OPTIMIERUNG RELATIONALER ABFRAGEN

Dr. Sándor Gajdos

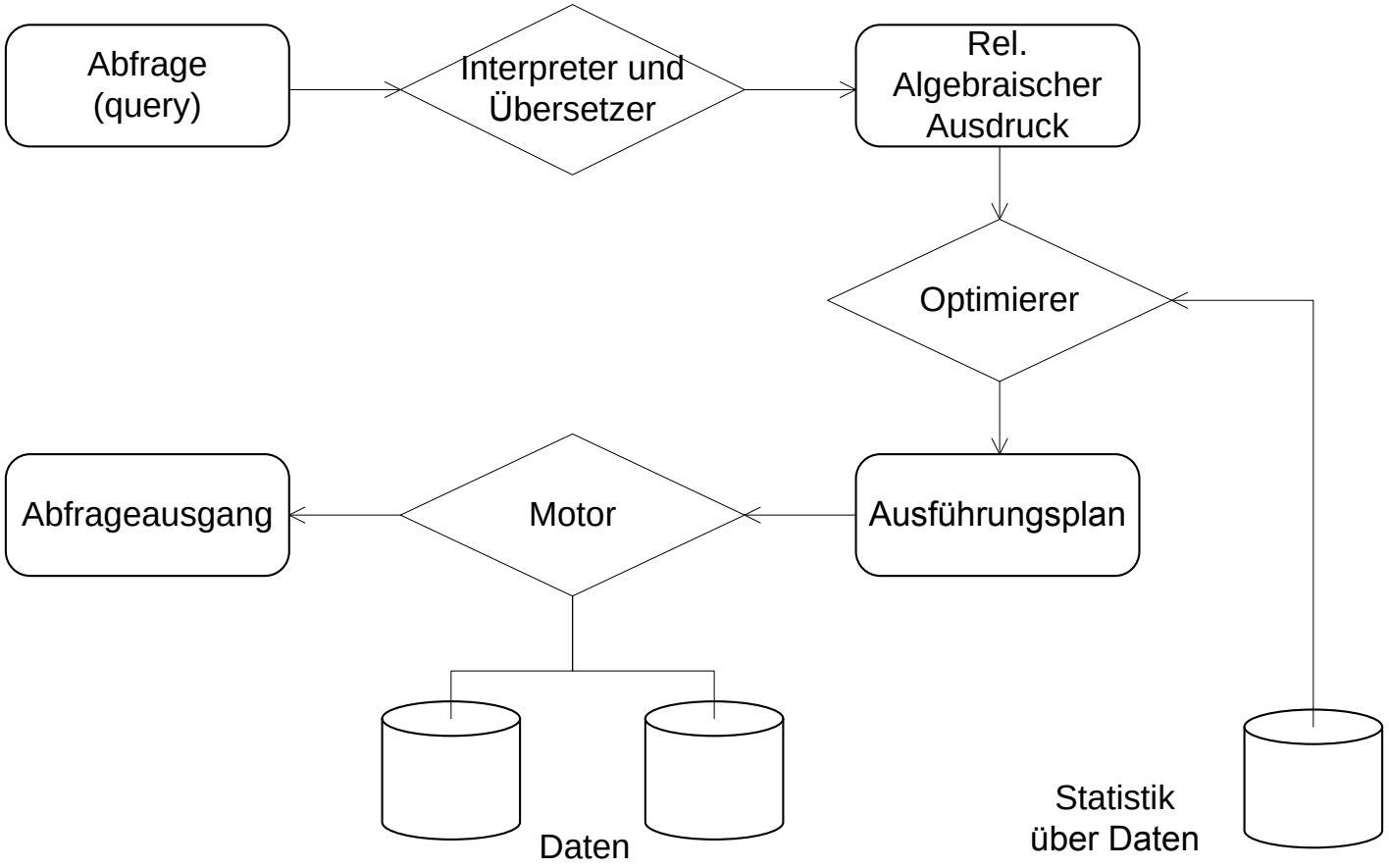
Dez. 2016.

BME–TMIT

INHALT

- Heuristische, regelbasierte Optimierung
- Kostenbasierte Optimierung
  - Kostenschätzung mit Hilfe von Kataloge
  - Durchblick der Operationen
  - Bewertung der Ausdücke
  - Die Selektion des optimalen Ausführungsplanes
- Manuelle vs. automatische Optimierung

ÜBERSICHTSDIAGRAMM



I. HEURISTISCHE, REGELBASIERTE OPTIMIERUNG

- Relationenalgebraischer Baum basierte Optimierung
- Abfragebaum, Beispiel dazu:

EMPLOYEE (EMPLOYEE_ID, LAST_NAME, FIRST_NAME, BIRTH_DATE, …)

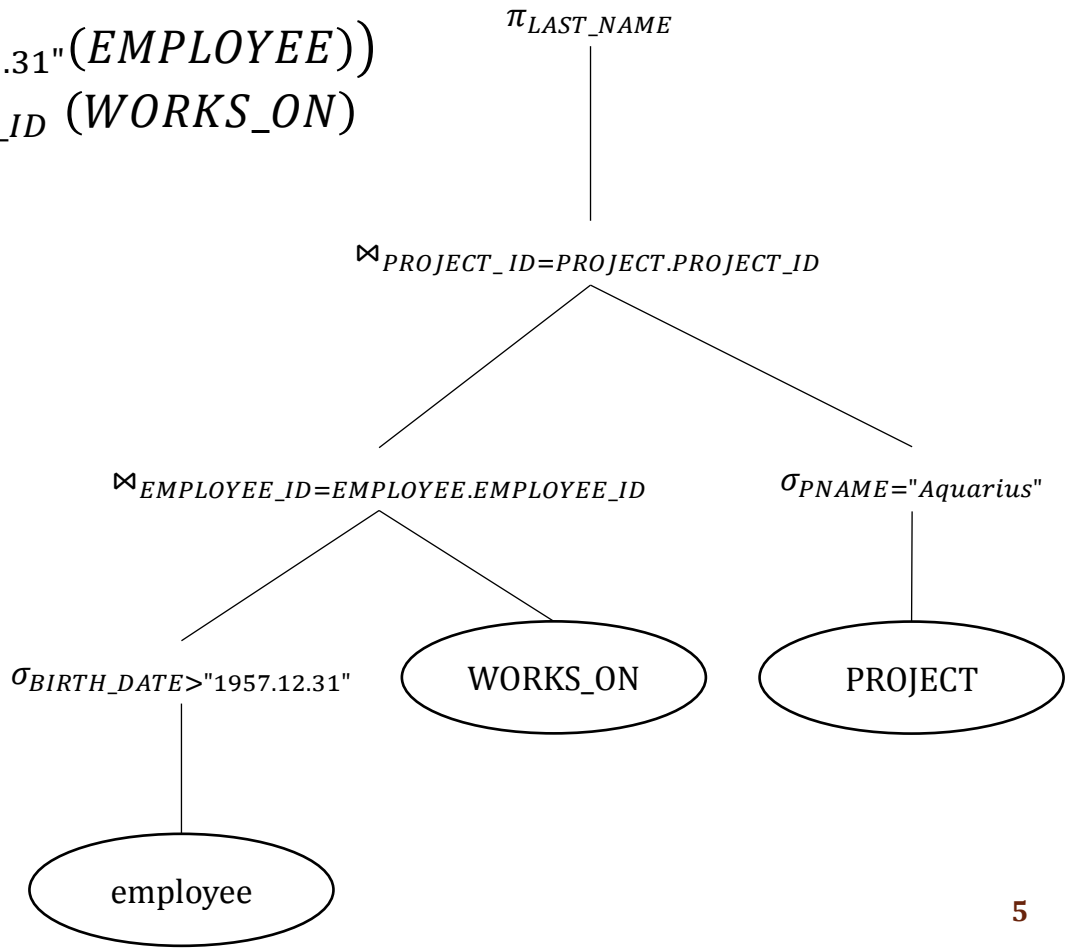
PROJECT (PROJECT_ID, PNAME, …)

WORKS_ON (PROJECT_ID, employee_ID)

```
select last_name
  from employee, works_on, project
 where employee.birth_date > '1957.12.31'
   and works_on.project_id = project.project_id
   and works_on.employee_id = employee.employee_id
   and project.pname = 'Aquarius'
```

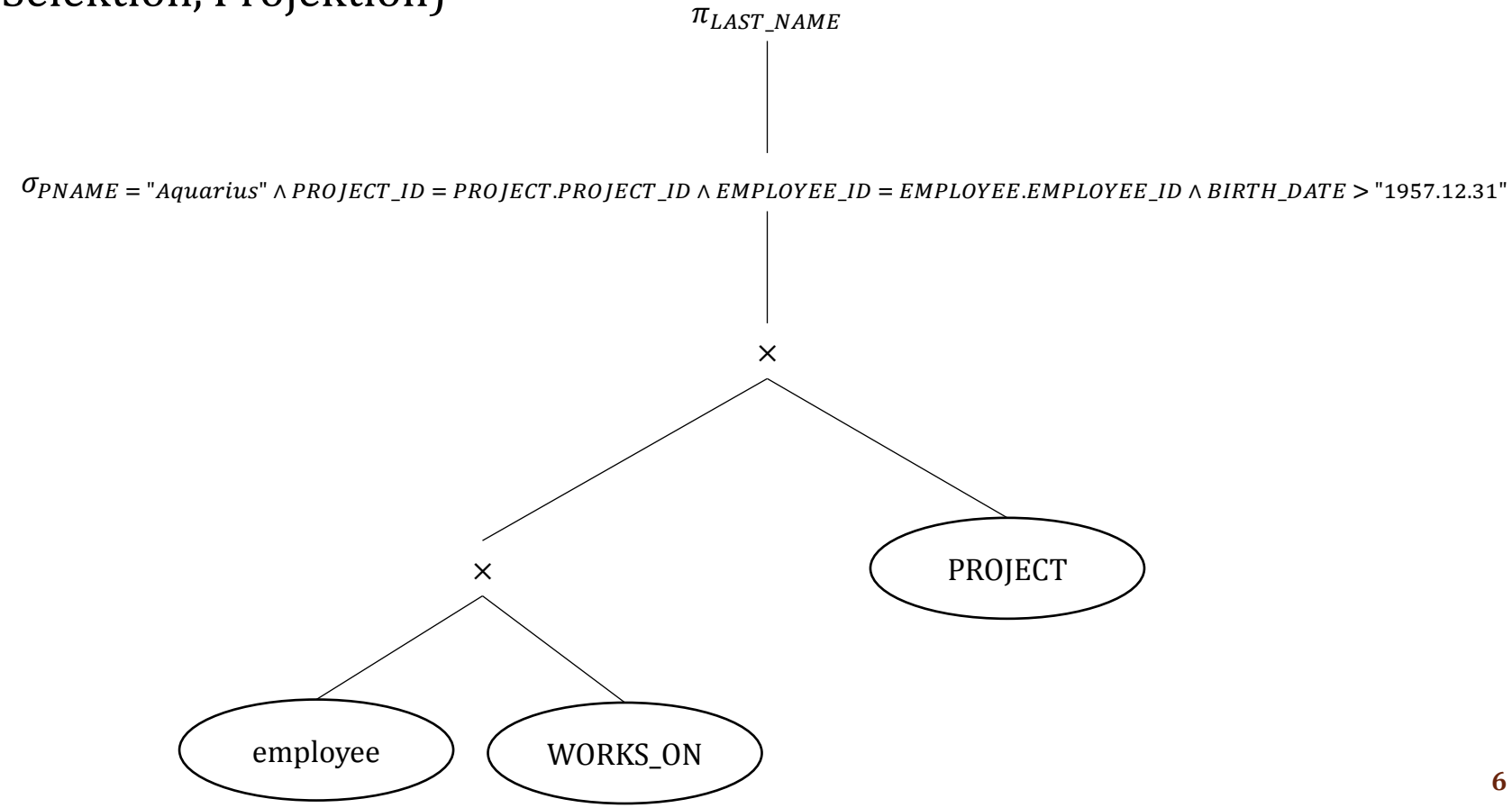
RELATIONENALGEBRAISCHER AUSDRUCK – EINE MÖGLICHKEIT

$$\pi_{LAST\_NAME}\left(\left(\sigma_{BIRTH\_DATE>"1957.12.31"}(EMPLOYEE)\right)\right.$$
$$\bowtie_{EMPLOYEE\_ID=EMPLOYEE.EMPLOYEE\_ID}\ (WORKS\_ON)$$
$$\left.\bowtie_{PROJECT\_ID=PROJECT.PROJECT\_ID}\sigma_{PNAME="Aquarius"}(PROJECT)\right)$$

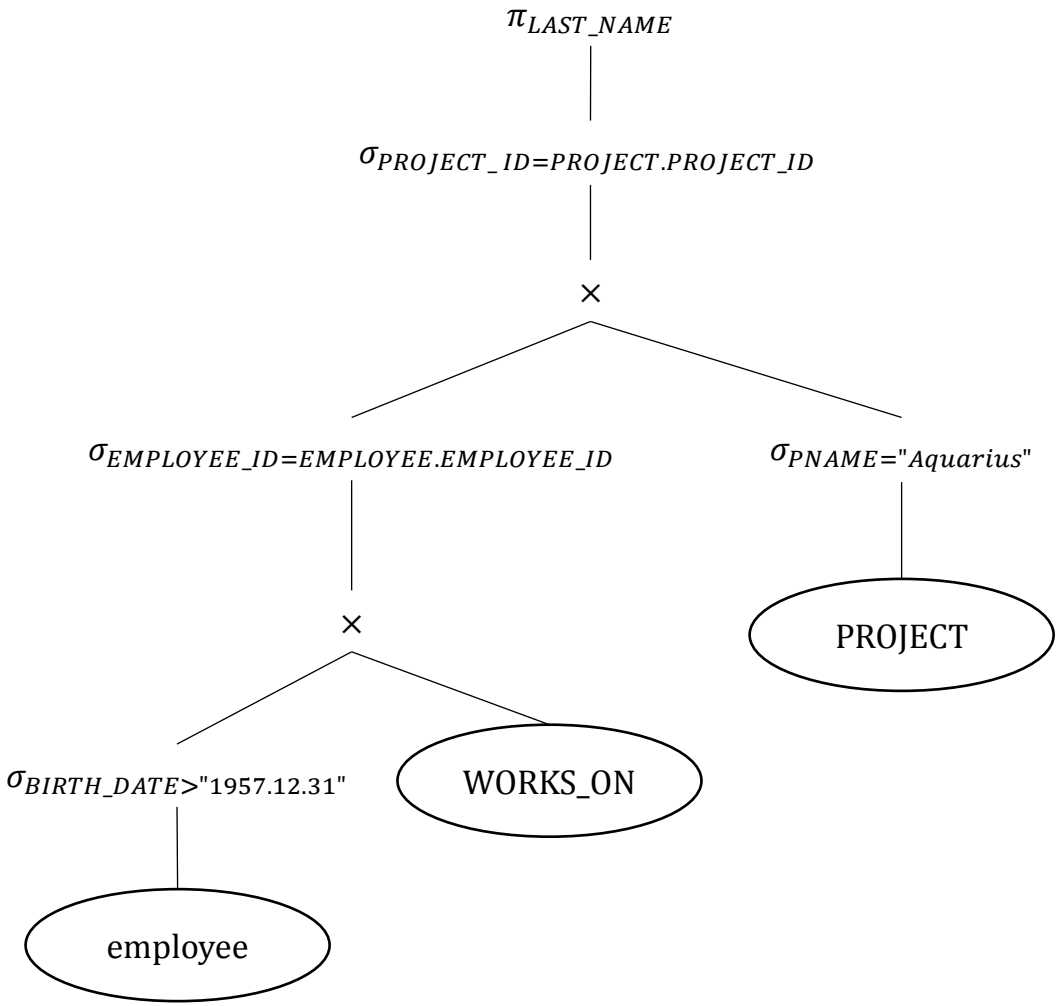


DER ZIEL: DIE SCHNELLSTE FORM ZU BESTIMMEN

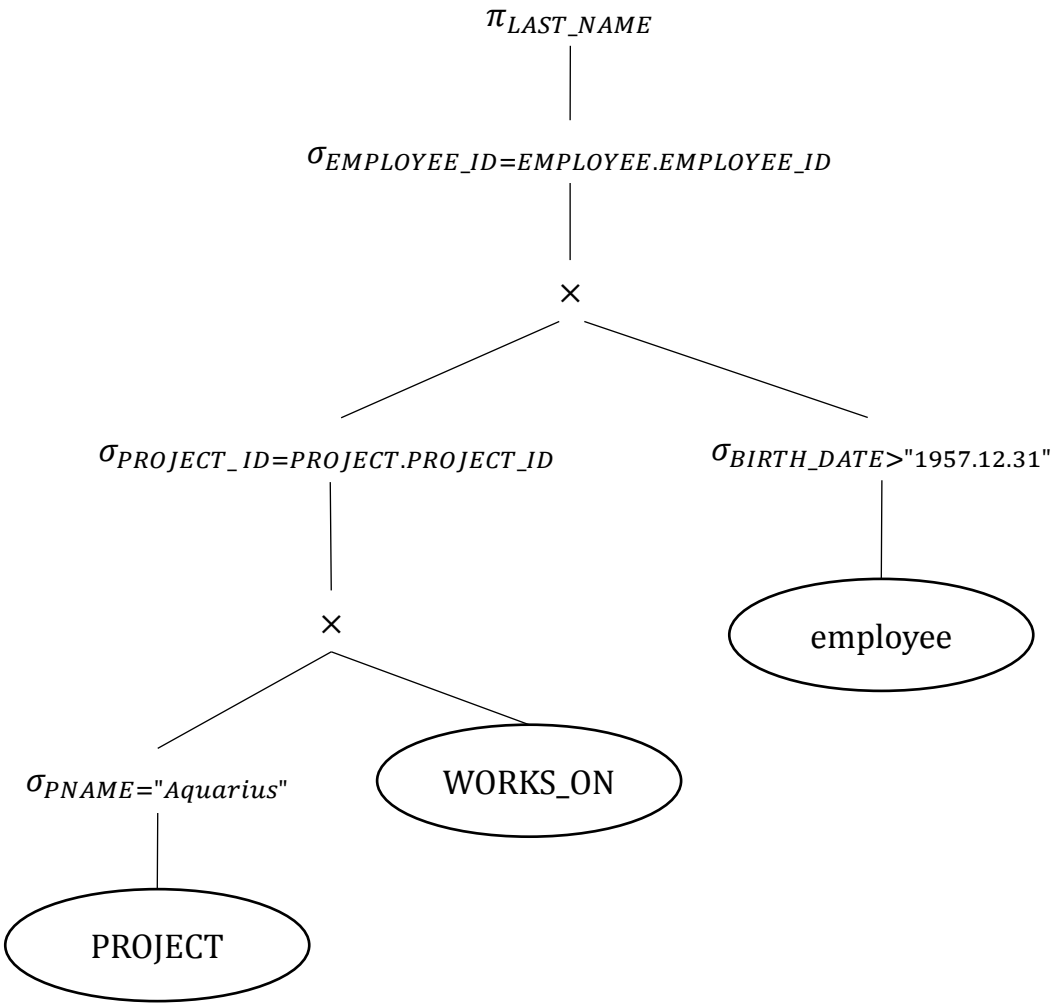
Ausgangspunkt: die kanonische Form (Cartesisches Produkt, Selektion, Projektion)



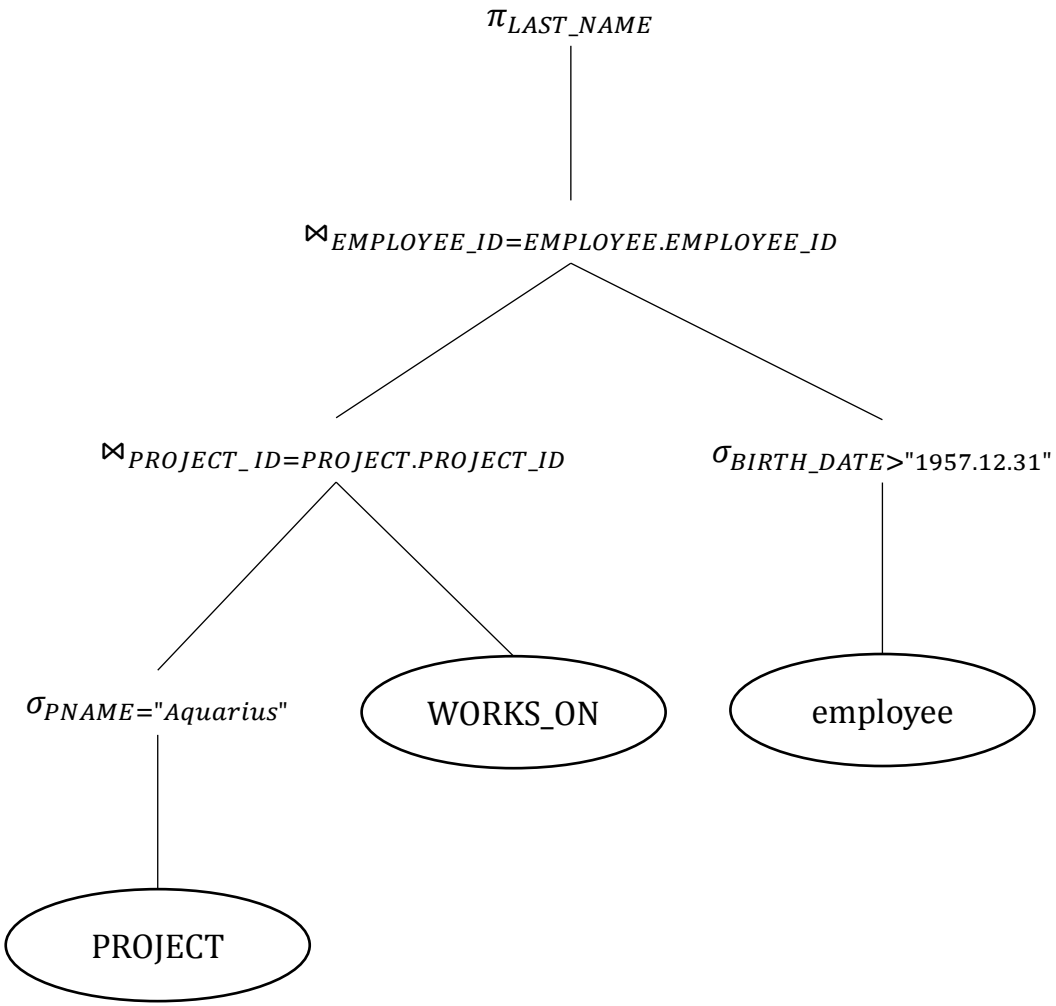
ZWEITER SCHRITT: SENKEN DIE SELEKTIONEN



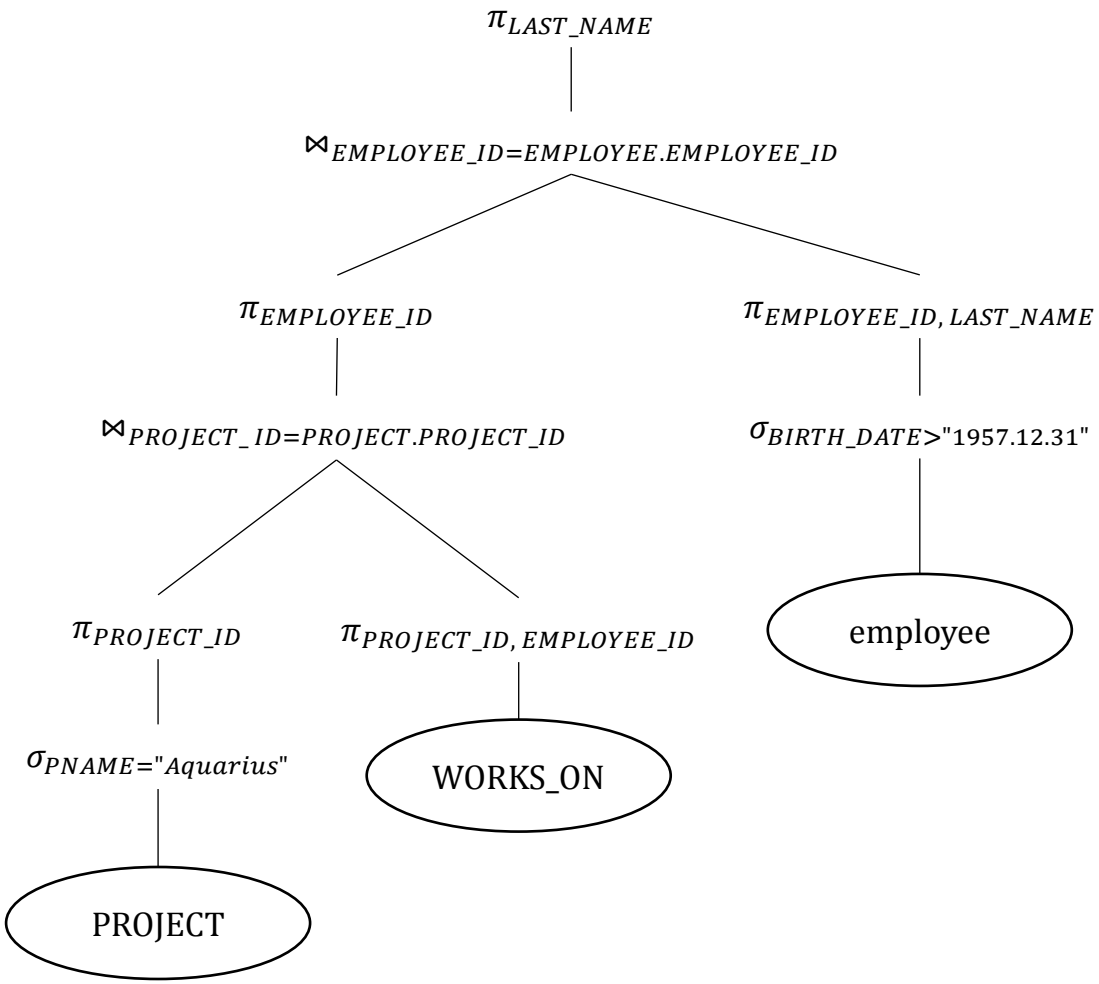
DRITTER SCHRITT: UMORDNEN DIE BLÄTTER



VIERTER SCHRITT: VERBUND EINFÜHREN



FÜNFTER SCHRITT: SENKEN DIE PROJEKTIONEN



WANN SIND ZWEI BÄUME ÄQUIVALENT?

RELATIONENALGEBRAISCHE TRANSFORMATIONEN I.

- $\sigma_{c_1 \wedge c_2 \wedge \cdots \wedge c_n}(r) \equiv \sigma_{c_1}\left(\sigma_{c_2}\left(\ldots\left(\sigma_{c_n}(r)\right)\ldots\right)\right)$
- $\sigma_{c_1}\left(\sigma_{c_2}(r)\right) \equiv \sigma_{c_2}\left(\sigma_{c_1}(r)\right)$
- $\pi_{List_1}\left(\pi_{List_2}\left(\ldots\left(\pi_{List_n}(r)\right)\ldots\right)\right) \equiv \pi_{List_1}(r)$
- $\pi_{A_1,A_2,\ldots,A_n}(\sigma_c(r)) \equiv \sigma_c\left(\pi_{A_1,A_2,\ldots,A_n}(r)\right)$

WANN SIND ZWEI BÄUME ÄQUIVALENT?

RELATIONENALGEBRAISCHE TRANSFORMATIONEN II.

- $r \bowtie_c s \equiv s \bowtie_c r$
- $\sigma_c(r \bowtie s) \equiv (\sigma_c(r)) \bowtie s$
- $\pi_L(r \bowtie_c s) \equiv \left(\pi_{A_1,\ldots,A_n}(r)\right) \bowtie_c \left(\pi_{B_1,\ldots,B_m}(s)\right)$
- $\pi_L(r \bowtie_c s) \equiv$
  $\pi_L\left(\left(\pi_{A_1,\ldots,A_n,A_{n+1},\ldots,A_{n+k}}(r)\right) \bowtie_c \left(\pi_{B_1,\ldots,B_m,B_{m+1},\ldots,B_{m+p}}(s)\right)\right)$

Die Mengenoperationen (Union, Durchschnitt) sind kommutativ.

Der Verbund, Cartesisches Produkt, Union und Durchschnitt sind assoziativ:

$$(r\theta s)\theta t \equiv r\theta(s\theta t)$$

WANN SIND ZWEI BÄUME ÄQUIVALENT?

RELATIONENALGEBRAISCHE TRANSFORMATIONEN III.

- $\sigma_C(r\ \theta\ s) \equiv (\sigma_C(r))\ \theta\ (\sigma_C(s))$
- $\pi_L(r\ \theta\ s) \equiv (\pi_L(r))\ \theta\ (\pi_L(s))$

Weitere Regel:

- $c \equiv \neg(c_1 \wedge c_2) \equiv (\neg c_1) \vee (\neg c_2)$
- $c \equiv \neg(c_1 \vee c_2) \equiv (\neg c_1) \wedge (\neg c_2)$

ZUSAMMENFASSENDE REGEL

- Die konjunktive Selektionsbedingungen müssen in eine Reihe von Selektionsbedingungen umgewandelt werden.
- Die Selektionen müssen mit anderen Operationen vertauscht werden.
- Die Blätter des Baumes müssen umgeordnet werden.
- Die Cartesische Produkte und die darüber stehende Selektionen müssen in einen Verbund zusammengezogen werden.
- Die Projektionen müssen mit den anderen Operationen vertauschen.

II. KOSTENBASIERTE OPTIMIERUNG

1. Syntaxanalyse, Überstzung
2. Kostenoptimierung
3. Auswertung

II. KOSTENBASIERTE OPTIMIERUNG

- Kostenschätzung mit Hilfe von Kataloge
  - Kataloginformationen über Relationen
  - Kataloginformationen über Indexe
  - Die Kosten der Abfrage
- Lösung für die Aktualisierung der Katalogdaten

KATALOGINFORMATIONEN ÜBER RELATIONEN

- n_r: die **N**ummer der Sätze in der Relation $\boldsymbol{r}$
- b_r: die Nummer der **B**löcke in der Relation $\boldsymbol{c}$
- s_r: Satzlänge in Relation $\boldsymbol{r}$ (**s**ize) in Bytes
- f_r: Die Nummer der Sätze in einem Block für Relation $\boldsymbol{r}$ (blocking **f**actor)

KATALOGINFORMATIONEN ÜBER RELATIONEN

- $V(A, r)$: wieviel verschiedene Werte (**V**alues) hat das Attribut $\boldsymbol{A}$ in Relation $\boldsymbol{r}$ (Kardinalität).
  - $V(A, r) = |\pi_A(r)|$
  - Falls A ein Schlüssel ist, dann $V(A, r) = n_r$
- $SC(A, r)$: (**S**election **C**ardinality) die durchschnittliche Nummer der Sätze, die eine Selektionbedingung befriedigen.
  - Falls A ein Schlüssel ist, dann $SC(A, r) = 1$
  - In der Regel: $SC(A, r) = \frac{n_r}{V(A,r)}$
- Wenn die Sätze der Relation zusammen gespeichert sind, dann:

$$b_r = \left\lceil \frac{n_r}{f_r} \right\rceil$$

KATALOGINFORMATIONEN ÜBER INDEXE

- f_i: die durchschnittliche Nummer der Zeiger einer Knote bei baumformigen Indexe, z.B. bei B*-Bäume
- HT_i: die Nummer der Ebenen (**H**eight of **T**ree)
  - $HT_i = \lceil \log_{f_i} V(A, r) \rceil$ (B*-Baum)
  - $HT_i = 1$ (Hash)
- LB_i: die Nummer der Blöcke auf dem niedrigsten Ebene (Blätterebene) (**L**owest level index **B**locks)

DIE SCHÄTZUNG DER KOSTEN

Die Kostenschätzung kann basieren auf:

- Ressourcen, die verlangt oder benutzt werden?
- Ansprechzeit?
- Kommunikationszeit?

Definiton:

- Die Nummer der Blocklese- oder schreibeoperationen zu Hintergrundspeicher, ohne die Kosten das Ergebnisauschreiben

Weitere Vereinfachungen.

DIE KOSTEN DER EINZELNEN OPERATIONEN

- Selection
  - Verschiedene Algorithmen (Basis-, indexierte-, vergleichungsbasierte)
  - Komplexe Selektion
- Verbund
  - Typen
  - Verbund Größeschätzung
  - Verbundalgorithmen
  - Komplexer Verbund
- Sonstige
  - Filtrierung der Wiederholunge
  - Union, Durchschnitt, Differenz

BASISALGORITHMEN FÜR SELEKTION (=)

A1: Lineares Suchen

- Kosten:

$$E_{A1} = b_r$$

A2: Binäres Suchen

- Bedingung:
  - Die Blöcke sind nach Attribut A geordnet
  - Die Selektionbedingung ist die Gleichheit mit Attribut A
- Kosten:

$$E_{A2} = \lceil \log_2(b_r + 1) \rceil + \left\lceil \frac{SC(A, r)}{f_r} \right\rceil - 1$$

INDEXIERTE ALGORITHMEN FÜR SELEKTION

A3: Mit Hilfe von Primärindexe, Gleichheitsbedingung wird auf Schlüsselwert definiert

- $E_{A3} = HT_i + 1$

A4: Mit Hilfe von Primärindexe, Gleichheitsbedingung wird auf Nicht-Schlüsselwert definiert (Primärindex wird auf Nicht-Schlüsselattribut gebaut)

- $E_{A4} = HT_i + \left\lceil \frac{SC(A,r)}{f_r} \right\rceil$

A5: Mit Hilfe von Sekundärindexe.

- $E_{A5} = HT_i + SC(A,r)$
- $E_{A5} = HT_i + 1$, Falls A ein Schlüssel ist.

VERGLEICHUNGSBASIERTE SELEKTION– $\sigma_{A \leq v}(R)$

Die Schätzung der Nummer der Ergebnissätze:

- Falls v unbekannt ist: $\frac{n_r}{2}$

- Falls v bekannt ist, angenommen eine gleichmäßige Verteilung:

$$n_{\text{durchschnittlich}} = n_r \cdot \frac{v - \min(A, r)}{\max(A, r) - \min(A, r)}$$

VERGLEICHUNGSBASIERTE SELEKTION– $\sigma_{A \leq v}(R)$

A6: Mit Hilfe von Primärindexe.

- Falls v unbekannt ist:

$$E_{A6} = HT_i + \frac{b_r}{2}$$

- Falls v bekannt ist :

$$E_{A6} = HT_i + \left\lceil \frac{c}{f_r} \right\rceil,$$

wobei c bezeichnet die Nummer der Sätze, wofür $A \leq v$

A7: Mit Hilfe von Sekundärindexe

$$E_{A7} = HT_i + \frac{LB_i}{2} + \frac{n_r}{2}$$

VERBUNDOPERATIONEN

Definiton:

$$r_1 \bowtie_\theta r_2 = \sigma_\theta(r_1 \times r_2)$$

Verbundtypen:

- Natürlicher Verbund (natural join)

$$r_1 \bowtie r_2 = \pi_{A \cup B}(\sigma_{R1.X=R2.X}(r_1 \times r_2))$$

- Äußerer Verbund (outer join)
  - Äußerer linker Verbund: $r_1 * (+) r_2$
  - Äußerer rechter Verbund: $r_1 (+) * r_2$
  - Voller äußerer Verbund: $r_1 (+) * (+) r_2$
- Theta Verbund:

$$r_1 \bowtie_\theta r_2 = \sigma_\theta(r_1 \times r_2)$$

VERBUND DURCH VERSCHACHTELTEN SCHLEIFEN

Zwei Relationen werden gegeben, r und s:

FOR jede $t_r \in r$ Sätze DO BEGIN

FOR jede $t_s \in s$ Sätze DO BEGIN

(t_r, t_s) muss getestet werden ob sie die θ Verknüpfungsbedingung erfüllen

IF ja, THEN der Satz $t_r . t_s$ muss Teil des Ergebnisses werden

END

END

- „worst case" Kosten: $n_r \cdot b_s + b_r$
- Fall mindestens einer der Relationen kann vollstandig in den Hauptspeicher gebracht werden, dann die Kosten sind: $b_r + b_s$

VERBUND DURCH BLOCK-VERSCHACHTELTEN SCHLEIFEN

FOR jede $b_r \in r$ Blöcke DO BEGIN

FOR jede $b_s \in s$ Blöcke DO BEGIN

FOR jede $t_r \in b_r$ Sätze DO BEGIN

FOR jede $t_s \in b_s$ Sätze DO BEGIN

(t_r, t_s) muss getestet werden ob sie die θ Verknüpfungsbedingung erfüllen

END

END

END

END

- „worst-case" Kosten: $b_r \cdot b_s + b_r$
- Mit viel Haptspeicherplatz: $b_r + b_s$

INDEXIERTER VERBUND DURCH VERSCHACHTELTEN SCHLEIFEN

Es gibt einen Index für eine der Relationen (s)

Die indexierte Relation wird in die innere Schleife des ersten Algorithmes gelegt:

$\Longrightarrow$ Das Suchen kann mit dem gegebenen Index auf niedrigem Kostenniveau durchgeführt werden.

Kosten:

$$b_r + n_r \cdot c,$$

wo c ist die Kosten der Selektion bei Relation s.

WEITERE VERBUND-ALGORITHMEN

- sorted merge join
  - Die Relationen müssen zuerst nach bestimmten Attributen (gegeben in der Verbundbedingung) ordnen, dann die Relationen können einfach zusammengelegt werden.
- hash join
  - Die Sätze einer der Relationen werden durch eine Hash-Tabelle erreicht, solange wir suchen die anpassende Sätze zu den Sätzen der anderen Relation
- sonstige
  - z.B. mit bitmap Indexe (bitmap join)

WEITERE OPERATIONEN

- *Filtrierung der wiederholende Sätze* (Ordnen, dann Löschen)
- *Projektion* (Projektion, dann Filtrierung der wiederholende Sätze)
- *Union* (beide Relationen müssen ordnen, dann die Duplikaten können beim Zusammenlegung gelöscht werden.)
- *Durchschnitt* (beide Relationen müssen ordnen, dann beim Zusammenlegung werden nur die Sätze bewahrt, die in beiden Relationen vorhanden waren)
- *Differenz* (beide Relationen müssen ordnen, dann beim Zusammenlegung werden nur die Sätze bewahrt die zur ersten Relation gehören)
- *Aggregierung* z.B.

$$_{\text{Markename}}G_{\textbf{sum}(\text{Saldo})}\,(\text{Rechnung})$$

Die Relation Rechnung muss nach Markename ordnen, Dann Saldo kann on-the-fly gebildet weden.

MÖGLICHKEITEN DIE AUSDRÜCKE ZU BEWERTEN

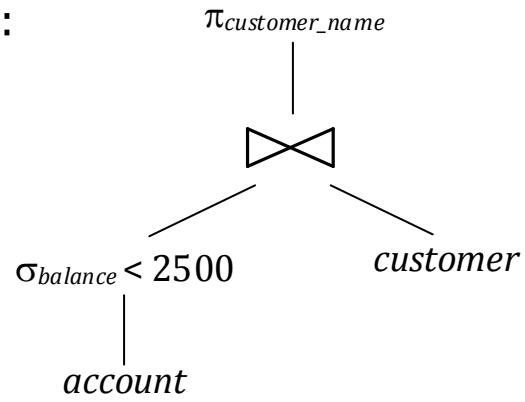
- Materialisierung
  - Nur eine der Operationen des zusammengesetzten Ausdruckes wird gleichzeitig nach einem bestimmten Reihenfolge bewertet, und das Ergebnis wird auf Platte gespeichert.
- Pipelining
  - Mehrere Operationen des zusammengesetzten Ausdruckes können gleichzeitig bewertet werden
  - Die folgende Operation bekommt sofort das Ergebnis der vorigen Operation

MATERIALISIERUNG

- Kanonische Form:

$$\pi_{customer\_name}(\sigma_{balance<2500}(account) \bowtie customer)$$

- Baum der Operationen:



- Gesamtkosten: die Kosten der durchgeführten Operationen + die Kosten der Teilergebnis-abspeicherungen
- Vorteil: einfache Implementierung
- Nachteil: viele Hintergrund-operationen

PIPELINING

- Simultane Bewertung der Teiloperationen
- Die Teile stellen Teilergebnisse für die Teile hinter ihnen aus der Ergebnisse des Teiles vor ihnen her
- Die ganze Relation muß in voraus nicht hergestellt werden.

Vorteile:

- Die Teilergebnisse müssen provisorisch nicht gespeichert werden
- Niedrige Speicherplatzbedarf

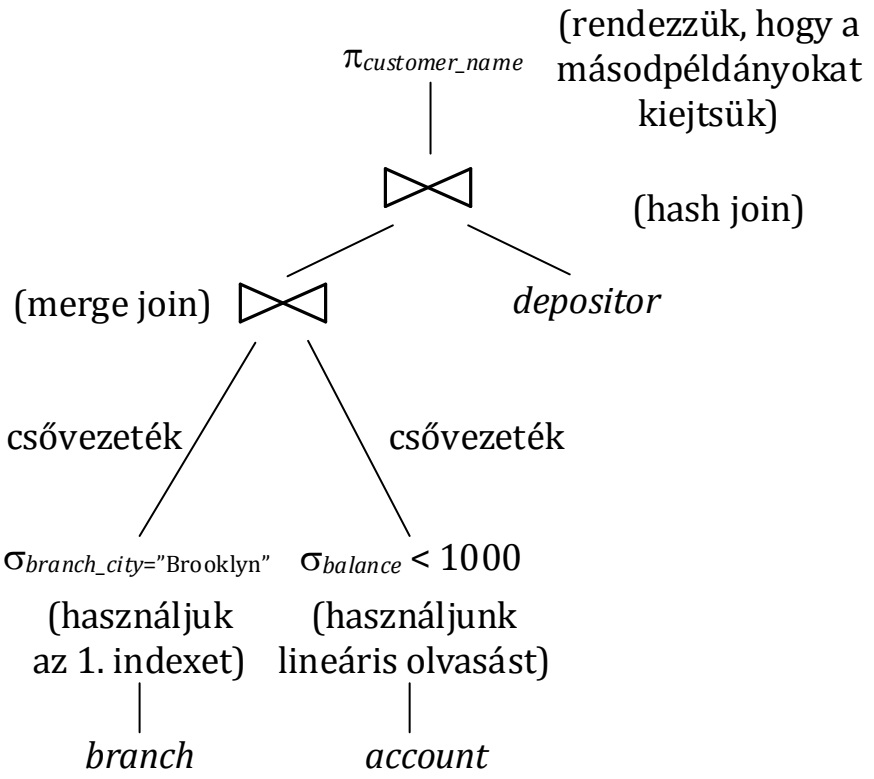
Nachteil:

- Nur gewisse Algorithmen können verwendet werden

DIE SELEKTION DES OPTIMALEN AUSFÜHRUNGSPLANES

- Welche Operationen
- In welchen Reihenfolge
- Mit welchem Algorithmus
- Nach welchem Arbeitsablauf

Ein bestimmter Ausführungsplan



KOSTENBASIERTE OPTIMIERUNG

Gierige und gleichzeitig schlechte Strategie:

- Alle äquivalente Ausdrücke bestimmen
- Alle diese Ausdrücke bewerten
- Den Ausdruck von niedrigstem Kosten wählen

z.B.: $r_1 \bowtie r_2 \bowtie r_3 \rightarrow$ hat 12 äquivalente Ausdrücke

Im Allgemeinen: n Relationen zu verbinden gibt es $\frac{(2(n-1))!}{(n-1)!}$ äquivalente Möglichkeiten.

Es wäre eine zu große Last für das System.

Die Lösung: heuristische kostenbasierte Optimierung

AUTOMATISCHE VS. MANUELLE OPTIMIERUNG

Vorteile der automatischen Optimierung:

- Mehrere Kentnisse/Informationen über gespeicherte Daten.
- Schnellere numerische Bewertungsmögichkeit
- Systematische Bewertung
- Der Algorithmus enthält die Erfahrungen von mehreren Spezialisten.
- Die Bewertung kann vor jedem Programmablauf durchgeführt werden, mit Hilfe der neuesten Informationen/Bedingungen.

Vorteile der manuellen (menschlichen) Optimierung :

- Sogar semantische Kenntnisse können verwendet werden.
- Es gibt eine größere Freiheit die Methoden, Werkzeuge zu wählen.
- Unerwartende Situationen können besser behandelt werden.