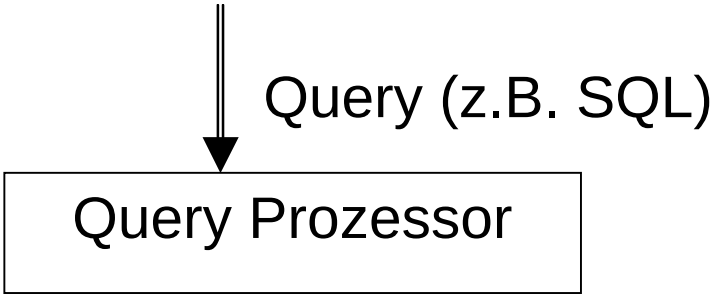
Logische Optimierung

Höhere, nichtprozedurale Abfragesprachen (SQL, QBE, ...) verlangen keine Kenntnisse des Benutzers über die Implementierung, müssen aber in prozedurale Form (z. B. Relationenalgebra) umgesetzt werden.

Um trotzdem effiziente Bearbeitung von Queries zu erzielen wird die gestellte Anfrage intern umformuliert und verbessert. Diesen Vorgang nennt man ***Query Optimization***.

Im Allgemeinen wird keine optimale Lösung erzielt, sondern nur eine Verbesserung.



- Analyse
- Umwandlung in relationale Algebra
- Datenzugriff
- Ausführung

Fragen:

Welche Operationen benötigen viel Zeit für ihre Ausführung?

Können diese vermieden werden, indem man die Anfrage neu formuliert?

Beispiel:

Gegeben sei folgender relationaler Ausdruck:

$$\pi_A(\sigma_{B=C \wedge D=\text{”99”}}(AB \times CD))$$

Dieser offensichtlich teuere Ausdruck (wg. kartesischem Produkt) kann besser formuliert werden:

$$\pi_A(\sigma_{B=C}(AB \times \sigma_{D=\text{”99”}}(CD)))$$

Das kartesische Produkt in dieser Abfrage ist offensichtlich durch die Selektion über B=C äquivalent zu einem Gleichverbund:

$$\pi_A(AB\ [B=C]\ \sigma_{D=\text{”99”}}(CD))$$

Grundlegende Aspekte

Zu betrachten sind für eine Optimierung die fünf Grundoperationen. Wo liegt ihr Schwachpunkt und wie können diese Schwachpunkte umgangen werden?

- Die auf jeden Fall aufwendigste Operation ist das *kartesische Produkt* bzw. der *Verbund*:
  Bei einfachster Implementierung eines Verbundes zwischen A und B erfolgt ein Durchlauf aller Tupel von B für jedes Tupel von A. Dies ist ein Aufwand mit o(nm).
- Die *Projektionen* sind aufwendig durch das Entfernen von Duplikaten.
- Die *Selektionen* sollte man so früh wie möglich durchführen, da dies zu kleineren Zwischenresultaten führt.

- Die unären Operationen (Projektion/Selektion) bedingen je einen Durchlauf aller Tupel, daher mehrere möglichst zusammenziehen oder mit einer binären Operation zusammenfassen.
- Nach gemeinsamen Teilausdrücken suchen, damit diese nur einmal abgearbeitet werden.
- Eventuell temporäre Verwendung bestimmter Dateiorganisationen (Indizes, Sortieren) einführen
  → **Physische Optimierung**

Der Zeitaufwand für das Untersuchen der verschiedenen Möglichkeiten ist im Allgemeinen viel geringer als für das Durchführen einer ineffizienten Query. Daher wird die Optimierung immer durchgeführt!

Algebraische Manipulation

- Gesetze der relationalen Algebra.
- Äquivalenz von Ausdrücken.

  Es gilt $E_1 \equiv E_2$ falls sie dieselbe Abbildung repräsentieren, d.h. falls dieselben Relationen für identische Bezeichnungen in den beiden Ausdrücken eingesetzt werden, erhallten wir gleiche Ergebnisse.

Vereinfachungen und gemeinsame Unterausdrücke

$$r \cup r \equiv r$$

$$r \cap r \equiv r$$

$$r \bowtie r \equiv r$$

$$r - r \equiv \varnothing$$

$$r \cup \varnothing \equiv r$$

$$r \cap \varnothing \equiv \varnothing$$

$$r \bowtie \varnothing \equiv \varnothing$$

$$r - \varnothing \equiv r$$

$$\varnothing - r \equiv \varnothing$$

$$\pi_x(\varnothing) \equiv \varnothing$$

$$\sigma_c(\varnothing) \equiv \varnothing$$

$$\delta_N(\varnothing) \equiv \varnothing$$

$$r[c]\varnothing \equiv \varnothing$$

$$\varnothing[c]r \equiv \varnothing$$

Beispiel:

$$(r - r) \bowtie (r \cup r)$$

$$\equiv \varnothing \bowtie r$$

$$\equiv \varnothing$$

Diese Regeln gelten auch für gemeinsame Unterausdrücke:

$$(r \bowtie s) \cup (r \bowtie s) \equiv r \bowtie s$$

Operatorbäume oder Ausdrucksbäume

Mittels Operatorbäumen können gemeinsame Unterausdrücke und äquivalente Ausdrücke leicht erkannt werden.

Ein Operatorbaum ist folgendermaßen charakterisiert:

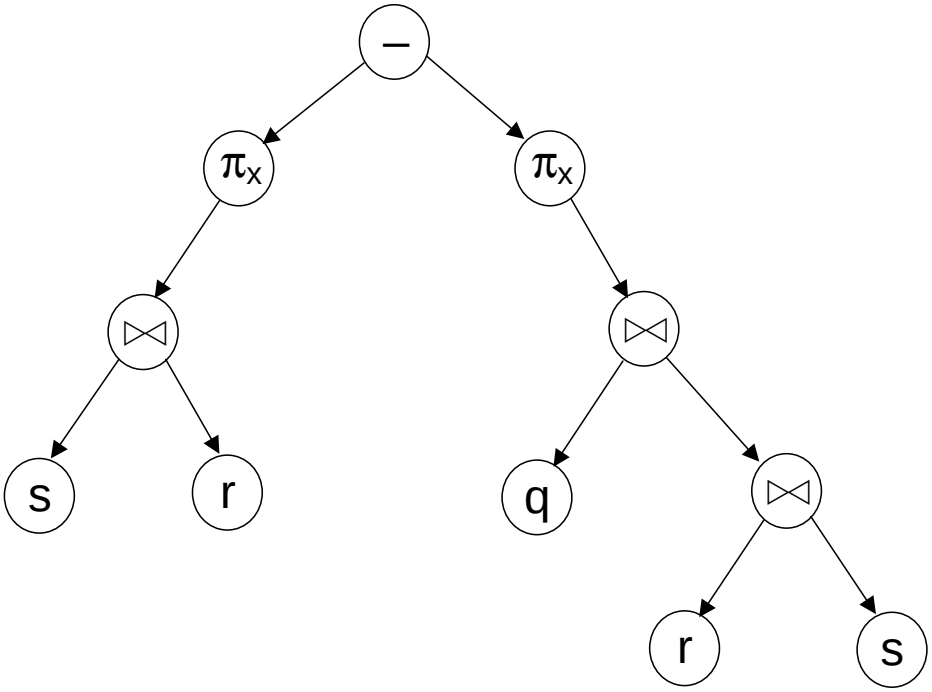
- Gerichteter Baum
- Die Knoten sind Operatoren
- Die Blätter sind Relationen oder Konstanten

Algebraische Optimierung

Für die Durchführung der Optimierung wird die Anfrage zunächst in einen Operatorbaum umgewandelt:

Beispiel:

$\pi_X(s \bowtie r) - \pi_X(q \bowtie r \bowtie s)$



Annahme:

Bottom-up-Auswertung des Operatorbaumes

Aufwand:

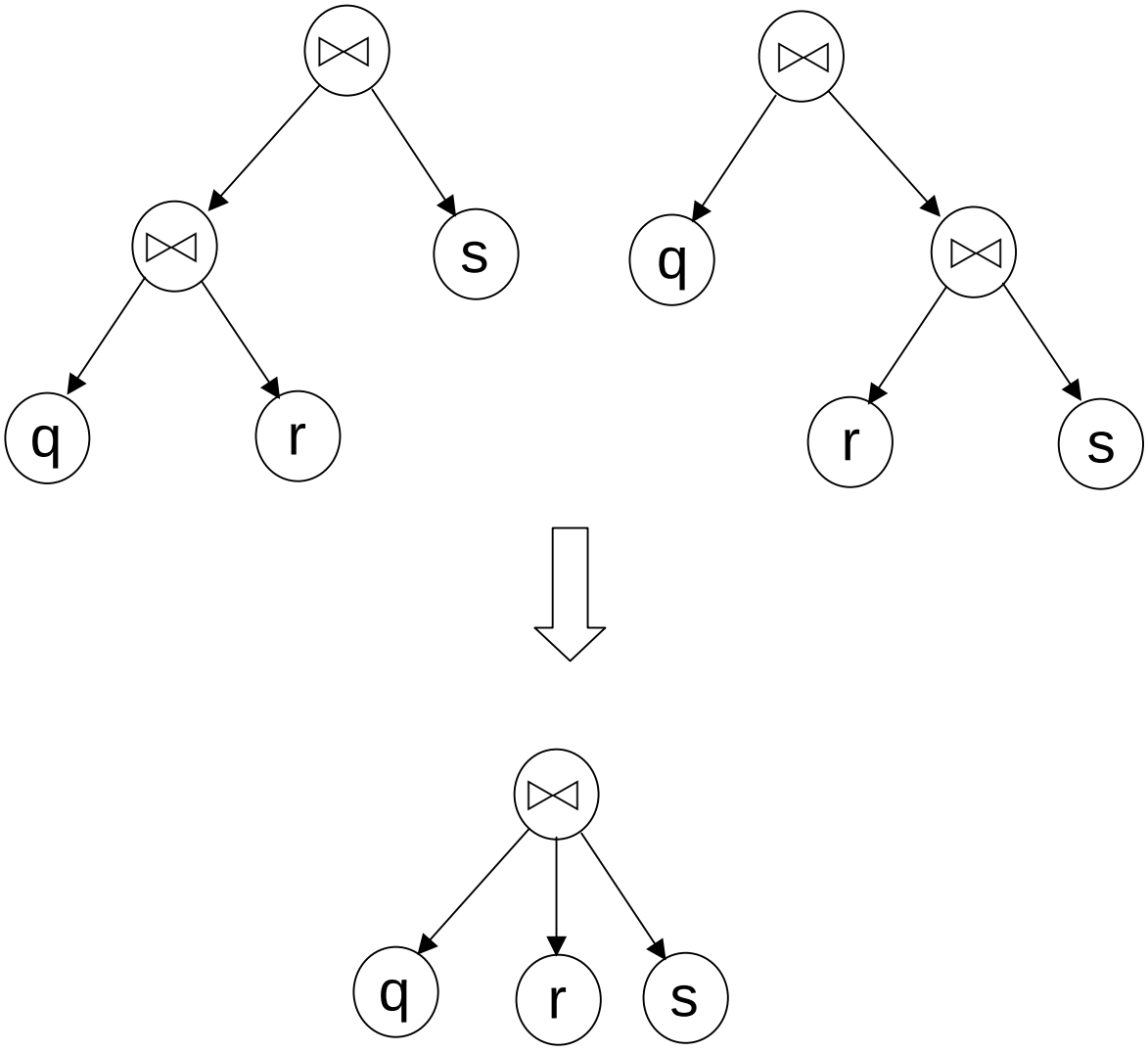
Zeit- und Platzaufwand binärer Operationen steigen mit

- Anzahl der Tupel
- Anzahl der Attribute

in den Argumentrelationen

$(q \bowtie r \bowtie s)$

assoziativ und kommutativ

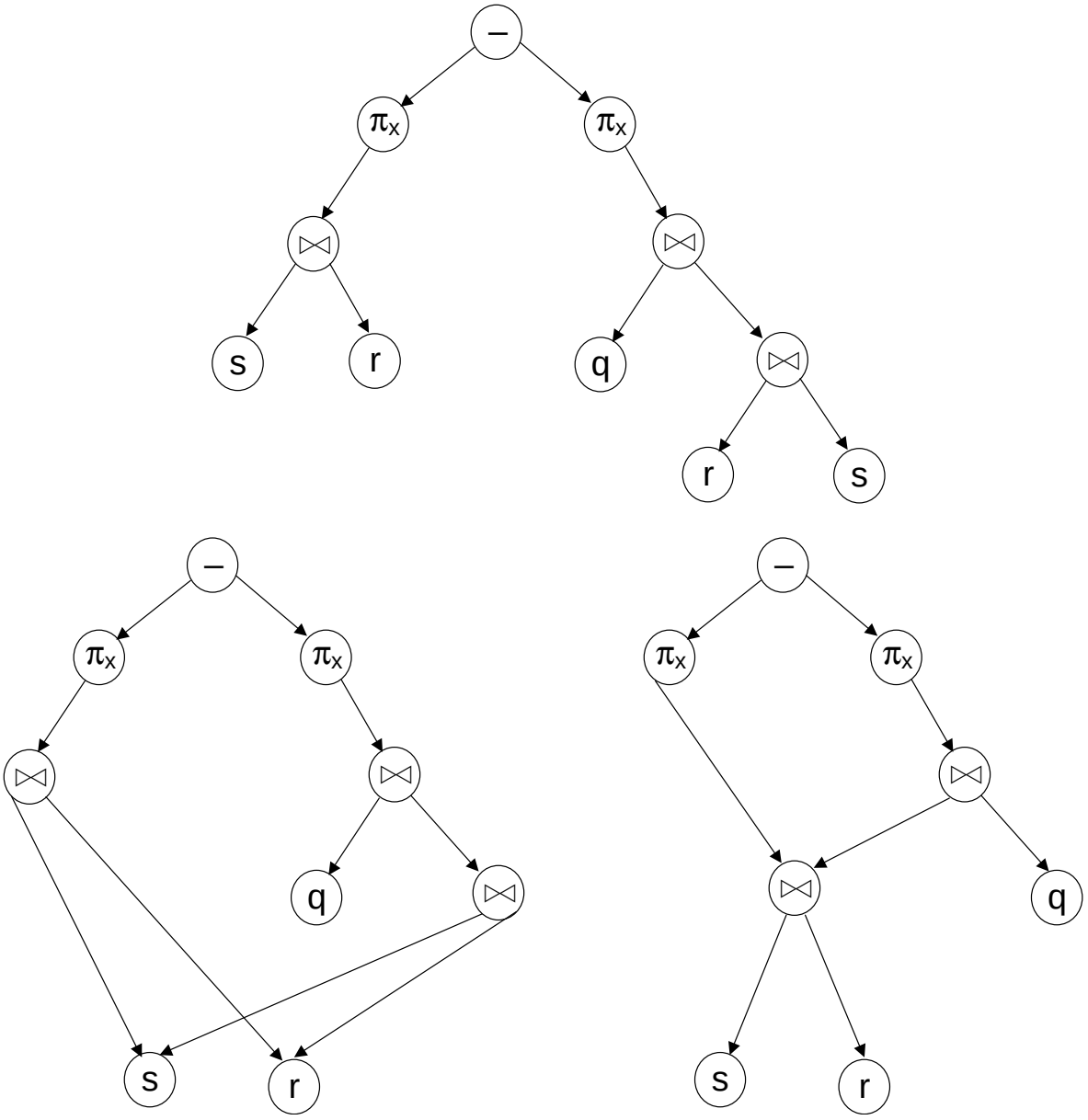


Grundprinzipien:

Es wird versucht, auf diesen Operatorbaum die folgenden drei Grundprinzipien anzuwenden:

- Verlagerung unärer Operatoren in Richtung auf die Blätter des Baumes
  (Ziel: möglichst frühzeitige Reduktion der Größe von Verbund-Operanden)
- Zusammenfassung unärer Operationen
- Da die Projektion Entfernung doppelter Tupel bedingt, sollte man die Selektion möglichst vor der Projektion durchführen (im Allgemeinen ist σ vor π statistisch besser).

Zusammenfassung gleicher Teilausdrücke:



Das Problem bei der Feststellung gemeinsamer Teilausdrücke ist, dass die algebraischen Transformationen deren Existenz verschleiern können.

Umformungsregeln

Eine Reihe von Regeln, die auf der Relationalen Algebra aufbauen, erlauben die Umformung der Anfrage:

Regeln für Verbund und kartesisches Produkt

(1) Kommutativität

$$E_1 \bowtie E_2 \equiv E_2 \bowtie E_1$$

$$E_1 \times E_2 \equiv E_2 \times E_1$$

(2) Assoziativität

$$E_1 \bowtie (E_2 \bowtie E_3) \equiv (E_1 \bowtie E_2) \bowtie E_3 .$$

$$E_1 \times (E_2 \times E_3) \equiv (E_1 \times E_2) \times E_3 .$$

Regeln für Selektion und Projektion

(3) Zusammenfassung von Projektionen

Bedingungen:
$\{A_i \mid i = 1, ..., n\} \subseteq \{B_i \mid i = 1, ..., m\}$.

$$\pi_{A_i,...,A_n}(\pi_{B_1,...,B_m}(E)) \equiv \pi_{A_1,...A_n}(E).$$

(3) Zusammenfassung/Kommutativität von Selektionen:

$$\sigma_{F_1}(\sigma_{F_2}(E)) \equiv \sigma_{F_2 \wedge F_1}(E).$$

$$\sigma_{F_1}(\sigma_{F_2}(E)) \equiv \sigma_{F_2}(\sigma_{F_1}(E)).$$

(5) Kommutativität Selektion-Projektion

Bedingung F bezieht sich nur auf Attribute A_i:

$$\pi_{A_1,...,A_n}(\sigma_F(E)) \equiv \sigma_F(\pi_{A_1,...,A_n}(E)).$$

Bedingung F bezieht sich auf alle Attribute B und möglicherweise auf Attribute A_i:

$$\pi_{A_1,...,A_n}(\sigma_F(E)) \equiv$$
$$\pi_{A_1,...,A_n}(\sigma_F(\pi_{A_1,...,A_n,B_1,...,B_m}(E))).$$

(6) Kommutativität Selektion-Kartesisches Produkt

Bedingung $F = F_1 \wedge F_2$ bezieht sich auf Attribute von E_1 und E_2

F_i bezieht sich nur auf Attribute von E_i

$$\sigma_F(E_1 \times E_2) \equiv \sigma_{F_1}(E_1) \times \sigma_{F_2}(E_2).$$

F_1 bezieht sich nur auf Attribute von E_1, F_2 bezieht sich auf Attribute von E1 und E2

$$\sigma_F(E_1 \times E_2) \equiv \sigma_{F_2}(\sigma_{F_1}(E_1) \times E_2).$$

(7) Kommutativität Selektion-Vereinigung

$$\sigma_F(E_1 \cup E_2) \equiv \sigma_{F_1}(E_1) \cup \sigma_{F_2}(E_2).$$

(8) Kommutativität Selekton-Mengendifferenz

$$\sigma_F(E_1 - E_2) \equiv \sigma_{F_1}(E_1) - \sigma_{F_2}(E_2).$$

(9) Kommutativität Projektion-Kartesisches Produkt

B_i sind Attribute von E_1, C_i sind Attribute von E_2.

$\{A_i \mid i = 1,..,n\} = \{B_i \mid i = 1,...,m\} \cup \{C_i \mid i = 1,...,k\}$

$\pi_{A_1,...,A_n}(E_1 \times E_2) \equiv$
$\pi_{B_1,...,B_m}(E_1) \times \pi_{C_1,...,C_k}(E_2)$.

(10) Kommutativität Projektion-Vereinigung

$\pi_{A_1,...,A_n}(E_1 \cup E_2) \equiv \pi_{A_1,...,A_n}(E_1) \cup$
$\pi_{A_1,...,A_n}(E_2)$.

Der Verbund ist hierbei darstellbar als Kombination von kartesischem Produkt, Projektion und Selektion, deshalb folgen die Regeln für Kommutativität von Selektion und Verbund aus (4), (5) und (6).

Achtung: Es gilt keine Kommutativität zwischen Mengendifferenz und Projektion!

Ein einfacher Optimierungsalgorithmus:

(1) Umformulierung der Anfrage, so daß nur noch Grundoperationen verwendet werden.
(Auflösung der Verbünde und Divisionen)

(2) Zuordnung der Attribute zu den Relationen (Volle Qualifizierung)

(3) Aufstellen des Operatorbaumes

(4) Zerlegung der Selektionen der Art $\sigma_{F_1 \wedge ... \wedge F_n}(E)$ nach Regel (4) in $\sigma_{F_1}(...(\sigma_{F_n}(E))...)$.

(5) Verlagerung der Selektionen soweit wie möglich in Richtung Blätter mit Regeln (4) - (8).
(Selektionen, die sich nur auf ein Attribut und eine Konstante beziehen, können fast immer zu ihrer Relation wandern)

(6) Zusammenfassung aller direkt aufeinanderfolgenden Selektionen zu einer einzigen Selektion

(7) Verlagerung der obersten Projektion mit Regeln (3), (5), (9) und (10) durch den Baum bis hin zu den Blättern. (Bei binären Operationen aufspalten)

(8) Ergebnisangabe als Relationaler Ausdruck.

Beispiele:

BOOKS (TITLE,AUTHOR,PNAME,LC_NO)

PUBLISHERS (PNAME,PADDR,PCITY)

BORROWERS (NAME,ADDR,CITY,CARD_NO)

LOANS (CARD_NO,LC_NO,DATE)

PNAME = publisher’s name

LC_NO = Library of Congress number

PADDR = the street address of a publisher

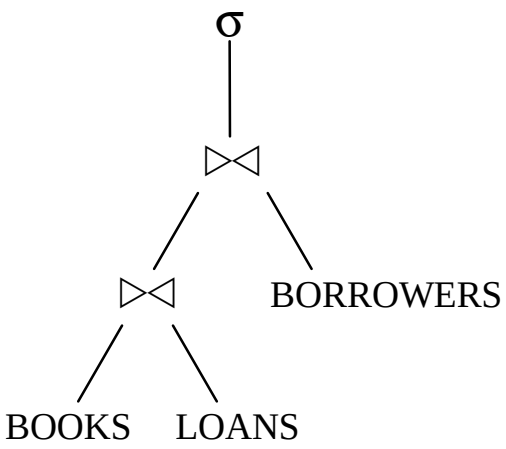
PCITY = the city in which a publisher is located

CARD_NO = the library card number

DATE = the date on which a book was borrowed

Beispiel 1

$$\sigma_{CITY='Frankfurt' \wedge DATE<1.1.1997}(BOOKS \bowtie LOANS \bowtie BORROWERS)$$



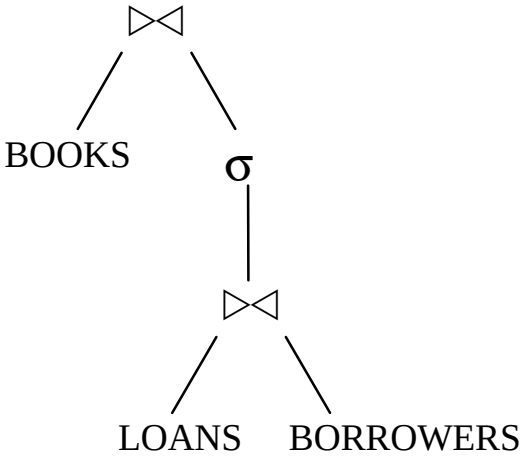
Assoziativität des Verbundes

(2) (6)

⇩

Kommutativität Selektion / kartesisches Produkt

$$BOOKS \bowtie (\sigma_{CITY='Frankfurt' \wedge DATE<1.1.1997}(LOANS \bowtie BORROWERS))$$



Beispiel 1 (Fortsetzung)

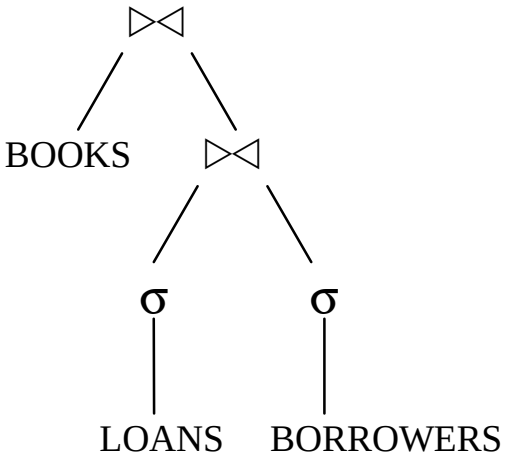
$$\text{BOOKS} \bowtie \left(\sigma_{\text{CITY}=\text{'Frankfurt'} \wedge \text{DATE}<1.1.1997}\left(\text{LOANS} \bowtie \text{BORROWERS}\right)\right)$$

Kommutativität (6)

Selektion / kartesisches Produkt

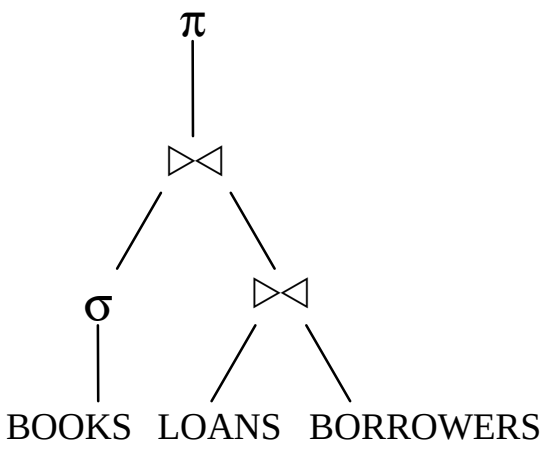
⇩

$$\text{BOOKS} \bowtie \left(\sigma_{\text{DATE}<1.1.1997}\left(\text{LOANS}\right) \bowtie \sigma_{\text{CITY}=\text{'Frankfurt'}}\left(\text{BORROWERS}\right)\right)$$



Beispiel 2

$$\pi_{\text{NAME,CARD\_NO,DATE}}\left(\sigma_{\text{TITLE='DATABASES'}}\left(BOOKS\right)\right.$$
$$\left.\bowtie\left(LOANS \bowtie BORROWERS\right)\right)$$



Assoziativität des Verbundes

(2) (3) (9)

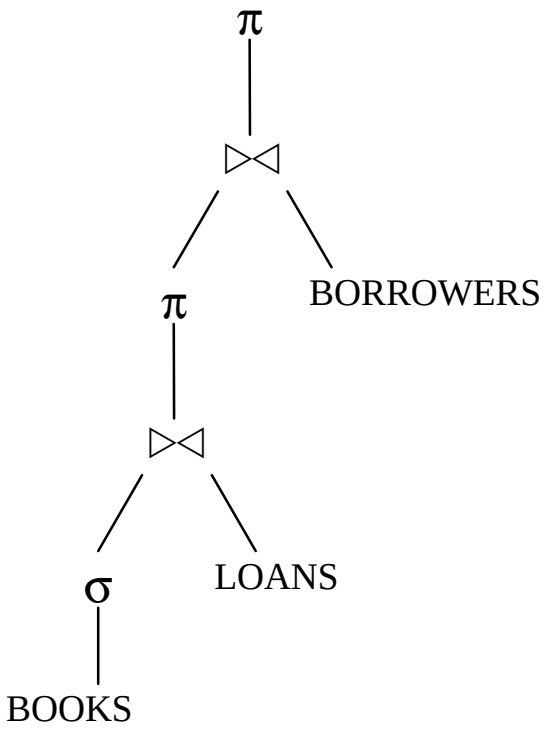
⇩

Kommutativität Selektion (Projektion)/ kartesisches Produkt

$$\pi_{\text{NAME,CARD\_NO,DATE}}($$
$$\pi_{\text{CARD\_NO,DATE}}\left(\sigma_{\text{TITLE='DATABASES'}}\left(BOOKS\right)\bowtie LOANS\right)$$
$$\left.\bowtie BORROWERS\right)$$

Beispiel 2 (Fortsetzung)

$$\pi_{\text{NAME,CARD\_NO,DATE}}(\pi_{\text{CARD\_NO,DATE}}(\sigma_{\text{TITLE='DATABASES'}}(BOOKS) \bowtie LOANS) \bowtie BORROWERS)$$



Beispiel 3 (Ullman)

$$XLOANS = \pi_S(\sigma_F(LOANS \times BORROWERS \times BOOKS))$$

mit:

$$F \equiv BORR.CARD\_NO = LOANS.CARD\_NO \wedge BOOKS.LC\_NO = LOANS.LC\_NO$$

$$S \equiv TITLE, AUTHOR, PNAME, LC\_NO, NAME, ADDR, CITY, CARD\_NO, DATE$$

$$\pi_{TITLE}(\sigma_{DATE<1/1/82}(XLOANS))$$

Schritt 1: Teilen der Selektion F in F1 und F2:

$$F1 \equiv BORR.LC\_NO = LOANS.LC\_NO$$

$$F2 \equiv BOOKS.CARD\_NO = LOANS.CARD\_NO$$

Schritt 2: Selektionen den Baum „hinunterbewegen“

$$\pi_{TITLE}(\pi_S(\sigma_{F2}(\sigma_{F1}(\sigma_{DATE<1/1/82}(LOANS) \times BORR) \times BOOKS)))$$

Schritt 3: Vereinigung der beiden Projektionen zu:

$$\pi_{TITLE}$$

Schritt 4: Aufspalten der Folge

$$\pi_{TITLE}\left(\sigma_{F2}(\ldots)\right)$$

mit Regel 5 in:

$$\pi_{TITLE}\left(\sigma_{F2}\left(\pi_{TITLE,BOOKS.LC\_NO,LOANS.LC\_NO}(\ldots)\right)\right)$$

Dann Aufspalten der zweiten Projektion nach Regel 9:

$$\pi_{TITLE,BOOKS.LC\_NO} \quad \text{und} \quad \pi_{LOANS.LC\_NO}$$

$$\Downarrow$$

$$\pi_{TITLE}\left(\sigma_{F2}\left(\pi_{LOANS.LC\_NO}\left(\sigma_{F1}\left(\sigma_{DATE<1/1/82}(LOANS)\right.\right.\right.\right.$$
$$\left.\left.\left.\left.\times BORR\right)\right)\times\pi_{TITLE,BOOKS.LC\_NO}(Books)\right)\right)$$

Schritt 5: Aufspalten der Folge

$$\pi_{LOANS.LC\_NO}\left(\sigma_{F1}(\ldots)\right)$$

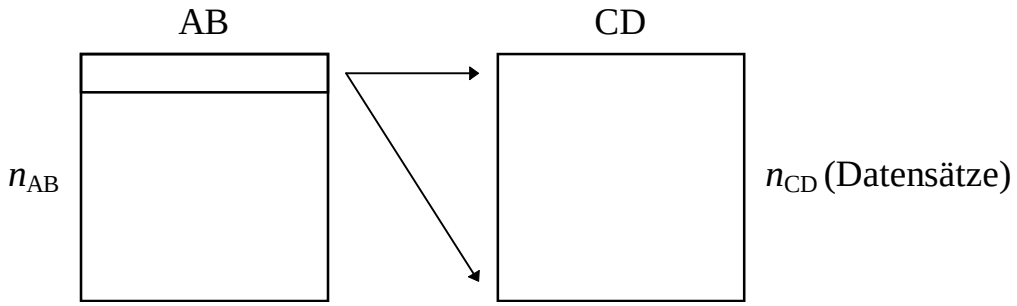
mit Regel 5 in:

$$\pi_{LOANS.LC\_NO}\left(\sigma_{F1}\left(\pi_{LOANS.LC\_NO,BORR.CARD\_NO,LOANS.CARD\_NO}(\ldots)\right)\right)$$

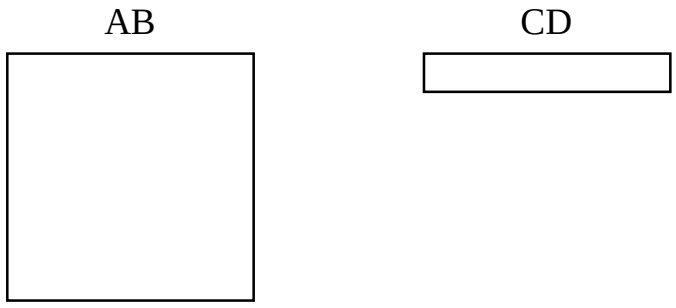
Schritt 6: Die zweite Projektion aufspalten und vor das kartesische Produkt bringen (Regeln 5 und 9):

$$\pi_{TITLE}\left(\sigma_{F2}\left(\pi_{LOANS.LC\_NO}\left(\sigma_{F1}\left(\pi_{LOANS.LC\_NO,LOANS.CARD\_NO}\left(\sigma_{DATE<1/1/82}(LOANS)\right)\times\pi_{BORR.CARD\_NO}(BORR)\right)\right)\times\pi_{TITLE,BOOKS.LC\_NO}(Books)\right)\right)$$

Beispiel: Kartesisches Produkt



Strategie: Lade so viele *Blöcke* von AB wie möglich in den Hauptspeicher und lasse dabei Platz für einen Block von BC.



m Hauptspeicher-Blöcke

- n_{AB}, n_{CD} : Sätze.
- b_{AB}, b_{CD} : Sätze/Block.
- m : Anzahl der Blöcke im Hauptspeicher.

- Anzahl der Block-Zugriffe um AB zu lesen: n_{AB}/b_{AB} .
- CD muss $n_{AB}/(m-1)b_{AB}$ - mal gelesen werden.

  Jedes Mal werden dazu n_{CD}/b_{CD} Zugriffe benötigt.

Anzahl der Block-Zugriffe:

$$\frac{n_{AB}}{b_{AB}} + \frac{n_{AB}}{(m-1)\cdot b_{AB}} \cdot \frac{n_{CD}}{b_{CD}} =$$

$$\frac{n_{AB}}{b_{AB}} \cdot \left(1 + \frac{n_{CD}}{(m-1)b_{CD}}\right)$$

Beispiel:

$$n_{AB} = n_{CD} = 10\,000$$
$$b_{AB} = b_{CD} = 5$$
$$m = 100$$

Anzahl der Zugriffe = 42.400.

Bei 20 Block-Zugriffen pro Sekunde wird dieses kartesische Produkt ca. 35 Minuten benötigen.

Wähle AB als die Relation, mit dem kleineren Quotienten $\frac{n_{AB}}{b_{AB}}$.

Das heißt, die Relation, die in weniger Blocks passt.