Discente: Emillyn Souza Panserini

Docente: Pablo  
Disciplina: Banco de dados 2

Atividade: Relatório técnico das consultas nas bases de IDH e base acadêmica

--questão 1

--Primeira consulta:

SELECT \*

FROM regiao AS r

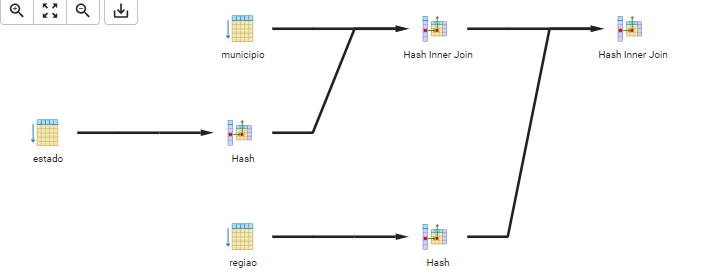
INNER JOIN estado AS e

ON r.codregiao = e.codregiao

INNER JOIN municipio mu

ON e.codestado = mu.codestado

WHERE r.codregiao > 1



--Segunda consulta:

SELECT \*

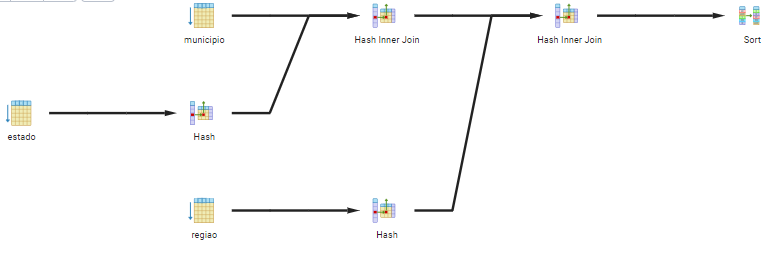
FROM regiao r

JOIN estado e ON r.codregiao = e.codregiao

JOIN municipio mu ON e.codestado = mu.codestado

WHERE r.codregiao > 1

ORDER BY e.nomeestado ASC;



A segunda terá mais custosa por conta da cláusula ORDER BY

-- questão 2

-- primeira consulta

SELECT nomemunicipio,

COUNT(m.codmunicipio) AS repeticoes

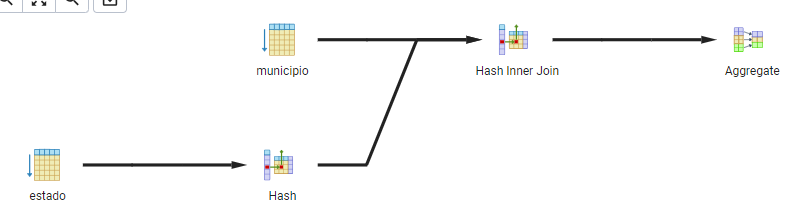
FROM estado AS e

INNER JOIN municipio AS m

ON m.codestado = e.codestado

GROUP BY m.nomemunicipio

having count(nomemunicipio) > 1;



-- segunda consulta

SELECT nomemunicipio

FROM municipio AS m

GROUP BY m.nomemunicipio

having count(nomemunicipio) > 1;



A primeira consulta é mais custosa por conta do INNER JOIN

-- questão 3

-- primeira consulta

SELECT avg(municipios\_por\_regiao) AS Media\_municipios\_por\_regiao

FROM(

SELECT COUNT(codmunicipio) AS municipios\_por\_regiao, codregiao

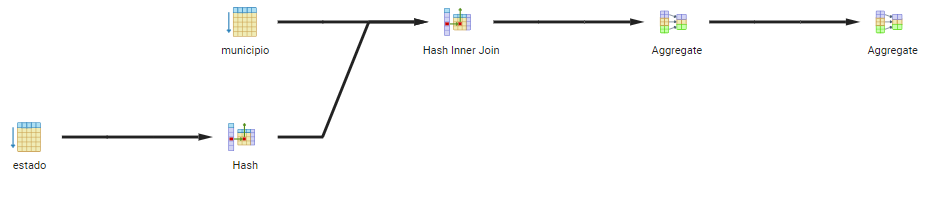
FROM municipio as m

INNER JOIN estado as e

ON m.codestado = e.codestado

GROUP BY codregiao

);



-- segunda consulta

SELECT AVG(subquery.municipios\_por\_regiao + 0) AS Media\_municipios\_por\_regiao

FROM (

SELECT COUNT(m.codmunicipio) + 0 AS municipios\_por\_regiao, e.codregiao

FROM municipio AS m

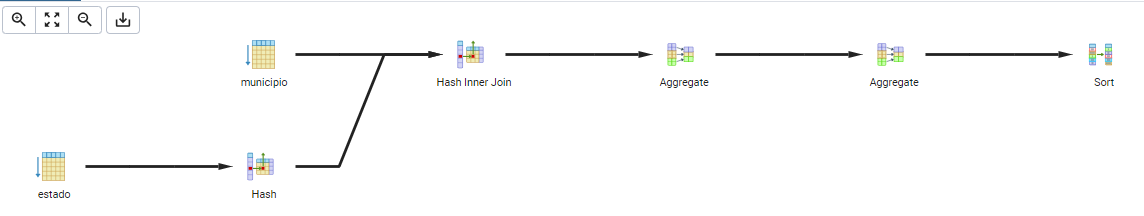
INNER JOIN estado AS e ON m.codestado = e.codestado

GROUP BY e.codregiao

HAVING COUNT(m.codmunicipio) > 0

) AS subquery

ORDER BY RANDOM();



COUNT(m.codmunicipio) + 0: A adição de + 0 impede que o banco de dados otimize a contagem, tornando-a um pouco mais custosa.

HAVING COUNT(m.codmunicipio) > 0: Essa condição é redundante, já que a contagem sempre será maior que zero, mas força o banco de dados a realizar uma verificação desnecessária.

ORDER BY RANDOM(): Ordenar os resultados de forma aleatória é uma operação custosa, especialmente quando a consulta lida com muitos registros.

-- questão 4

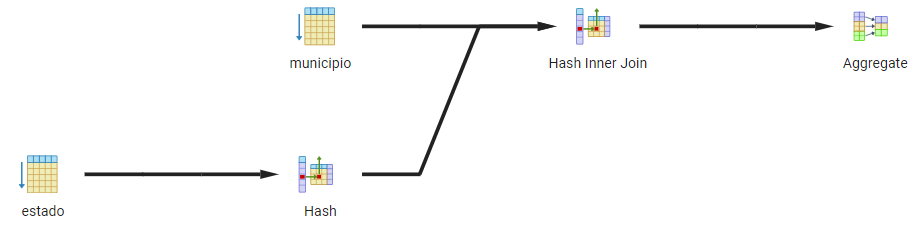
-- primeira consulta

SELECT e.siglaestado AS sigla, COUNT (codmunicipio) AS "qtd de municipios"

FROM estado as e

INNER JOIN municipio AS mu ON mu.codestado = e.codestado

GROUP BY sigla;



-- segunda consulta

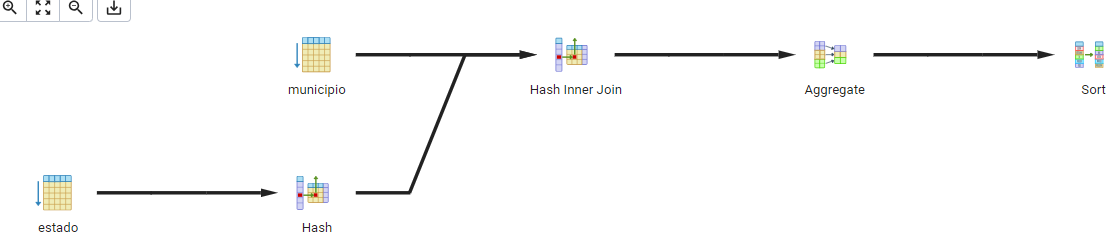
SELECT e.siglaestado AS sigla, COUNT (codmunicipio) AS "qtd de municipios"

FROM estado as e

INNER JOIN municipio AS mu ON mu.codestado = e.codestado

GROUP BY sigla

ORDER BY RANDOM();



ORDER BY RANDOM(): Ordena os resultados aleatoriamente, o que aumenta o custo, especialmente em grandes volumes de dados.

-- questão 5

-- segunda consulta

SELECT m.nomemunicipio AS "Municipio", idh\_longevidade AS "Longevidade"

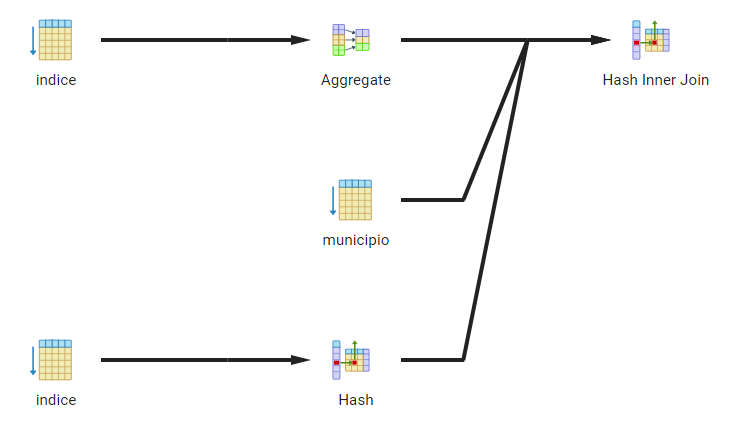
FROM indice AS i

INNER JOIN municipio AS m on i.codmunicipio = m.codmunicipio

WHERE idh\_longevidade = (select max(idh\_longevidade)

FROM indice

WHERE ano = 2000);



-- segunda consulta

SELECT m.nomemunicipio AS "Municipio", i.idh\_longevidade AS "Longevidade"

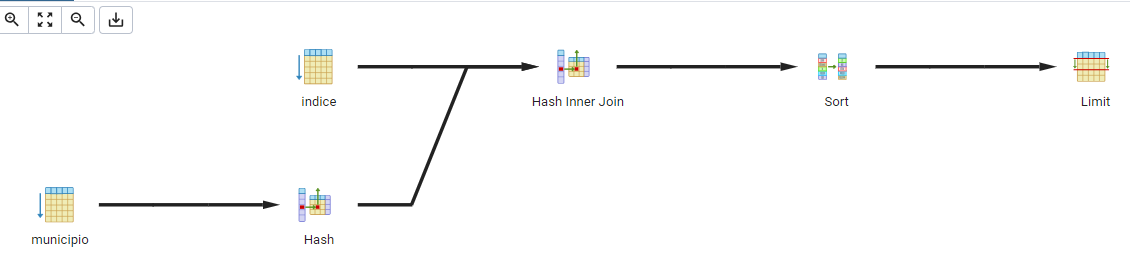
FROM indice AS i

INNER JOIN municipio AS m ON i.codmunicipio = m.codmunicipio

WHERE i.ano = 2000

ORDER BY i.idh\_longevidade DESC

LIMIT 1;



Segunda consulta é menos custosa pois:

Eliminei a subconsulta: Em vez de usar uma subconsulta para buscar o valor máximo, usei ORDER BY i.idh\_longevidade DESC para ordenar os valores por idh\_longevidade em ordem decrescente, e o LIMIT 1 retorna apenas o município com o maior valor.

Filtro direto por ano: Movi a condição WHERE ano = 2000 diretamente para a consulta principal, restringindo a busca apenas para esse ano antes de calcular o máximo.

Adicionei LIMIT 1: Para garantir que apenas o município com o maior idh\_longevidade seja retornado.

-- questão 6

-- primeira consulta

SELECT nomemunicipio, max(idh\_educacao) AS idh, ano

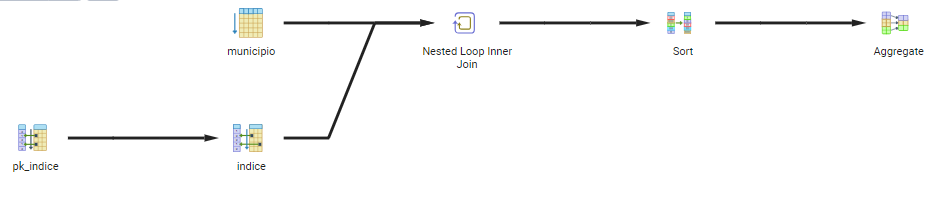
FROM municipio

INNER JOIN indice

ON indice.codmunicipio = municipio.codmunicipio

GROUP BY nomemunicipio, ano

HAVING nomemunicipio = 'Salvador';



-- segunda consulta  
SELECT nomemunicipio, MAX(idh\_educacao) AS idh, ano

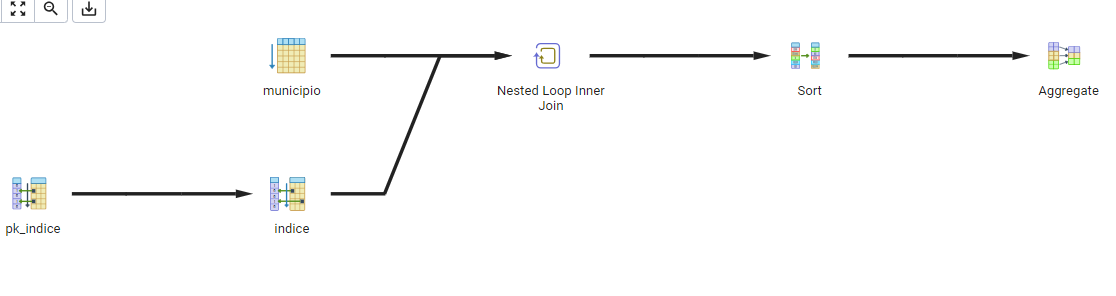
FROM municipio

INNER JOIN indice

ON indice.codmunicipio = municipio.codmunicipio

WHERE nomemunicipio = 'Salvador'

GROUP BY nomemunicipio, ano;  
  
  
Filtragem antecipada: Mudei a condição nomemunicipio = 'Salvador' para a cláusula WHERE antes do GROUP BY. Isso reduz o número de registros antes de aplicar a agregação e melhora a performance.



-- questão 7

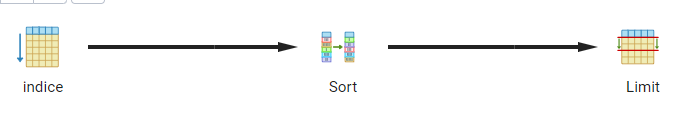
-- primeira consulta

SELECT codmunicipio, idh\_renda

FROM indice

ORDER BY idh\_renda desc

LIMIT 1;



--segunda consulta

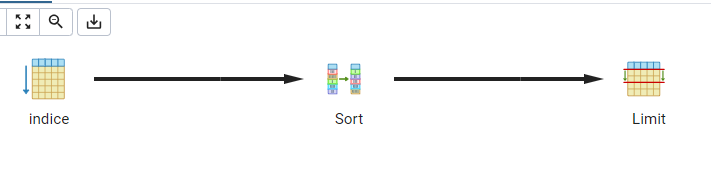
SELECT codmunicipio, idh\_renda

FROM indice

WHERE idh\_renda IS NOT NULL

ORDER BY idh\_renda DESC

LIMIT 1;



A segunda consulta é mais custosa pois, tem uma filtragem desnecessária, adiciona uma condição WHERE idh\_renda IS NOT NULL para garantir que todos os registros sejam considerados, mas isso não altera o resultado final e só adiciona um pouco mais de processamento.

-- questao 8

-- primeira consulta

SELECT nomeestado

FROM estado as e1

INNER JOIN

(

SELECT distinct codestado

FROM municipio AS m1

INNER JOIN

(

SELECT codmunicipio, idh\_renda

FROM indice

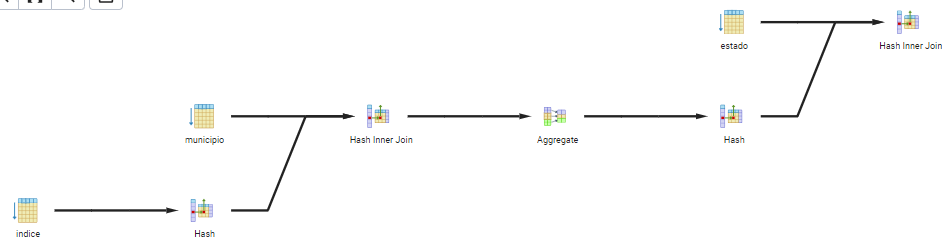
WHERE idh\_geral > 0.8

) AS m2

ON m1.codmunicipio = m2.codmunicipio

) AS e2

ON e1.codestado = e2.codestado;



--segunda consulta

SELECT DISTINCT e.nomeestado

FROM estado AS e

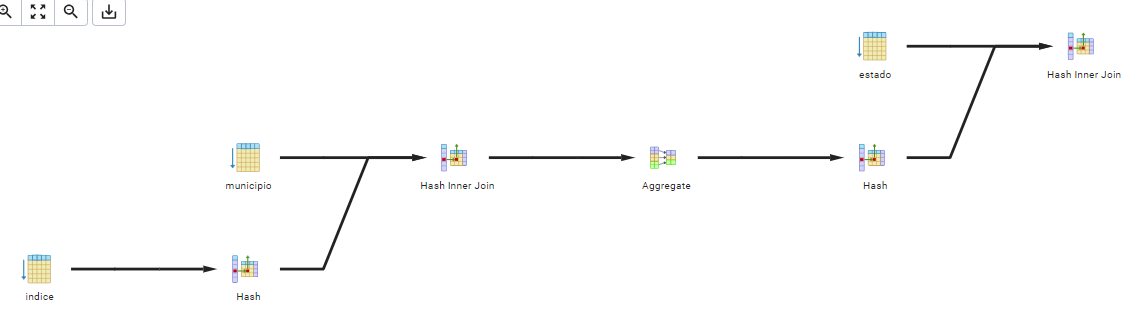
INNER JOIN municipio AS m

ON e.codestado = m.codestado

INNER JOIN indice AS i

ON m.codmunicipio = i.codmunicipio

WHERE i.idh\_geral > 0.8;



Otimização da segunda consulta em relação a primeira:

Uso direto de INNER JOIN: Realiza a junção diretamente entre as tabelas estado, municipio, e indice com a condição de filtragem aplicada diretamente no WHERE.

Uso de DISTINCT: Garante que o resultado final contenha apenas valores únicos para nomeestado.

-- questão 9

-- primeira consulta

SELECT max(idh\_educacao) AS idh\_educacao, codestado

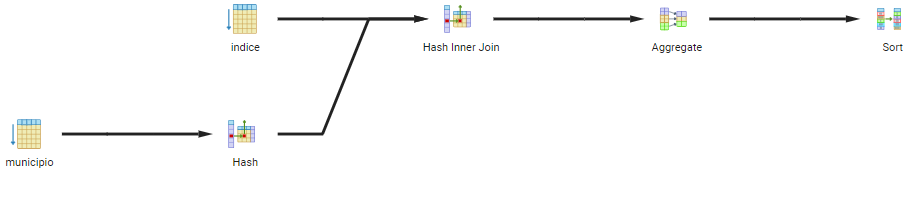
FROM municipio

INNER JOIN indice

ON municipio.codmunicipio = indice.codmunicipio

GROUP BY codestado

ORDER BY codestado;



-- segunda consulta

SELECT codestado, MAX(idh\_educacao) AS idh\_educacao

FROM (

SELECT municipio.codestado, indice.idh\_educacao, RANDOM() AS rand\_val

FROM municipio

INNER JOIN indice ON municipio.codmunicipio = indice.codmunicipio

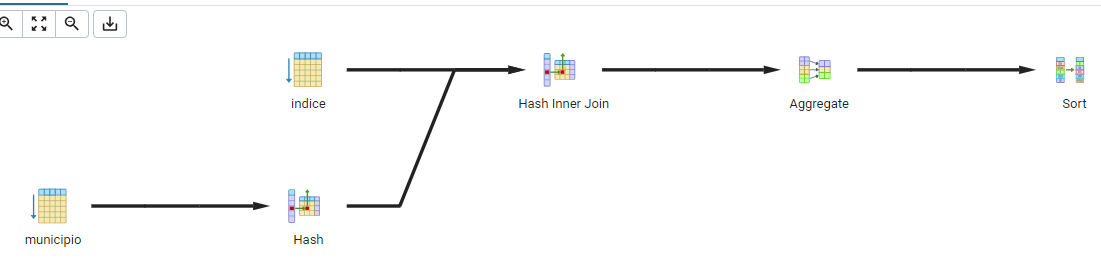
WHERE indice.idh\_educacao IS NOT NULL

) AS subquery

GROUP BY codestado

HAVING COUNT(idh\_educacao) > 0

ORDER BY codestado;



A segunda consulta é mais custosa pois: a subconsulta com RANDOM(): Adiciona uma coluna rand\_val com valores aleatórios. Embora essa coluna não afete o resultado final, ela adiciona uma carga adicional ao processamento. HAVING: Adicionei uma cláusula HAVING COUNT(idh\_educacao) > 0, mas não é necessário mas força o banco a processar a contagem.

--questão 10

-- primeira consulta

SELECT \*

FROM

(

SELECT i.ano, m.nomemunicipio,idh\_geral AS idh\_geral\_91,

idh\_renda AS idh\_renda\_91,

idh\_longevidade AS idh\_longevidade\_91,

idh\_educacao AS idh\_educacao\_91

from indice AS i

inner join municipio AS m

ON m. codmunicipio = i.codmunicipio

WHERE m.codestado = 5 and i.ano = 1991

UNION

SELECT i.ano, m.nomemunicipio,idh\_geral AS idh\_geral\_00,

idh\_renda AS idh\_renda\_00,

idh\_longevidade AS idh\_longevidade\_00,

idh\_educacao AS idh\_educacao\_00

from indice AS i

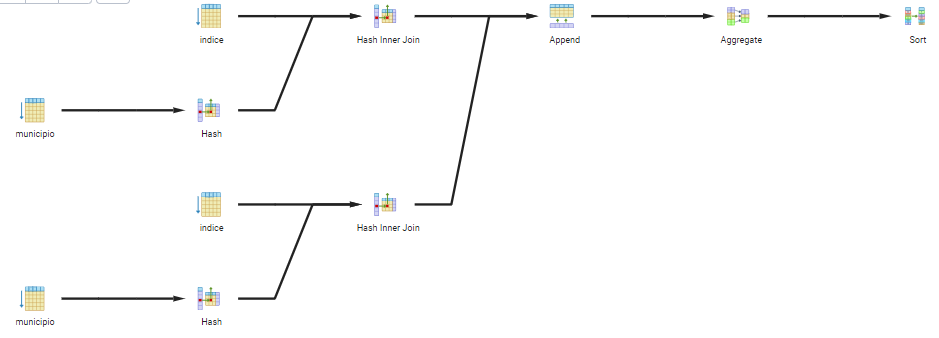
INNER JOIN municipio AS m

ON m. codmunicipio = i.codmunicipio

WHERE m.codestado = 5 and i.ano = 2000

)

ORDER BY nomemunicipio, ano;



-- segunda consulta

SELECT i.ano, m.nomemunicipio,

CASE WHEN i.ano = 1991 THEN idh\_geral ELSE NULL END AS idh\_geral,

CASE WHEN i.ano = 1991 THEN idh\_renda ELSE NULL END AS idh\_renda,

CASE WHEN i.ano = 1991 THEN idh\_longevidade ELSE NULL END AS idh\_longevidade,

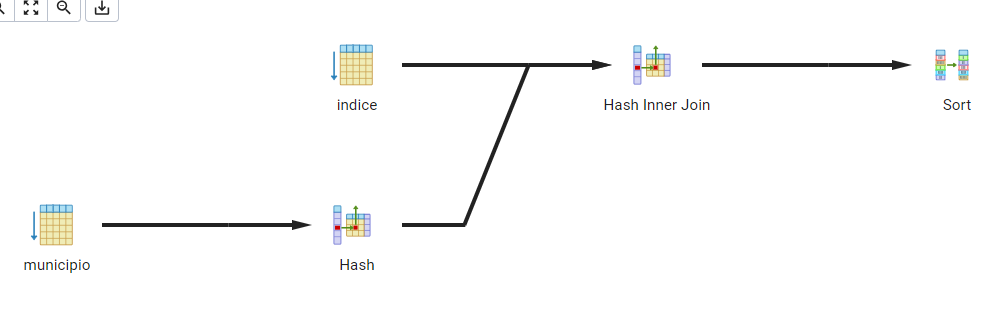
CASE WHEN i.ano = 1991 THEN idh\_educacao ELSE NULL END AS idh\_educacao

FROM indice AS i

INNER JOIN municipio AS m ON m.codmunicipio = i.codmunicipio

WHERE m.codestado = 5 AND i.ano IN (1991, 2000)

ORDER BY m.nomemunicipio, i.ano;



A segunda é mais otimizada pois:

Ao usar CASE: Em vez de usar UNION, que pode ser mais custoso, usei CASE para selecionar os valores com base no ano. Isso simplificou a consulta e evitou a necessidade de combinar conjuntos de resultados.

Filtrei por i.ano IN (1991, 2000): Filtragem dos anos diretamente na cláusula WHERE, reduzindo o número de registros processados.

Ordenação Simples: Ordenei diretamente por nomemunicipio e ano, que são necessários para o resultado final.

-- questão 11

-- primeira consulta

SELECT

e.siglaestado,

avg(case when ano = 1991 then i.idh\_geral end) AS idh\_geral\_1991,

avg(case when ano = 2000 then i.idh\_geral end) AS idh\_geral\_2000

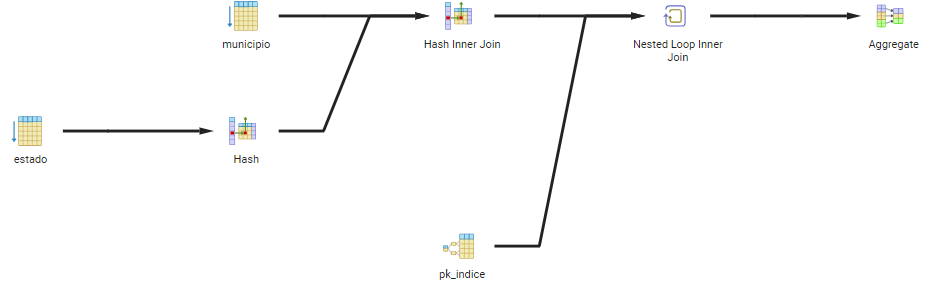
FROM indice AS i

INNER JOIN municipio AS mu on i.codmunicipio = mu.codmunicipio

INNER JOIN estado AS e on mu.codestado = e.codestado

WHERE e.siglaestado = 'SC' or e.siglaestado = 'AL'

GROUP BY e.siglaestado;



-- segunda consulta  
  
SELECT

e.siglaestado,

AVG(CASE WHEN i.ano = 1991 THEN i.idh\_geral ELSE NULL END) AS idh\_geral\_1991,

AVG(CASE WHEN i.ano = 2000 THEN i.idh\_geral ELSE NULL END) AS idh\_geral\_2000

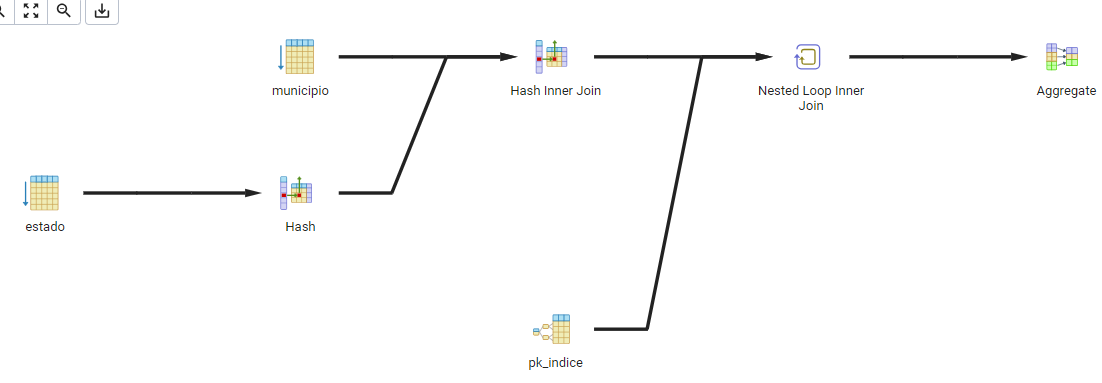
FROM indice AS i

INNER JOIN municipio AS mu ON i.codmunicipio = mu.codmunicipio

INNER JOIN estado AS e ON mu.codestado = e.codestado

WHERE e.siglaestado IN ('SC', 'AL')

GROUP BY e.siglaestado;



Motivos da segunda consulta ser mais otimizada:

Substituí e.siglaestado = 'SC' or e.siglaestado = 'AL' por e.siglaestado IN ('SC', 'AL'), que é mais eficiente e mais legível.

Uso de ELSE NULL: Incluí ELSE NULL no CASE para garantir que apenas valores correspondentes são considerados na média, o que é mais claro e evita a necessidade de conversão implícita.

Base acadêmica:  
  
ALTER TABLE takes ADD COLUMN gradenum NUMERIC(3,1);

CREATE OR REPLACE PROCEDURE atualizacao\_grade()

LANGUAGE plpgsql

AS $$

BEGIN

UPDATE takes

SET gradenum = CASE

WHEN grade = 'A' THEN 9.5

WHEN grade = 'A+' THEN 10.0

WHEN grade = 'A-' THEN 9.0

WHEN grade = 'B' THEN 8.0

WHEN grade = 'B+' THEN 8.5

WHEN grade = 'B-' THEN 7.5

WHEN grade = 'C' THEN 6.5

WHEN grade = 'C+' THEN 7.0

WHEN grade = 'C-' THEN 6.0

WHEN grade = 'D' THEN 5.0

WHEN grade = 'D+' THEN 5.5

WHEN grade = 'D-' THEN 4.5

WHEN grade = 'F' THEN 0.0

ELSE NULL -- Padrão para notas inesperadas

END;

END;

$$;

SELECT

dep.dept\_name AS departamento,

ins.name AS professor,

cla.room\_number AS turma,

tks.year,

AVG(tks.gradenum) AS media\_notas

FROM takes tks

INNER JOIN instructor ins ON ins.id = tks.id

INNER JOIN department dep ON dep.dept\_name = ins.dept\_name

INNER JOIN classroom cla ON cla.building = dep.building

WHERE tks.year BETWEEN 2006 AND 2010

GROUP BY dep.dept\_name, ins.name, cla.room\_number, tks.year;

Segunda questão base acadêmica   
  
SELECT stu.name AS aluno,

tks.course\_id AS disciplina,

ins.name AS professor,

tks.grade AS nota,

ins.dept\_name AS departamento\_professor

FROM

student stu

JOIN takes tks ON stu.id = tks.id

JOIN instructor ins ON tks.id = ins.id

JOIN department dep ON ins.dept\_name = dep.dept\_name

WHERE tks.grade IS NOT NULL