige Relation* über den Mengen A_1, \dots, A_n :

$$R \subseteq \{(x_1, \dots, x_n) \mid x_1 \in A_1 \text{ und } \dots \text{ und } x_n \in A_n\} \ .$$

- (flaches) *kartesisches Produkt* zweier Relationen:

$$R \times S := \{(x_1, \dots, x_{m+n}) \mid (x_1, \dots, x_m) \in R \text{ und} \\ (x_{m+1}, \dots, x_{m+n}) \in S\} \ .$$

Aufgabe

Wir nehmen an, wir wollen die Daten zu einer von Menge von Autos in unserer Datenbank halten.

Für jedes Auto soll dessen **Marke und Farbe** abgespeichert werden.



Relationen und Tabellen

Wir können jedes Auto als Paar (Marke, Farbe) darstellen.

Eine Menge von Autos entspricht somit einer Menge von solchen Paaren.

Das heisst wir können eine Menge von Autos durch eine 2-stellige Relation repräsentieren.

$$\text{Autos} := \{(\text{Opel}, \text{silber}), (\text{VW}, \text{rot}), (\text{Audi}, \text{schwarz})\} .$$

Wir können diese Relation als Tabelle mit zwei Spalten darstellen:

Opel	silber
VW	rot
Audi	schwarz

Attribute und Domänen

Attribut Name der Spalte

Domäne Mögliche Werte, die ein Attribut annehmen kann

Autos

Marke	Farbe
-------	-------

Opel	silber
------	--------

VW	rot
----	-----

Audi	schwarz
------	---------

Null-Werte

Null wird verwendet, wenn der Wert eines Attributs *unbekannt* ist,

- weil wir den Wert nicht kennen oder
- oder, weil das Attribut keinen Wert besitzt.

Konvention

- *Null gehört zu jeder Domäne.*
- *In der Tabellenform schreiben wir häufig – anstelle von Null.*

Autos

Marke	Farbe	Fahrer
Opel	silber	Tom
VW	rot	-
Audi	schwarz	Eva

Null

I call it my billion-dollar mistake. It was the invention of the null reference in 1965. [...] This has led to innumerable errors, vulnerabilities, and system crashes, which have probably caused a billion dollars of pain and damage in the last forty years.

Tony Hoare

Semantik von Null

Da wir Null verwenden, um auszudrücken, dass der Wert eines Attributs unbekannt ist, erhält Null eine spezielle Semantik bezüglich der Gleichheit:

Es gilt *nicht*, dass $\text{Null} = \text{Null}$.

(1)

Wenn wir zwei unbekannte Werte vergleichen, so wissen wir eben nicht, ob sie gleich sind. Deshalb verwenden wir eine Semantik für die (1) der Fall ist.

Schwache Gleichheitsrelation

An einigen Stellen werden wir zwei `Null` Werte als gleichwertig betrachten müssen. Dazu führen wir folgende schwache Gleichheitsrelation zwischen zwei atomaren Objekten a und b ein:

$$a \simeq b \quad \text{g.d.w.} \quad a = b \quad \text{oder} \quad (a \text{ ist Null und } b \text{ ist Null}).$$

Für zwei n -Tupel definieren wir analog:

$$(a_1, \dots, a_n) \simeq (b_1, \dots, b_n) \quad \text{g.d.w.} \quad a_i \simeq b_i \text{ für alle } 1 \leq i \leq n.$$

Beispiel

Seien a , b , und c paarweise verschiedene Objekte, die verschieden von `Null` sind.

- $a \simeq a$
- $a \not\simeq b$
- $a \not\simeq \text{Null}$
- $\text{Null} \simeq \text{Null}$
- $(a, b) \simeq (a, b)$
- $(a, \text{Null}, c) \simeq (a, \text{Null}, c)$
- $(a, \text{Null}, c) \not\simeq (a, b, c)$

Relation und n -Tupel über Domänen D_1, \dots, D_n

Gegeben seien n Domänen D_1, \dots, D_n . Ein n -Tupel über D_1, \dots, D_n ist ein Objekt der Form

$$(a_1, \dots, a_n),$$

wobei $a_i \in D_i$ für alle $1 \leq i \leq n$. Eine n -stellige *Relation* R über den Domänen D_1, \dots, D_n ist eine Menge von n -Tupeln über D_1, \dots, D_n . Das heisst

$$R \subseteq \{(x_1, \dots, x_n) \mid x_1 \in D_1 \text{ und } \dots \text{ und } x_n \in D_n\} \text{ .}$$

Relationenschema

Relationenschemata (oder einfach nur *Schemata*) spezifizieren die bei Relationen verwendeten Attribute und Domänen. Es handelt sich dabei um Sequenzen der Form

$$(A_1 : D_1, \dots, A_n : D_n) ,$$

wobei A_1, \dots, A_n Attribute mit den jeweiligen Domänen D_1, \dots, D_n sind.

Beispiel

(Marke : text, Baujahr : integer, Farbe : color_enum, Fahrer : text)

Relationenschema

Konvention

- *Wir schreiben manchmal*

(A_1, \dots, A_n) anstelle von $(A_1 : D_1, \dots, A_n : D_n)$,

wenn sich die Domänen unmittelbar aus dem Kontext ergeben oder unwichtig sind.

- *Wir sagen R ist eine Relation über A_1, \dots, A_n , wenn R eine Relation über den dazugehörenden Domänen D_1, \dots, D_n ist.*
- *Wir sagen dann auch R ist eine Instanz des Schemas (A_1, \dots, A_n) .*

Relationales Datenbank-Schema

Als *relationales Datenbank-Schema* (oder kurz *DB-Schema*) bezeichnen wir die Menge aller verwendeten Relationenschemata.

Relationale Datenbank

Als *relationale Datenbank* (oder kurz *relationale DB*) bezeichnen wir das verwendete relationale Datenbank-Schema zusammen mit den momentanen Werten der Relationen.

Eine relationale Datenbank besteht somit aus

- einem DB-Schema und
- den aktuellen Instanzen aller Schemata des DB-Schemas.

Wir sprechen in diesem Zusammenhang auch von einer *Instanz* eines DB-Schemas und meinen damit die Menge der aktuellen Instanzen aller Schemata des DB-Schemas.

Projektion auf Attribute

Es seien A_1, \dots, A_n Attribute,

$$\mathcal{S} = (A_1, \dots, A_n)$$

ein Relationenschema und R eine Instanz von \mathcal{S} .

- ① Ist t ein n -Tupel, das zu R gehört, so schreiben wir $t[A_i]$ für den Wert von t bei Attribut A_i . Für $(a_1, \dots, a_i, \dots, a_n) \in R$ heisst das

$$(a_1, \dots, a_i, \dots, a_n)[A_i] = a_i \quad .$$

- ② Ist $K = (A_{i_1}, \dots, A_{i_m})$ eine Sequenz von Attributen, so definieren wir für ein n -Tupel $(a_1, \dots, a_i, \dots, a_n) \in R$

$$(a_1, \dots, a_i, \dots, a_n)[K] := (a_{i_1}, \dots, a_{i_m}) \quad . \quad (2)$$

Beispiel

Betrachten wir die Relation

Autos

Marke	Farbe	Baujahr	Vorname	Nachname
Opel	silber	2010	Tom	Studer
Opel	schwarz	2010	Eva	Studer
VW	rot	2014	Eva	Studer
Audi	schwarz	2014	Eva	Meier

Es sei nun

$$t := (\text{Opel}, \text{schwarz}, 2010, \text{Eva}, \text{Studer}) .$$

Damit gilt

$$t[(\text{Marke}, \text{Farbe})] = (\text{Opel}, \text{schwarz})$$

und

$$t[(\text{Nachname}, \text{Baujahr})] = (\text{Studer}, 2010) .$$

Beobachtung

Für $s, t \in R$ bedeutet $s[K] = t[K]$, dass die Werte von s und t in allen Attributen aus K übereinstimmen.

Schlüssel

Frage

Wie können wir in einer Instanz R von \mathcal{S} die einzelnen Elemente unterscheiden?

Primärschlüssel

Dazu wählen wir einen sogenannten *Primärschlüssel*. Dies ist eine Sequenz von Attributen

$$K = (A_{i_1}, \dots, A_{i_m}) \ .$$

Dann verlangen wir für alle Instanzen R von \mathcal{S} und alle $s, t \in R$, dass

$$s[K] = t[K] \quad \implies \quad s \simeq t \ .$$

Konvention

Wir geben den Primärschlüssel an, indem wir beim Relationenschema diejenigen Attribute unterstreichen, welche zum gewählten Primärschlüssel gehören.

Beispiel Primärschlüssel

Betrachten wir die Relation

Autos

Marke	Farbe	Baujahr	Vorname	Nachname
Opel	silber	2010	Tom	Studer
Opel	schwarz	2010	Eva	Studer
VW	rot	2014	Eva	Studer
Audi	schwarz	2014	Eva	Meier

Es sind bspw. die beiden folgenden Primärschlüssel möglich:

(Marke, Farbe) und (Baujahr, Vorname, Nachname) . (3)

Falls wir (Marke, Farbe) als Primärschlüssel wählen, so geben wir das Schema wie folgt an:

(Marke, Farbe, Baujahr, Vorname, Nachname) .

Sprechende Schlüssel

In einer echten Datenbankanwendung sind wahrscheinlich beide möglichen Primärschlüssel aus (3) ungeeignet.

Es ist nämlich gut möglich, dass wir später dieser Relation weitere Autos hinzufügen möchten. Da kann es dann sein, dass ein zweiter roter VW eingefügt werden soll oder ein weiteres Auto mit Baujahr 2010 und dem Fahrer Tom Studer.

Sprechende Schlüssel

In der Praxis wird oft ein zusätzliches Attribut, z.B. `AutoId`, hinzugefügt, welches als Primärschlüssel dient.

Dieses Attribut hat nur den Zweck, die verschiedenen Elemente der Relation eindeutig zu bestimmen. Es beschreibt aber keine echte Eigenschaft von Autos. Wir nennen einen solchen Primärschlüssel einen *nicht-sprechenden* Schlüssel.

Ein *sprechender* Schlüssel hingegen hat eine logische Beziehung zu einem oder mehreren Attributen des Schemas.

Sprechende Schlüssel

Es ist gute Praxis *keine* sprechenden Schlüssel zu verwenden, da diese die Tendenz haben zu zerbrechen.

Das heisst, früher oder später wird eine Situation auftreten, in der ein neues Tupel eingefügt werden soll, dessen sprechender Schlüssel bereits ein Tupel in der Relation bezeichnet. Oder es kann sein, dass das System der Schlüsselgenerierung komplett geändert wird. Beispiele dazu sind:

- 1 das System der AHV-Nummern (eindeutige Personennummer der Alters- und Hinterlassenenversicherung) in der Schweiz, welches 2008 geändert wurde,
- 2 die Internationale Standardbuchnummer (ISBN), für die 2005 ein revidierter Standard eingeführt wurde.

Probleme: Gefahr von Inkonsistenzen

Autos

Marke	Farbe	Baujahr	Vorname	Nachname
Opel	silber	2010	Tom	Studer
Opel	schwarz	2010	Eva	Studer
VW	rot	2014	Eva	Studer
Audi	schwarz	2014	Eva	Meier

In dieser Tabelle sind die Daten zu Eva Studer doppelt abgespeichert.

Besser: zwei Tabellen

Autos

<u>Marke</u>	<u>Farbe</u>	Baujahr	FahrerId
Opel	silber	2010	1
Opel	schwarz	2010	2
VW	rot	2014	2
Audi	schwarz	2014	3

Personen

<u>PersId</u>	Vorname	Nachname
1	Tom	Studer
2	Eva	Studer
3	Eva	Meier

FahrerId *referenziert* die Tabelle Personen

FahrerId ist ein *Fremdschlüssel* für Personen

Integritätsbedingungen

Unter *Integritätsbedingungen* (oder *Constraints*) werden Zusicherungen verstanden, welche die in der Datenbank enthaltenen Daten betreffen.

Wir unterscheiden

- 1 *Strukturelle Regeln* (statische Integritätsbedingungen), die in einem Zustand erfüllt sein müssen, damit er erlaubt ist.
- 2 *Verhaltensregeln* (dynamische Integritätsbedingungen), die bei der Ausführung von Änderungen erfüllt sein müssen.

Unique Constraints

Ein *unique Constraint* auf einem DB-Schema

$$\mathcal{S} = (A_1, \dots, A_n)$$

ist bestimmt durch eine Sequenz von Attributen

$$U = (A_{i_1}, \dots, A_{i_m}) .$$

Definition (Integritätsregel 1)

Gegeben sei ein Relationenschema \mathcal{S} mit einem unique Constraint U . Für jede Instanz R von \mathcal{S} und alle $s, t \in R$ muss gelten

$$s[U] = t[U] \implies s \simeq t .$$

Beispiel

Sei $\mathcal{S} := (\text{Marke}, \text{Farbe}, \text{Baujahr})$

mit einem unique Constraint $U := (\text{Marke}, \text{Farbe})$.

Die folgende Instanz Autos von \mathcal{S} erfüllt den unique Constraint U :

Autos		
Marke	Farbe	Baujahr
Opel	silber	2010
Null	schwarz	2012
Null	schwarz	2014

In der Tat ist es *nicht* der Fall, dass

$$(\text{Null}, \text{schwarz}) = (\text{Null}, \text{schwarz}) ,$$

da $\text{Null} = \text{Null}$ nicht gilt. Somit ist die Integritätsregel 1 erfüllt, obwohl die beiden Einträge mit $(\text{Null}, \text{schwarz})$ unterschiedliche Werte für das Attribut Baujahr haben.

Not null Constraints

Ein *not null Constraint* auf einem gegebenen DB-Schema $\mathcal{S} = (A_1, \dots, A_n)$ ist bestimmt durch ein Attribut A_i .

Definition (Integritätsregel 2)

Gegeben sei ein Relationenschema \mathcal{S} mit einem not null Constraint A_i . Für jede Instanz R von \mathcal{S} und alle $s \in R$ muss gelten

$s[A_i]$ hat nicht den Wert Null.

Primary key Constraints

Definition (Integritätsregel 3)

Gegeben sei ein Relationenschema \mathcal{S} mit einem Primärschlüssel K . Für jede Instanz R von \mathcal{S} und alle $s, t \in R$ muss gelten

$$s[K] = t[K] \implies s \simeq t .$$

Zusätzlich muss für jedes Attribute A_i , welches in K vorkommt, gelten:
Für jede Instanz R von \mathcal{S} und alle $s \in R$

$$s[A_i] \text{ hat nicht den Wert Null.}$$

Die Kombination aus unique Constraint und not null Constraints garantiert, dass der Primärschlüssel tatsächlich *alle* Elemente einer Relation eindeutig identifizieren kann. Der Fall aus dem vorherigen Beispiel wird durch die not null Constraints ausgeschlossen.

References Constraints

Es seien zwei Schemata $\mathcal{S}_1 = (A_1, \dots, A_n)$ und $\mathcal{S}_2 = (B_1, \dots, B_h)$ gegeben. Zusätzlich sei auf \mathcal{S}_2 ein unique Constraint $U := (B_{j_1}, \dots, B_{j_m})$ definiert worden. Ein *references Constraint* von \mathcal{S}_1 nach U ist bestimmt durch eine Sequenz

$$F := (A_{i_1}, \dots, A_{i_m})$$

von Attributen aus \mathcal{S}_1 , welche dieselbe Länge hat wie U . Wir sagen dann F *referenziert* U .

Definition (Integritätsregel 4)

Gegeben seien ein Relationenschema \mathcal{S}_2 mit einem unique Constraint U sowie ein Relationenschema \mathcal{S}_1 mit einem references Constraint $F = (A_{i_1}, \dots, A_{i_m})$ nach U . Für jede Instanz R von \mathcal{S}_1 und jede Instanz S von \mathcal{S}_2 muss gelten: Für jedes $t \in R$, falls

$$t[A_{i_k}] \text{ ist nicht Null für alle } 1 \leq k \leq m,$$

dann gibt es $s \in S$ mit

$$t[F] = s[U] \text{ .}$$

References Constraints 2

Da jeder primary key Constraint einen unique Constraint impliziert, können wir anstelle von U auch den Primärschlüssel von S_2 verwenden.

Da ein Schema nur einen Primärschlüssel besitzen kann, brauchen wir diesen nicht explizit anzugeben.

Wir sagen dann F *referenziert das Schema S_2* oder eben F *ist ein Fremdschlüssel (foreign key) für S_2* .

References Constraints 3

Autos

<u>Marke</u>	<u>Farbe</u>	Baujahr	FahrerId
Opel	silber	2010	1
Opel	schwarz	2010	2
VW	rot	2014	2
VW	blau	2015	-

Personen

<u>PersId</u>	Vorname	Nachname
1	Tom	Studer
2	Eva	Studer

Für den blauen VW ist der Fahrer unbekannt. Dieser Eintrag verletzt die referentielle Integrität nicht, da die Integritätsregel 4 nur Tupel betrifft, welche in den Attributen des references Constraints nicht den Wert Null haben.

Postulat

Das Einhalten der vier Integritätsregeln sollte von einem Datenbanksystem laufend automatisch überprüft werden.

- Unique Costraint
- Not Null Constraint
- Primary Key Constraint
- References Constraint