Overview

SELECT A.a1, B.a2, B.a3, D.a4
FROM A
JOIN B ON A.id = B.a_id
JOIN C ON B.id = C.b_id
WHERE A.a1 = 42
AND B.a3 = 12

Query Parser

Catalog

Query Optimizer

SHJ

iseek(a1=42, A)

sort bi

filter(a3=12)

scan(C)

isam(id, B)

Precompiled Modules

Query Plan Interpreter

Code Generation

Code Execution

declarative query

physical query execution plan (QEP)

how to execute?
The Story so far:

```
SELECT A.a1, B.a2, B.a3, D.a4
FROM A JOIN B ON A.id = B.a_id
    JOIN C ON B.id = C.b_id
WHERE A.a1 = 42
    AND B.a3 = 12
```
How to Proceed?

iseek(a1=42, A) → sort_{b_id} → filter_{a3=12} → SHJ → \text{how to execute?} → \text{isam}(id, B) \text{ and scan}(C)
Approach 1: Function Library

\[
\text{SHJ( } \text{iseek(a1=42, A), MJ_{C.b_{id}=B.id}( sort_{b_{id}}( \text{scan(C) } ), \text{filter}_{a3=12}( \text{isam(id, B) } ) ) ) } \]

\[
\text{iseek(a1=42, A) } \text{ sort}_{b_{id}} \text{ filter}_{a3=12} \text{ scan(C) isam(id, B) } \]

\[
\text{how to execute?}
\]
Problem: Intermediate Results

Diagram:
- SHJ(.,.)
  - 10 GB
  - 1 TB
  - M_{JC_{b_id=B_{id}(.,.)}}
    - 1 GB
    - 1 TB
    - isam(id, B)
      - 1 TB
      - isam(C)
      - 1 TB
    - scan(C)
      - 1 TB
      - 1 TB
    - sort_{b_id(.)}
      - 1 TB
      - 1 TB
    - isek(a1=42, A)
      - 1 TB
      - 1 TB
Approach 2: Pipelining

\[ \text{iseek(a1=42, A)} \rightarrow \text{sort}_{b \_id} \rightarrow \text{filter}_{a3=12} \rightarrow \text{SHJ} \]

\[ \text{iseek(a1=42, A)} \rightarrow \text{sort}_{b \_id} \rightarrow \text{filter}_{a3=12} \rightarrow \text{SHJ} \]

\[ \text{unix/linux} \]

\[ \text{grep } < \text{strip} > \mid \text{grep ... } \mid \text{out.txt} \]

\[ \text{pipe symbol} \]
Approach 2: Pipelining

iseek(a1=42, A) -> sort_{b_id} -> scan(C) -> filter_{a3=12} -> isam(id, B)

SHJ

M_{J_{C,b_id=B.id}}
Pipeline Breakers?

water tank

iseek(a1=42, A)

sort_{b_id}

water tank

scan(C)

isam(id, B)

MJ_{C,b_id=B.id}

filter_{a3=12}
Stages of SHJ/INLJ

1) Reading
2) Left input exhausted
3) Looping and Probing

(A) Just buffer
(B) Buffer in der
(C) Done.

(A + 13) Index probe
Stages of Quick-Sorting

1. Reading
2. Sorting
3. Outputting

Diagram showing the stages of quick-sorting.
Stages of External Merge Sort

1. Reading (with run generation)
2. Merging (without final merge)
3. Outputting (with final merge)

online-merge
Blocking vs Non-Blocking Algorithms

INLJ/SHJ (with bulkload)

bulkload

sort

INLJ/SHJ
(without bulkload)

filter

project

INLJ/SHJ

Merge

F ≥ 2

F ≥ 2

DPHJ

INLJ/SHJ (with bulkload) = bulkload

+ INLJ/SHJ (without bulkload) ∈ index probe
Copyrights and Credits

© iStock.com:
Horned_Rat