Relationale Algebra

Wechseln zu:Navigation, Suche

Dieser Artikel erfüllt die GlossarWiki-Qualitätsanforderungen **nur teilweise**:

Korrektheit: 2 (teilweise überprüft)	**Umfang**: 1 (zu gering)	**Quellenangaben**: 3 (wichtige Quellen vorhanden)	**Quellenarten**: 5 (ausgezeichnet)	**Konformität**: 5 (ausgezeichnet)

In der Theorie der Datenbanken versteht man unter einer **relationalen Algebra** oder **Relationenalgebra** eine Menge von Operationen zur Manipulation von Relationen. Sie ermöglicht es, Relationen zu filtern, zu verknüpfen, zu aggregieren oder anderweitig zu modifizieren, um Anfragen an eine Datenbank zu formulieren.[1][2]

Normalerweise werden Anfragen und Programme nicht direkt in einer relationalen Algebra formuliert, sondern in einer deklarativen Sprache wie SQL,[3] XQuery[4], SPARQL[5] oder auch Datalog[6]. Diese Programme und Anfragen werden üblicherweise zunächst in eine (i. Allg. erweiterte) relationale Algebra übersetzt. Der entstehende Operatorbaum wird dann mit Hilfe relationaler Gesetze transformiert, um eine möglichst effiziente Auswertung der Anfragen zu ermöglichen.[7]

Inhaltsverzeichnis

1 Geschichte

Im Jahr 1941 stellte Alfred Tarski in seinem Papier “On the calculus of relations” erstmals Ideen einer relationalen Algebra vor.[8] Insbesondere führte er die relationalen Operationen „Vereinigung“, „Durchschnitt“ und „Join“ ein, wobei er sich allerdings auf zweistellige Relationen beschränkte.

Am Ende seines Artikels erwähnt er, dass er eigentlich nicht so sehr das Ziel hatte, neue Ergebnisse zu präsentieren, als vielmehr das Interesse an einer bestimmten logischen Theorie zu wecken, die bislang nicht beachtet wurde:

> *The aim of this paper has been, not so much to present new results, as to awaken interest in a certain neglected logical theory, and to formulate some new problems concerning this theory.*

(Tarski (1941)[8])

Ende der 1960er-Jahre entwickelte Edgar F. Codd am IBM Research Laboratory in San Jose die Grundlagen der heutigen relationalen Algebra.[9][10] Ob ihn die Arbeit Tarskis dazu inspirierte, ist nicht bekannt. Zu Beginn seines Papiers von 1969 stellt er die Behauptung auf, dass das relationale Modell in vielen Aspekten dem Graphenmodell und dem Netzwerkmodell, die zu dieser Zeit „en vogue“ (franz. „in Mode“) waren, überlegen sei.

> *The first part of this paper is concerned with an explanation of a relational view of data. This view (or model) of data appears to be superior in several respects to the graph or network model [1, 2] presently in vogue. (Codd (1969)[9])*

Er bezieht sich damit auf die Tatsache, dass die Dauer der Beantwortung von Anfragen sehr stark vom Aufbau des jeweiligen Netzwerks abhängt. Sofern Daten abgerufen werden sollen, die im Netzwerk benachbart sind, muss der Benutzer nur sehr kurz auf eine Antwort warten. Sind die gewünschten Daten jedoch im Netzwerk stark verstreut, kann die Wartezeit unzumutbar lang werden. Die Datenbankentwickler mussten bei der Erstellung eines Netzwerkmodells von vorneherein sämtliche denkbaren Anfragen berücksichtigen, da nachträgliche Änderungen am Datenmodell nur noch sehr schwer umgesetzt werden konnten. Um dieses Problem zu beheben, hatte Codd die Idee, die Daten nicht mehr in einem Netzwerk zu speichern, sondern in Relationen (Tabellen), die je nach Anfrage unterschiedlich miteinander verknüpft werden können:

> *Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation (Codd (1970)[10])*

Er wagte folgende geradezu prophetische Prognose, dass Datenbanken künftig viele Relationen in gespeicherter Form enthalten würden:

> *The large, integrated data banks of the future will contain many relations of various degrees in stored form. (Codd (1969)[9])*

Ende 1970, d. h. im selben Jahr, in dem Codds Arbeit publik wurde, stellen Rudolf Bayer und Ed McCreight den B-Baum vor, eine Datenstruktur, die es ermöglicht, Relationen mit einer großen Anzahl von Tupel so auf einer Festplatte zu speichern, dass der lesende Zugriff auf Tupel sowie die Modifikation von Tupeln hocheffizient erfolgen kann.[11][12]

In den 1970er-Jahren begann auf Basis dieser beiden Arbeiten die Erfolgsgeschichte der Relationalen Datenbanken einschließlich der zugehörigen Sprache SQL. An Codds Arbeitsstätte, d. h. am IBM Research Laboratory in San Jose, wurden die Sprache SEQUEL sowie das experimentelle Datenbanksystem System R entwickelt. Später wurde SEQUEL in SQL umbenannt. Zu Beginn der 1980er-Jahre gab es für die Anfragesprache SQL die ersten kommerziellen relationalen Datenbanksysteme: Db2 von IBM und Oracle von Relational Software Inc.[13] Heute ist SQL aus der Welt der Datenbanken nicht mehr wegzudenken. Aber auch diverse weitere Sprachen, wie zunächst QBE[14] oder QUEL[15] und später Datalog[6], XQuery[16] oder SPARQL[5], basieren letztendlich auf der Idee Codds, Relationen zum Speichern von Daten einzusetzen.

2 Elemente einer Relationalen Algebra

Eine Relationale Algebra ist eine Algebra, deren Operationen Relationen auf Relationen abbilden. Abfragesprachen wie SQL, SPARQL, XQuery etc. liegt eine Relationale Algebra zugrunde. In derartigen

Sprachen werden allerdings üblicherweise Tabellen oder noch komplexere Datenstrukturen wie beispielsweise JSON oder XML an Stelle von Relationen eingesetzt.

Unterschiede

Relation	Tabelle
duplikatfrei: Relationen sind Mengen, daher kommt kein Tupel zweimal vor.	**Duplikate**: Tabellen sind Listen, daher können gleiche Tupel mehrfach vorkommen. Allerdings gilt dies in SQL nur bei berechneten Tabellen, nicht aber bei gespeicherten Tabellen.
ungeordnet: Relationen sind Mengen, daher sind die darin enthaltenen Tupel nicht geordnet. In einer Relationalen Algebra kann es keine Operation zum Sortieren von Relationen geben.	**geordnet**: Tabellen sind (geordnete) Listen, daher ist die Reihenfolge der Tupel festgelegt. Allerdings kann man sich bei SQL-Befehlen nicht darauf verlassen, dass das Ergebnis in einer bestimmten Reihenfolge ausgegeben wird. In SQL gibt es daher eine spezielle ORDER-BY-Klausel, um eine bestimmte Reihenfolge zu erzwingen.

Außerdem werden in SQL (Tabellen-)Zeilen (Rows) an Stelle von Tupeln verwendet. In Tabellenzeilen sind die einzelnen Spalten benannt und haben überdies eine Position.

Die klassische Tupeldefinition unterscheidet die verschiedenen Attribute anhand ihrer Position. Allerdings ist es nicht unüblich, diesen Positionen Namen zu geben. So wird bei einer Raumkoordinate die erste Position i. Allg. mit x bezeichnet, die zweite mit y und die dritte mit z.

In Programmiersprachen heißen Tupel, deren Elemente über die Position bestimmt werden, Arrays oder Listen: `[35, true, 'Hallo']`. Tupel, deren Elemente mittels Identifikatoren unterschieden werden, heißen Records, Hashmaps/Hasharrays oder auch – z. B. in JavaScript – Objekte: `{a: 35, b: true, c: 'Hallo'}`

Ein Attribut kann als geordnetes Paar aufgefasst werden, bestehend aus Attributbezeichner und Attributwert. Abhängig von der Art der Attributbezeichnung unterscheidet man folgende Tupelarten:

Positionstupel	Tupel mit Attributnamen	Rows (Tabellenzeilen)
Die Attribute werden über ihre Position bestimmt, d. h., der Attributbezeichner ist eine natürliche Zahl.	Die Attribute werden über ihre Namen bestimmt, d. h., der Attributbezeichner ist eine Zeichenkette.	Die Attribute werden über ihren Namen oder ihre Position bestimmt, d. h., der Attributbezeichner ist eine Zeichenkette oder eine natürliche Zahl.
`t = ['x', 'y', 'z']`	`t = {a:'x', b:'y', c:'z'}`	`t = {a/0:'x', b/1:'y', c/2:'z'}` Diese Art von Tupeln wird von üblichen Programmiersprachen nicht unterstützt.
`t[1] == 'y'`	`t['b'] == t.b == 'y'`	`t[1] == t['b'] == t.b == 'y'`

In SQL werden benannte Attribute, d. h. Tupel mit Attributnamen verwendet. Allerdings ist es üblich, dass in Hostsprachen auf beide Arten auf ein Attribut zugegriffen werden kann. Beispielsweise stellt JDBC (Java Database Connectivity) die Methode `getObject` zur Verfügung, die sowohl Integer- als auch Stringwerte als Attributbezeichner akzeptiert.[17]

Für Datenstrukturen wie `JSON` oder `XML` gelten analoge Aussagen: Es gibt an Stelle von Relationen allgemeinere Container, deren Elemente geordnet sind und Duplikate enthalten können. Außerdem gibt es Individuen, die als verallgemeinerte Tupel aufgefasst werden können. Dabei kommen sowohl Positions- als auch benannte Attribute zum Einsatz.

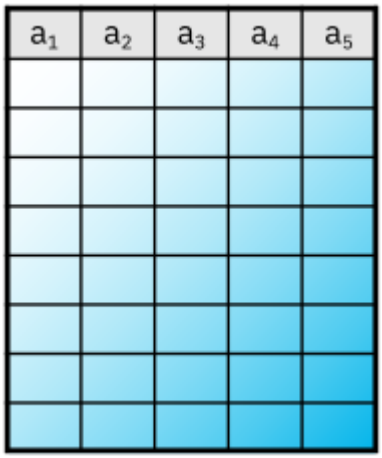
3 Definition (Kowarschick (2018))

Eine Algebra [math]\mathcal{R} = (R, id, \pi, \sigma, \times, \div, \cup, \cap, \setminus)[/math] heißt Relationale Algebra wenn die Trägermenge [math]R[/math] eine Menge von Relationen mit benannten Attributen ist.

[math]id: R \rightarrow R[/math] ist die so genannte Identitätsfunktion: Es gilt stets [math]id(r) = r[/math]
[math]\pi[/math] ist eine abzählbare Menge von (partiellen) Projektionsfunktionen [math]\pi_{f_1 \,\text{as}\, a_1, \,\ldots, \,f_n \,\text{as}\, a_n}: R \rightharpoonup R[/math]. Diese berechnen mit Hilfe der Funktionen f_i für jedes Tupel einer Relation r insgesamt n neue Attribue. Die neuen Attribute haben die Namen a_1 bis a_n. Jeder dieser Funktionen werden die Attribute des jeweiligen Tupels als Argumente übergeben. Das heißt, eine Projektionsfunktion kann nur auf diejenigen Relationen angewendet werden, deren Attribute kompatibel mit den Projektionsfunktionen sind.
[math]\sigma[/math] ist eine abzählbare Menge von (partiellen) Selektionsfunktionen [math]\sigma_b: R \rightharpoonup R[/math]. Mit ihrer Hilfe werden aus einer Relation r diejenigen Tupel selektiert, die die Bedingung b erfüllen, d. h., für die Funktion b den Wert `true` liefert, wenn sie auf das jeweilige Tupel angewendet wird. Eine Selektionsfunktion kann nur auf diejenigen Relationen angewendet werden, deren Attribute kompatibel mit den Funktion b sind.
[math]\times: R, R \rightharpoonup R[/math] ist das kartesische Produkt, das zwei Relationen zu einer Relation verknüpft. Für jede mögliche Kombination von zwei Tupeln der Urbildrelationen ist die Konkatenation der beiden Tupel Element der Bildrelation. Das kartesische Produkt kann nur auf Relationenpaare angewendet werden, deren Attributnamen sich unterscheiden (da ein Tupel mit benannten Attributen keinen Attributnamen mehrfach enthalten kann).
[math]\div: R, R \rightharpoonup R[/math] ist die Division. Im Prinzip ist dies die Umkehrfunktion des kartesischen Produktes.
[math]\cup: R, R \rightharpoonup R[/math] ermittelt die Vereinigung zweier (strukturgleicher) Relationen.
[math]\cap: R, R \rightharpoonup R[/math] ermittelt den Durchschnitt zweier (strukturgleicher) Relationen.
[math]\setminus: R, R \rightharpoonup R[/math] ermittelt die Differenz zweier (strukturgleicher) Relationen.

4 Relationale Operationen

4.1 Identität



Die Identitätsfunktion verändert eine Tabelle nicht.

Die Identitätsfunktion der Relationalen Algebra ist eine triviale Funktion: Sie bildet eine Relation (Tabelle) auf sich selbst ab:

[math]id: R \rightarrow R[/math]
[math]id(r) = r[/math]

Die Identiätsfunktion liefert als Ergebnisalso stets die Urbildrelation zurück. Das heißt, sie verändert eine Relation nicht.

Diese Funtion ist *idempotent*:

[math]id(id(r)) = id(r) = r[/math]

Die Identitätsfunktion ist die einzige totale Funktion der Relationalen Algebra. Das heißt, sie kann auf jede beliebige Relation angewendet werden.

In SQL wird eine derartige Funktion benötigt, um den gesamten Inhalt einer Relation zu erfragen, da in jeder SQL-Anweisung zum Lesen von Dateninhalten mindestens eine relationale Operation enthalten sein muss.

In SQL 92[18] wurde zu diesem Zweck der Befehl `table` eingeführt, dessen einzige Aufgabe es ist, die ihm übergebene Tabelle (Relation) vollständig auszugeben. Dieser Befehl ist jedoch nicht notwendig, da eine Projektion, die alle Attribute einer Relation unverändert zurück gibt, ebenfalls als Identitätsfunktion verwendet werden kann (siehe nachfolgenden Abschnitt). PostgreSQL beispielsweise unterstützt ihn dennoch.[19]

Beispiel

r					→	$id(r)$				
a_1	**a_2**	**a_3**	**a_4**	**a_5**		**a_1**	**a_2**	**a_3**	**a_4**	**a_5**
a	b	c	5	13	→	a	b	c	5	13
a	d	b	7	17		a	d	b	7	17
c	f	g	3	21		c	f	g	3	21

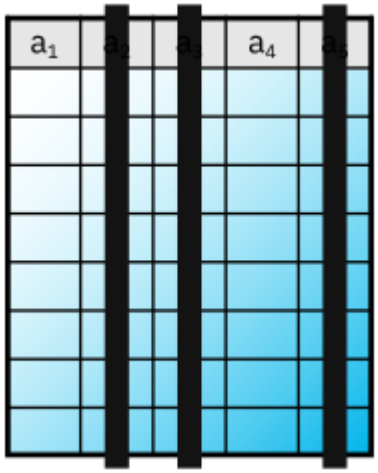
SQL

```sql
TABLE r;  -- seit SQL 92

SELECT a1, a2, a3, a4, a5 FROM r;
```

Siehe auch Händler-Datenbank (SQL-Beispiel)/Identität

4.2 Projektion



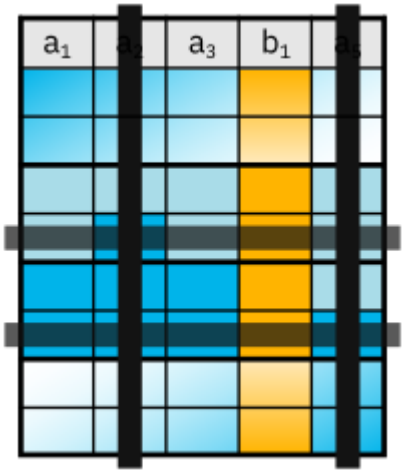
Eine Projektionsfunktion entfernt Spalten aus einer Tabelle.

Die Projektionsfunktionen π wurden von Codd eingeführt, um Attribute einer Relation entfernen können. Eine Projektionsfunktion

[math]π_{c_1, \ldots, c_k}: R \rightarrow R[/math]

dient also dazu, bestimmte Spalten aus einer beliebigen (d. h. gespeicherten oder berechneten Tabelle) zu selektieren. Wenn $r(a_1, \ldots, a_n)$ eine Relation ist und $\{c_1, \ldots, c_k\}$ eine Teilmenge der Attribute $\{a_1, \ldots, a_n\}$, dann ist $π_{c_1, \ldots, c_k}(r)$ diejenige Tabelle, die aus r entsteht, wenn man alle übrigen Spalten aus r entfernt. Dabei entstehende Duplikate müssen ebenfalls entfernt werden, sofern die relationale Algebra – wie von Codd ursprünglich gefordert – mengenbasiert ist. In SQL werden nur gespeicherte Tabellen als duplikatifreie Mengen behandelt, berechnete Tabellen sind dagegen Multimedngen, die Duplikate enthalten dürfen.

Außerdem führte Codd noch Permutationsfunktionen ein, um die Reihenfolge der Attribute zu ändern.[9][10] Der Begriff Permutation ist allerdings nur für Tupel mit Positionsattributen ist (`[10,30]` und `[30,10]` bezeichnen zwei verschiedene Punkte in einer Ebene). Für Tupel mit Attributnamen benötigt man dagegen keine Funktion, die Attribute vertauscht, da benannte Attribute keine Positionen besitzen (`{x: 10, y: 30}` und `{y: 30, x: 10}` bezeichnen denselben Punkt in einer Ebene). Hier muss eine Permutationsfunktion Attributnamen umbenennen können.



Genauer: Eine Projektionsfunktion berechnet neue Spalteninhalte aus den alten Spalteninhalten. Dabei entstehende Duplikattupel werden ebenfalls entfernt.

In der Zwischenzeit werden die Projektionsfunktionen allgemeiner definiert. Sie berechnen für jedes Tupel einer Relation aus dessen Attributwerten ein neues Tupel. Für die Benennung der neuen Attributwerte ist ebenfalls die Projektionsfunktion zuständig. Spezielle Permutationsfunktionen sind damit überflüssig geworden.

[math]π_{f_1 \,\text{as}\, b_1, \ldots, f_k \,\text{as}\, b_k}: R \rightarrow R[/math]

f_i ist dabei eine Funktion, die für jedes Tupel $(v_1:a_1, \ldots, v_n:a_n) \in r$ jeweils einen Wert $f_i((v_1:a_1, \ldots, v_n:a_n))$ berechnet, der in das Ergebnistupel unter dem Namen b_i eingefügt wird.

Die erste Definition ist dabei ein Spezialfall dieser allgemeineren Definition, wenn man c_i sowohl als Attributnamen also auch als Funktion auffasst: c_i entspricht $c_i \,\text{as}\, c_i$, wobei die Funktion c_i folgendermaßen definiert wird: $c_i((v_1:a_1, \ldots, v_n:a_n)) = v_j$, falls der Attributename c_i gleich a_j ist.

Beachten Sie bitte: Die folgenden Beispiele funktionieren sowohl für Positionsattribute als auch für benannte Attribute als auch für Rows, bei denen Attribute sowohl eine Position als auch einen Namen haben.

Beispiele

r → **$\pi_{a_4, \,a_1}(r)$**

$a_1	$a_2	$a_3	$a_4	$a_5
c	b	c	5	13
a	d	g	7	17
a	f	g	3	21

→

$a_4	$a_1
5	c
7	a
3	a

SQL

```
SELECT a4, a1 FROM r
```

Anmerkung: $\pi_{a_4, \,a_1}(r)$ ist eine Abkürzung für $\pi_{a_4 \rm{\,as\,} a_4, \,a_1 \rm{\,as\,} a_1}(r)$.

Attribute können auch berechnet und (um-)benannt werden. Duplikate, wie z. B. (a, g, 24), werden in

der Relationalen Algebra automatisch entfernt. In SQL werden Duplikate nur dann entfernt, wenn man das Schlüsselwort „`DISTINCT`“ angibt. SQL liegt eine so genannte Multimengen-Semantik zu Grunde.

r → **$\pi_{a_1, \,a_3, \,a_4+a_5 \rm{\,as\,} b_1}(r)$**

a_1	a_2	a_3	a_4	a_5
c	b	c	5	13
a	d	g	7	17
a	f	g	3	21

→

a_4	a_3	b_1
c	a	18
a	g	24

SQL

```
SELECT DISTINCT
a4, a3, a4+a5 AS
b1
FROM r
```

Die Identitätsfunktion kann – wie es bereits weiter oben angesprochen wurde – mit Hilfe der Projektionsfunktion nachgebildet werden. Allerdings benötigt man für jede Relation eine spezielle Projektionsfunktion. In der Projektiosnliste müssen alle Attribute der zugehörigen Funktion genau einmal aufgeführt werden:

r → **$\pi_{a_1, \,a_2, \,a_3, \,a_4, \,a_5}(r)$**

a_1	a_2	a_3	a_4	a_5
c	b	c	5	13
a	d	g	7	17
a	f	g	3	21

→

a_1	a_2	a_3	a_4	a_5
c	b	c	5	13
a	d	g	7	17
a	f	g	3	21

SQL

```
SELECT a1,
a2, a3, a4,
a5 FROM r;
SELECT r.*
FROM r;
SELECT *
FROM r;
```

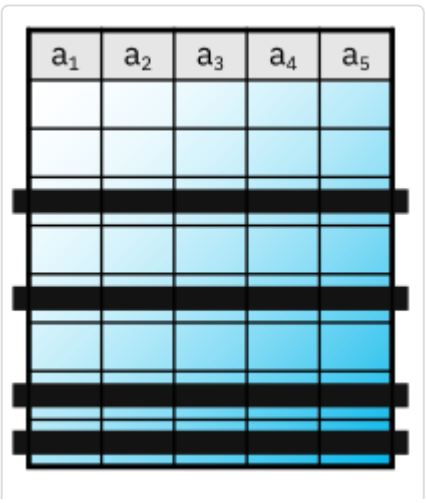
In SQL kann man abkürzend den Stern „*“ verwenden. Dieser bezeichnet alle Attribute der aktuellen Relation. Mit `r.*` werden in SQL alle Attribute der Relation `r` bezeichnet. Für das obige Beispiel kann man in der Relationalen Algebra analog folgenden Abkürzungen definieren:

$ r = \pi_{a_1, \,a_2, \,a_3, \,a_4, \,a_5}(r) = \pi_{r.*}(r) = \pi_{*}(r) = id(r)$

Mit dem Stern-Operator man also eine alternative Definition der Identitätsfunktion mit Hilfe des Projektionsoperators angeben, die ohne die Attributnamen der jeweiligen Relation auskommt. (In SQL ist die Verwendung des Sternoperator allerdings verpönt, da sich die Attribute einer Tabelle im Laufe der Zeit ändern können (Schemaevolution) und sich damit die Bedeutung des Sterns ändert. Das heißt, Anfragen, die einen * enthalten, können plötzlich weniger, mehr oder andere Spalten im Ergebnis liefern.)

Siehe auch Händler-Datenbank (SQL-Beispiel)/Projektion

4.3 Selektion



Eine Selektionsfunktion entfernt Zeilen (= Tupel) aus einer Tabelle.

Eine Selektionsfunktion entfernt Zeilen (= Tupel) aus einer Tabelle.

r → **$\sigma_{a_1= \tt{'a'}}(r)$**

a_1	a_2	a_3	a_4	a_5
c	b	c	5	13
a	d	g	7	17
a	f	g	3	21

→

a_1	a_2	a_3	a_4	a_5
a	d	g	7	17
a	f	g	3	21

SQL

```
SELECT *
FROM r
WHERE a1 = 'a';
```

r → **$\sigma_{a_1 = \tt{'a'} \,\wedge\, a_5<21}(r)$**

a_1	a_2	a_3	a_4	a_5
c	b	c	5	13
a	d	g	7	17
a	f	g	3	21

→

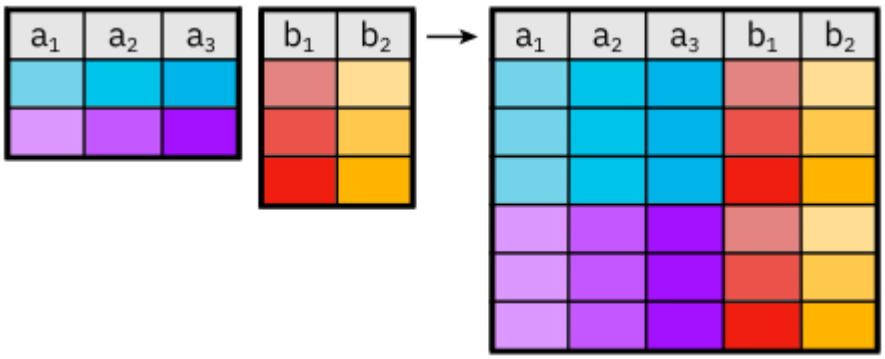
a_1	a_2	a_3	a_4	a_5
a	d	g	7	17

SQL

```
SELECT *
FROM r
WHERE a1 =
'a' AND a5
< 20;
```

Siehe auch Händler-Datenbank (SQL-Beispiel)/Selektion

4.4 Kartesisches Produkt



Das Kartesische Produkt verknüpft alle möglichen Tupelpaare zu jeweils einem Tupel, das alle Attribute beider Tupel enthält.

Das Kartesische Produkt verknüpft alle möglichen Tupelpaare zu jeweils einem Tupel, das alle Attribute beider Tupel enthält.

r **$\times$** **s** → **$r \times s$**

a_1	a_2	a_3	a_4	a_5
c	b	c	5	13
a	d	g	7	17
a	f	g	3	21

$\times$

b_1	b_2
true	3.14
false	2.71

→

a_1	a_2	a_3	a_4	a_5	b_1	b_2
c	b	c	5	13	true	3.14
a	d	g	7	17	true	3.14
a	f	g	3	21	true	3.14
c	b	c	5	13	false	2.71
a	d	g	7	17	false	2.71
a	f	g	3	21	false	2.71

SQL

```
SELECT *
FROM r, s
```

Siehe auch Händler-Datenbank (SQL-Beispiel)/Kartesisches Produkt

4.5 Join

Ein Join ist formal nichts weiter, als ein Kartesisches Produkt gefolgt von einer Selektion.

Siehe auch Händler-Datenbank (SQL-Beispiel)/Join

> TO BE DONE
>
> $□$, [math]□[/math], [math]□[/math], [math]□[/math], $\triangle$

5 Quellen

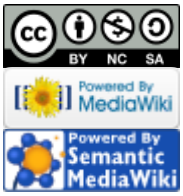
2. **Ullman (1988)**: Jeffrey D. Ullman; Principles of Database and Knowledge-Base Systems – Volume I: Classical Database Systems; Verlag: Computer Science Press; Adresse: New York, Oxford; ISBN: 0-7167-8158-1; Web-Link; 1988; Quellengüte: 5 (Buch), Seite=53
4. **Elmasri, Navathe (2002)**: Ramez Elmasri und Shamkant B. Navathe; Grundlagen von Datenbanksystemen; Auflage: 3; Verlag: Pearson Studium; ISBN: 3-8273-7021-3; 2002; Quellengüte: 5 (Buch), Seite 242
6. **Ullman (1988)**: Jeffrey D. Ullman; Principles of Database and Knowledge-Base Systems – Volume I: Classical Database Systems; Verlag: Computer Science Press; Adresse: New York, Oxford; ISBN: 0-7167-8158-1; Web-Link; 1988; Quellengüte: 5 (Buch), Seite=210
8. **Grust, Teubner (2004)**: Torsten Grust und Jens Teubner; Relational Algebra: Mother Tongue – XQuery; Proceedings of the first Twente Data Management Workshop on XML Databases; Seite(n): 9–16; Verlag: Association for Computing Machinery; Adresse: Enschede, The Netherlands; Web-Link; 2004; Quellengüte: 5 (Konferenzartikel), Seiten 9–16
10. **Cyganiak (2005)**: Richard Cyganiak; A Relational Algebra for SPARQL; Organisation: Hewlett-Packard Development Company; Adresse: Bristol; Web-Link 0, Web-Link 1; 2005; Quellengüte: 3 (Technischer Bericht)
12. **Kießling, Köstler (1998)**: Werner Kießling und Gerhard Köstler; Multimedia-Kurs Datenbanksysteme; Verlag: Springer-Verlag; Adresse: Berlin/Heidelberg; ISBN: 3-540-63836-9; 1998; Quellengüte: 5 (Buch)
14. **Ullman (1989)**: Jeffrey D. Ullman; Principles of Database and Knowledge-Base Systems – Volume II: The New Technologies; Verlag: Computer Science Press; Adresse: New York, Oxford; ISBN: 0-7167-8069-O, 0-7167-8182-X; Web-Link; 1989; Quellengüte: 5 (Buch), Chapter 11
16. **Tarski (1941)**: Alfred Tarski; On the Calculus of Relations; in: The Journal of Symbolic Logic; Band: 6; Nummer: 3; Seite(n): 73–89; Verlag: Association for Symbolic Logic; Adresse: New York; Web-Link; 1941; Quellengüte: 5 (Artikel)
18. **Codd (1969)**: Edgar Frank Codd; Derivability, Redundancy and Consistency of Relations Stored in Large Data Banks; in: ACM SIGMOD Record; Band: 38; Nummer: 1; Seite(n): 17–36; Verlag: Association for Computing Machinery; Adresse: New York; Web-Link; 2009; Quellengüte: 5 (Artikel)
20. **Codd (1970)**: Edgar Frank Codd; A Relational Model of Data for Large Shared Data Banks; in: Communications of the ACM; Band: 13; Nummer: 6; Seite(n): 377-387; Verlag: Association for Computing Machinery; Adresse: New York; Web-Link; 1970; Quellengüte: 5 (Artikel)
22. **Bayer, McCreight (1970)**: Rudolf Bayer und Edward M. McCreight; Organization and Maintenance of Large Ordered Indexes; Proceedings of the 1970 ACM SIGFIDET (SIGFIDET '70); Seite(n): 107–141; Verlag: Association for Computing Machinery; Adresse: New York; Web-Link; 1970; Quellengüte: 5 (Konferenzartikel)

24. **Bayer, McCreight (1972)**: Rudolf Bayer und Edward M. McCreight; Organization and Maintenance of Large Ordered Indexes; in: Acta Informatica; Band: 1; Nummer: 3; Seite(n): 173-189; Verlag: Springer-Verlag; Web-Link; 1972; Quellengüte: 5 (Artikel)
26. **Leon, Leon (1999)**: Alexis Leon und Mathews Leon; SQL – A Complete Reference; Verlag: Tata McGraw-Hill; Adresse: New Delhi; ISBN: 978-0-07-463708-1; 1999; Quellengüte: 5 (Buch)
28. **Ullman (1988)**: Jeffrey D. Ullman; Principles of Database and Knowledge-Base Systems – Volume I: Classical Database Systems; Verlag: Computer Science Press; Adresse: New York, Oxford; ISBN: 0-7167-8158-1; Web-Link; 1988; Quellengüte: 5 (Buch), Seiten 195–210
30. **Ullman (1988)**: Jeffrey D. Ullman; Principles of Database and Knowledge-Base Systems – Volume I: Classical Database Systems; Verlag: Computer Science Press; Adresse: New York, Oxford; ISBN: 0-7167-8158-1; Web-Link; 1988; Quellengüte: 5 (Buch), Seiten 185–195
32. **Grust, Teubner (2004)**: Torsten Grust und Jens Teubner; Relational Algebra: Mother Tongue – XQuery; Proceedings of the first Twente Data Management Workshop on XML Databases; Seite(n): 9–16; Verlag: Association for Computing Machinery; Adresse: Enschede, The Netherlands; Web-Link; 2004; Quellengüte: 5 (Konferenzartikel)
34. **Taylor (2003)**: Art Taylor; JDBC – Database Programming with J2EE; Auflage: 5; Verlag: Prentice Hall; Adresse: Upper Saddle River, New York, USA; ISBN: 0-13-045323-4; 2003; Quellengüte: 5 (Buch), Seite 332–333
36. **Date, Darwen (1993)**: Christopher J. Date und Hugh Darwen; A Guide to the SQL Standard – A user's guid to the standard relational language SQL; Auflage: 3; Verlag: Addison-Wesley; Adresse: Reading, Massachusetts, USA; 1993; Quellengüte: 5 (Buch)
38. https://www.postgresql.org/docs/current/static/sql-select.html PostgreSQL: Select
1. **Ullman (1988)**: Jeffrey D. Ullman; Principles of Database and Knowledge-Base Systems – Volume I: Classical Database Systems; Verlag: Computer Science Press; Adresse: New York, Oxford; ISBN: 0-7167-8158-1; Web-Link; 1988; Quellengüte: 5 (Buch)
2. **Ullman (1989)**: Jeffrey D. Ullman; Principles of Database and Knowledge-Base Systems – Volume II: The New Technologies; Verlag: Computer Science Press; Adresse: New York, Oxford; ISBN: 0-7167-8069-O, 0-7167-8182-X; Web-Link; 1989; Quellengüte: 5 (Buch)
3. **Garcia-Molina, Ullman, Widom (2002)**: Hector Garcia-Molina, Jeffrey D. Ullman und Jennifer Widom; Database Systems: The Complete Book; Verlag: Prentice Hall; Adresse: New Jersey, Upper Saddle River; ISBN: 0-13-031995-3; Web-Link; 2002; Quellengüte: 5 (Buch)
4. **Elmasri, Navathe (2011)**: Ramez Elmasri und Shamkant B. Navathe; Fundamentals of Database Systems; Auflage: 3; Verlag: Pearson Studium; ISBN: 978-0-136-08620-8; 2011; Quellengüte: 5 (Buch)

6 Siehe auch

Händler-Datenbank (SQL-Beispiel)

Kategorien:
Algebraische Struktur
Mathematische Definition
Datenmanagement
Glossar


BY NC SA
Powered By MediaWiki
Powered By Semantic MediaWiki