

Tipos avançados de junções

Sumário

- **Junção externa**
- Semi-junção
 - Semi-junção vs junção regular
- Anti-junção
 - anti-junção vs junção externa

Junção Externa

- Considerando que uma junção tem duas partes
 - Parte principal
 - Parte secundária
- A junção externa traz todos os registros da parte principal
 - Mesmo que alguns desses registros não tenham correspondência com nenhum registro da parte secundária
- Possibilidades mais usadas
 - **LEFT** (OUTER) JOIN: O lado da **esquerda** da junção externa é a parte principal
 - **RIGHT** (OUTER) JOIN: O lado da **direita** da junção externa é a parte principal

Junção Externa

- Ex.

```
SELECT m.title, mc.c_name  
FROM movie m LEFT JOIN movie_cast mc ON m.idM = mc.idM
```



- Partes da junção

- Principal: movie
- Secundária: movie_cast

- Retorno da junção

- Todos os filmes
 - combinados com os nomes de personagens(c_name) vindos de movie_cast
 - Filmes sem correspondência em movie_cast também são retornados
 - Nesses casos, a coluna c_name é preenchida com o valor **NULO**

Junção Externa

- As estratégias de junção externa podem ser classificadas em
 - Left Outer Join
 - A parte principal fica do lado externo da junção
 - Algoritmos
 - Nested Loop Left Outer Join
 - Hash Left Outer Join
 - Merge Left Outer Join
 - Right Outer Join
 - A parte principal fica do lado interno da junção
 - Algoritmos
 - Hash Right Outer Join
 - Merge Right Outer Join

Junção Externa

- **Exemplo 1**

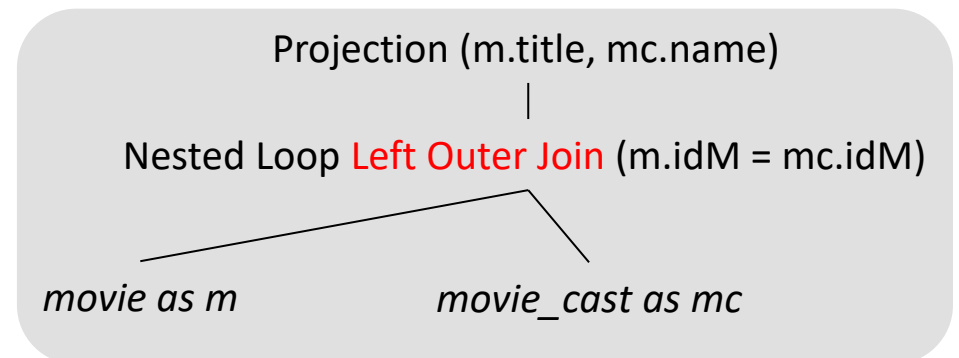
- retornar títulos de filmes e nomes de seus personagens.
- Filmes sem personagens também devem ser retornados
 - Com o nome do personagem nulo

```
SELECT m.title, mc.c_name  
FROM movie m  
LEFT JOIN movie_cast mc ON m.idM = mc.idM
```

Junção Externa

- O plano abaixo usa um Left Outer Join
 - A parte principal (movie) fica no lado externo da junção

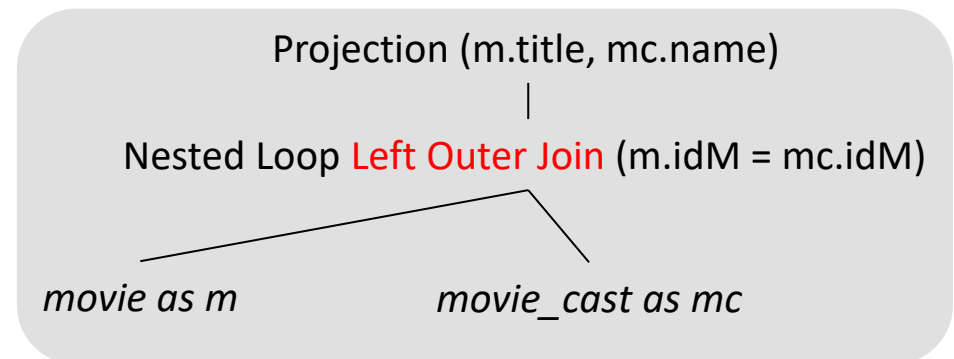
```
SELECT m.title, mc.c_name  
FROM movie m  
LEFT JOIN movie_cast mc ON m.idM = mc.idM
```



Junção Externa

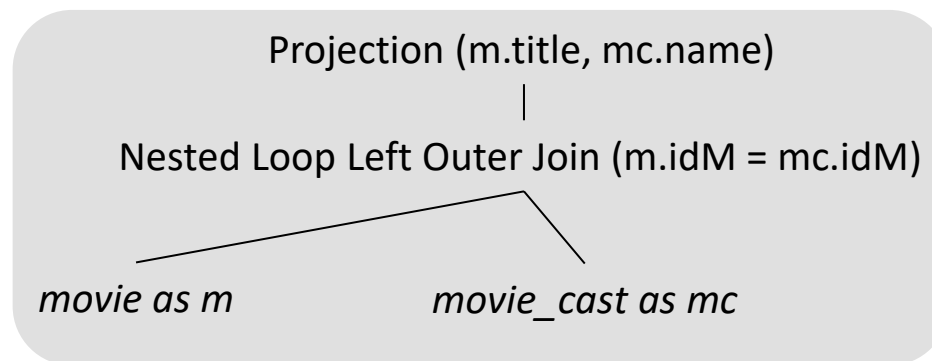
- O mesmo plano seria gerado usando **RIGHT JOIN** na consulta SQL, com movie do lado direito
 - A parte principal (movie) continua no lado externo da junção

```
SELECT m.title, mc.c_name  
FROM movie_cast mc  
RIGHT JOIN movie m ON mc.idM = m.idM
```



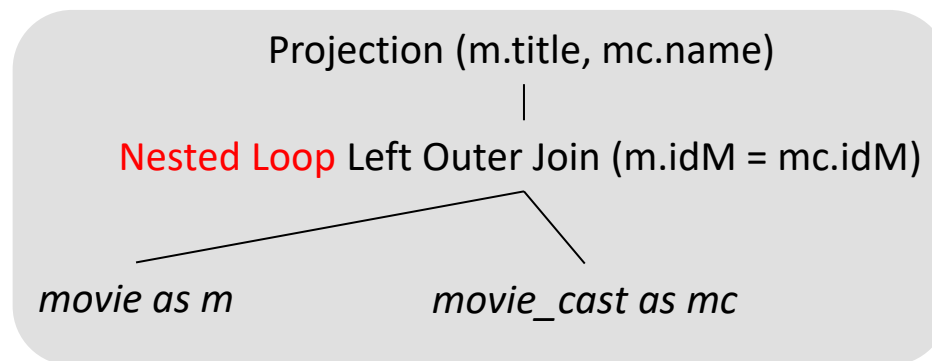
Junção Externa

- Funcionamento do Left Outer Join
 - Para cada registro do lado externo, as correspondências são buscadas
 - Se um registro não possuir correspondência
 - Ele é complementado com nulo
 - **Obs.** Essa verificação adicional torna a junção externa um pouco menos eficiente do que a junção regular



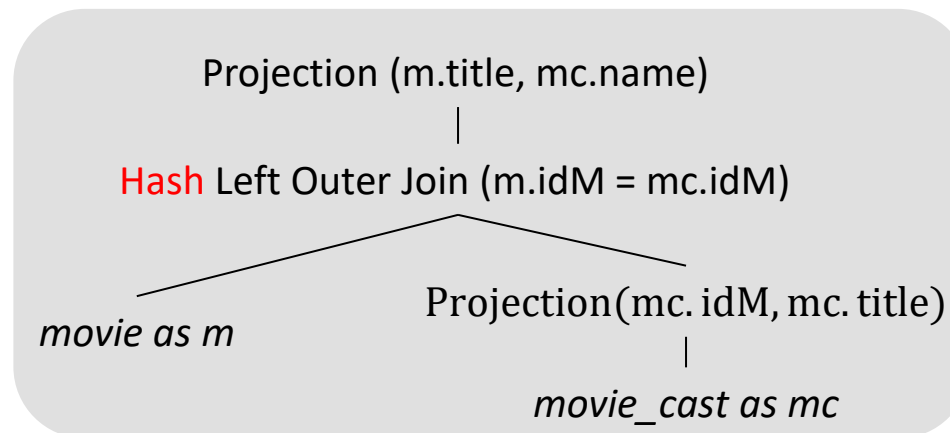
Junção Externa

- Existem várias estratégias de Left Outer Join
 - O plano abaixo usa um **Nested Loop**
- Essa abordagem não recorre à materialização



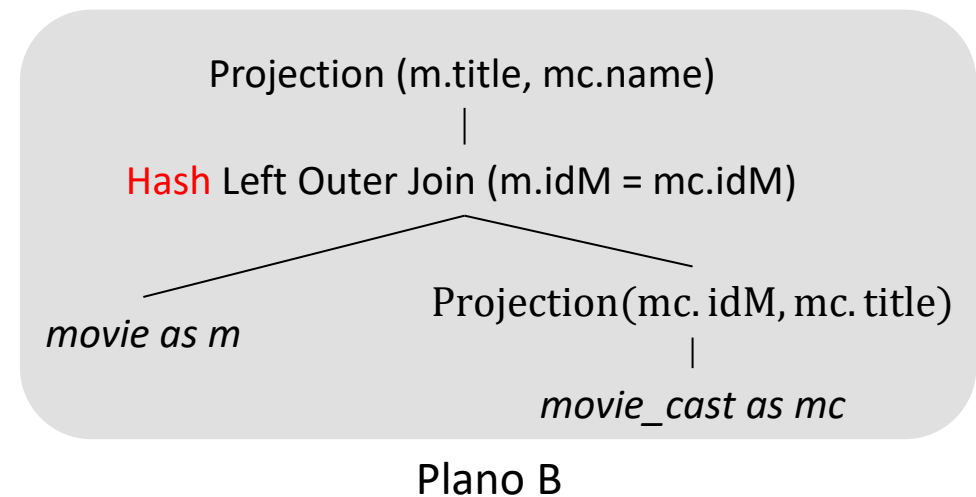
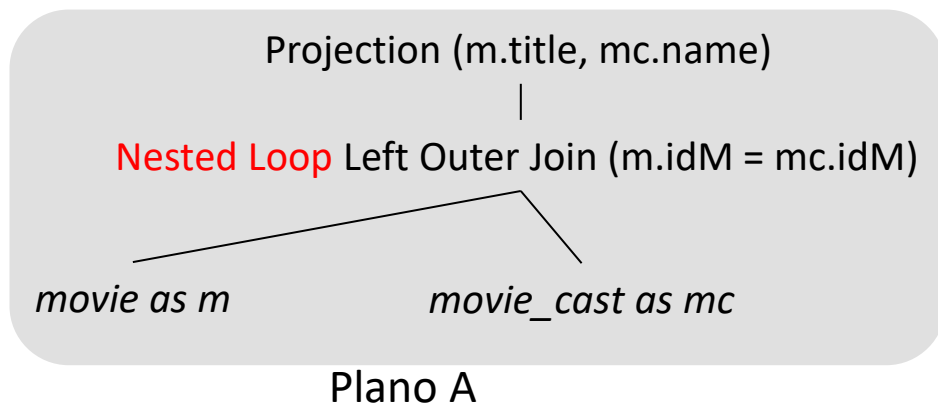
Junção Externa

- Já o plano abaixo usa **Hash Join**
- Baseada em tabela hash (usa materialização no lado interno)
 - A busca é feita sobre a tabela hash



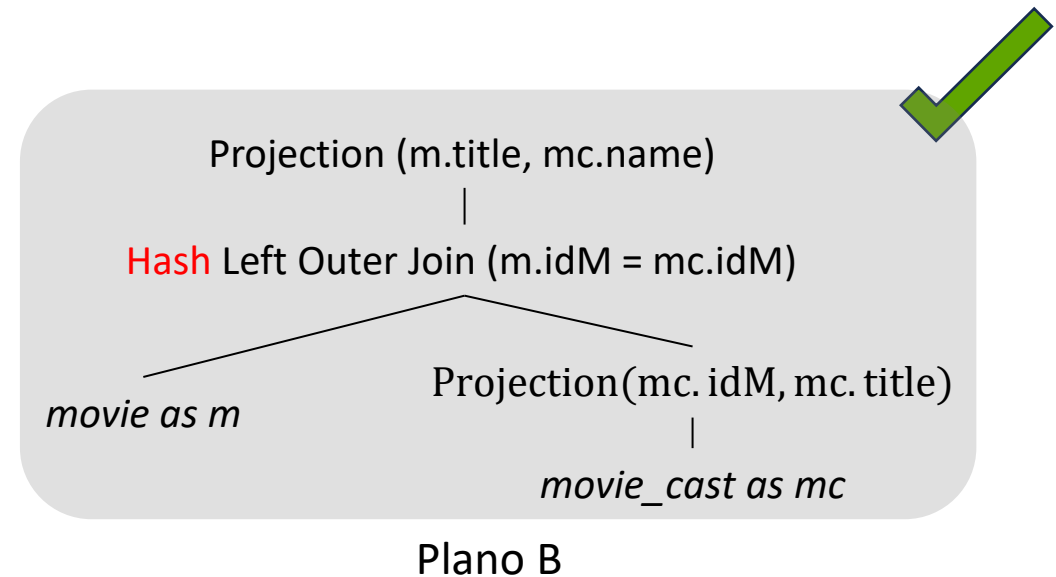
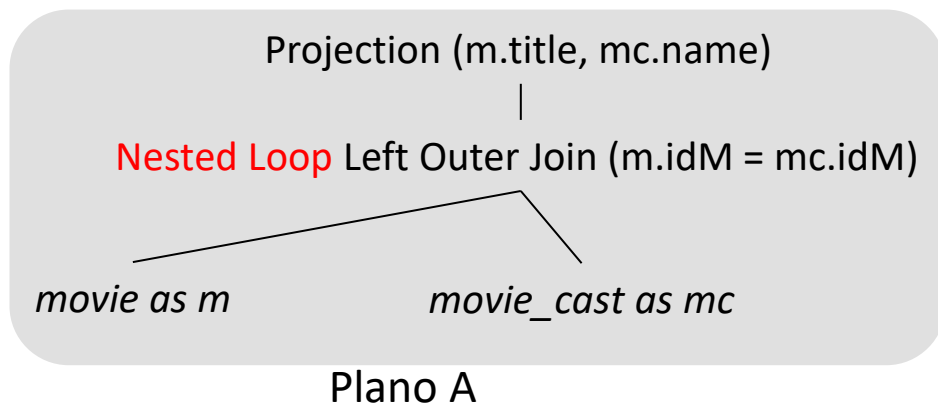
Junção Externa

- Os planos A e B mostram essas duas variações do left outer join
 - Plano A: com Nested Loop Left Outer Join
 - Plano B: com Hash Left Outer Join
- Qual deles é mais eficiente?



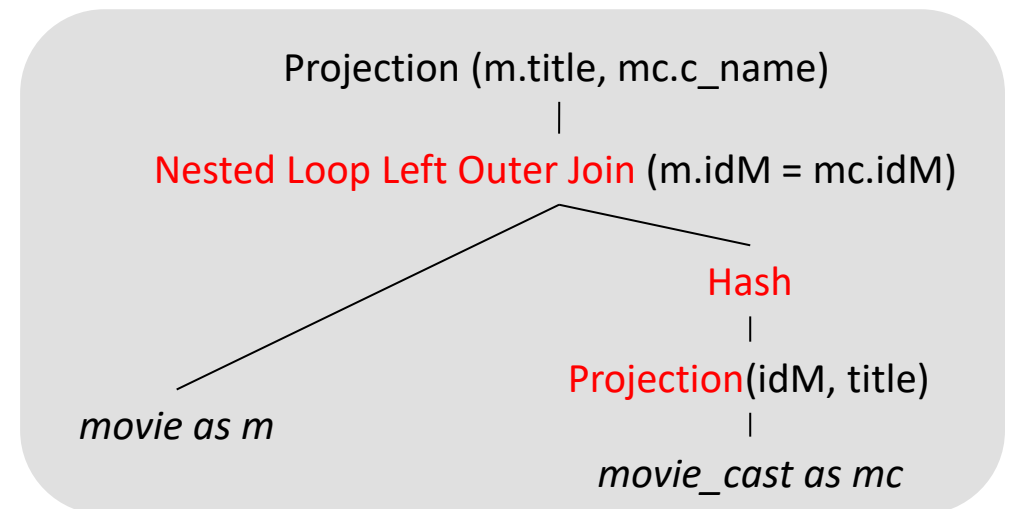
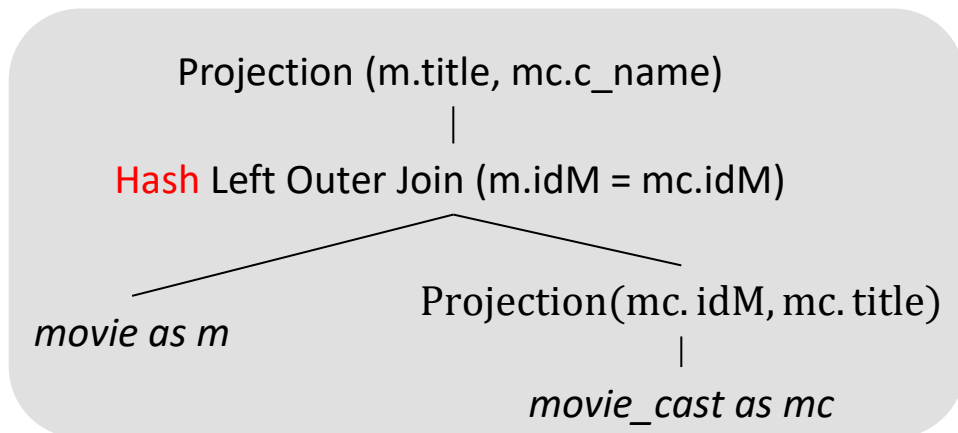
Junção Externa

- Plano B é mais eficiente
 - A busca é feita sobre uma tabela hash
 - Contudo, consome memória



Junção Externa

- Curiosidade: No **DBest**, o operador Hash Left Outer Join pode ser substituído por uma combinação de dois operadores
 - Nested Loop Left Outer Join
 - Hash



Junção Externa

- **Exemplo 2**

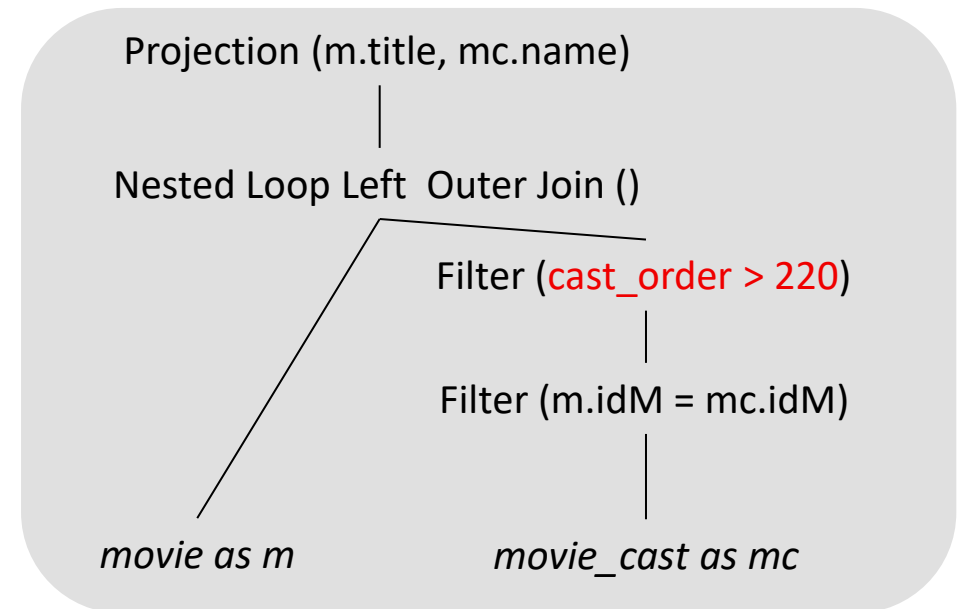
- retornar títulos de filmes e nomes de seus personagens.
- Filmes sem personagens também deve ser retornados
 - Com o nome do personagem nulo
- Existe um filtro seletivo sobre o lado secundário (movie_cast)

```
SELECT m.title, mc.c_name  
FROM movie m LEFT JOIN movie_cast mc  
ON m.idM = mc.idM  
AND cast_order > 220
```

Junção Externa

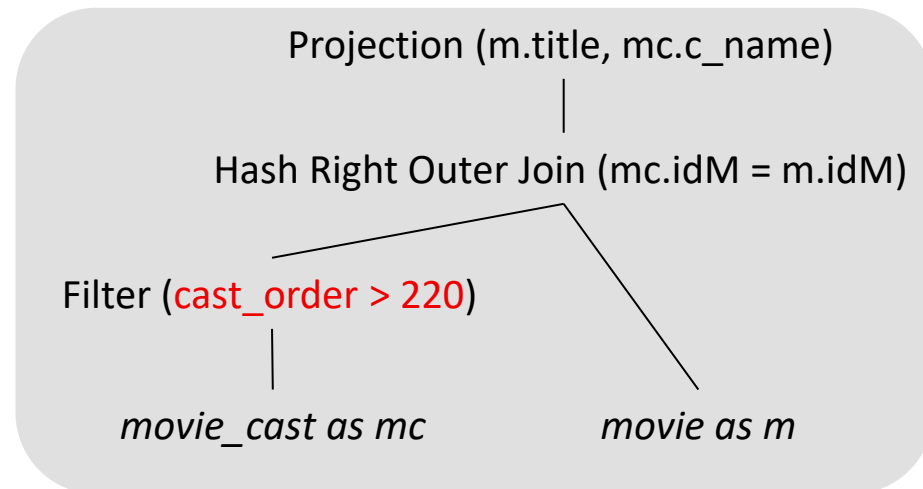
- O plano abaixo usa o Nested Loop Left Outer Join
 - Idealmente, a tabela movie_cast seria colocada no lado externo, devido a sua alta seletividade
 - Mas o nested loop exige que a parte principal(movie) fique do lado externo

```
SELECT m.title, mc.c_name  
FROM movie m LEFT JOIN movie_cast mc  
ON m.idM = mc.idM  
AND cast_order > 220
```



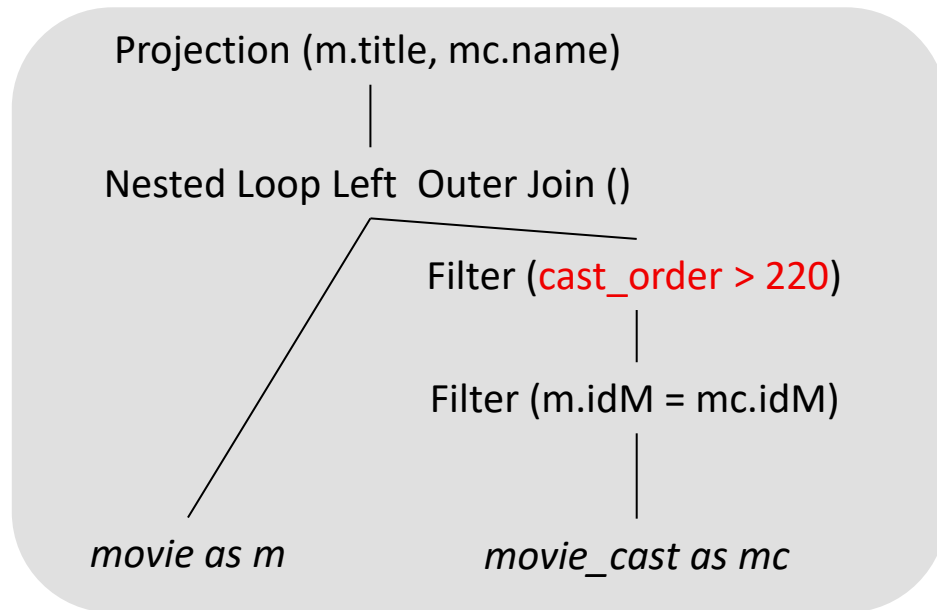
Junção Externa

- Alternativa: **Hash Right Outer Join**
- Funcionamento
 - O lado interno é materializado
 - Se um movie_cast tiver correspondência com o lado interno
 - A correspondência é retornada
 - Além disso, o filme na tabela hash é marcado
 - Ao final
 - filmes não marcados são retornados e complementados com nulos

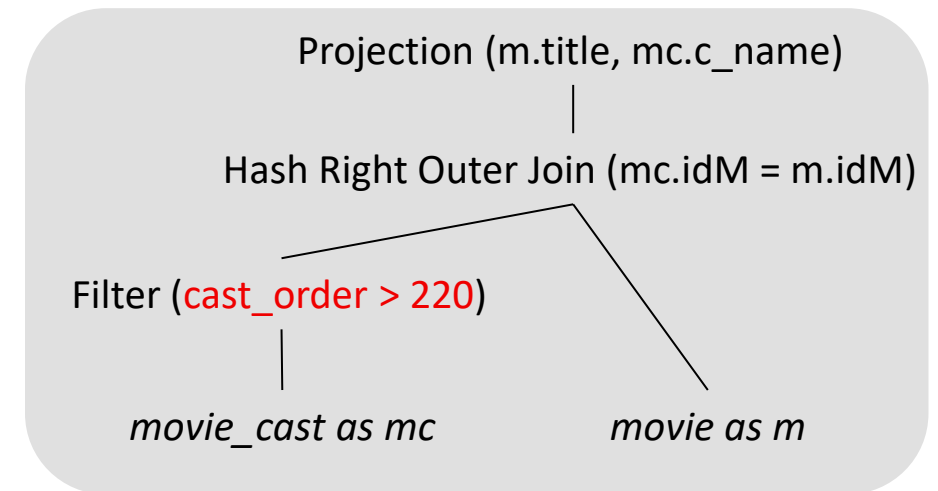


Junção Externa

- Qual é mais eficiente?



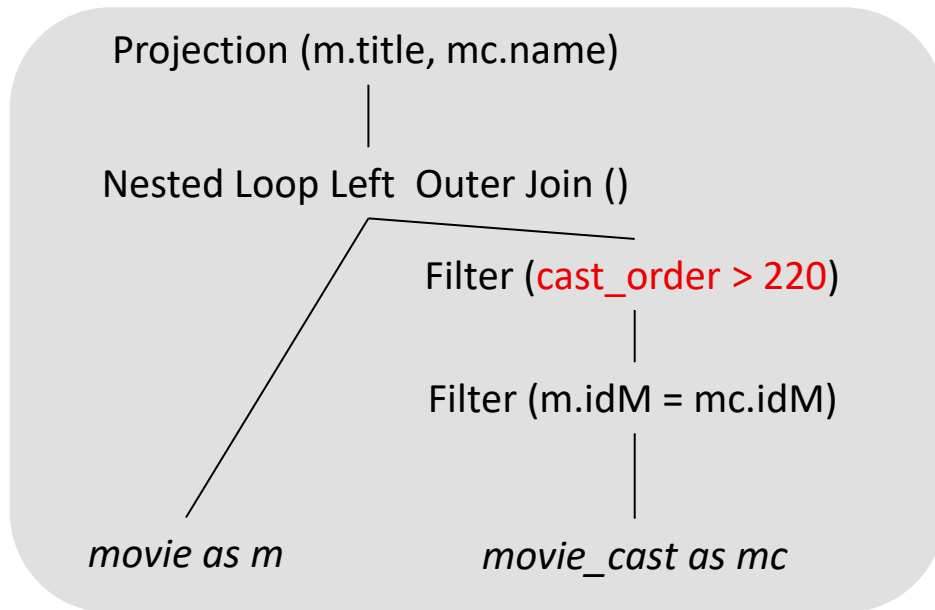
Plano A



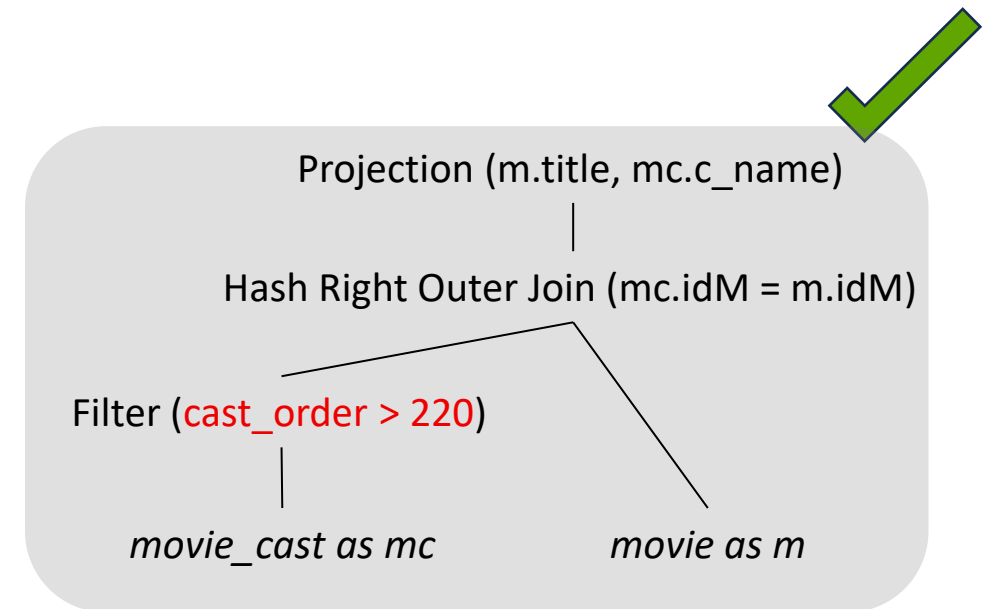
Plano B

Junção Externa

- O plano B é mais eficiente
 - Realiza menos buscas
 - Contudo, consome mais memória



Plano A



Plano B

Junção Externa

- **EXEMPLO 3:** títulos de filmes que não tenham membros do elenco

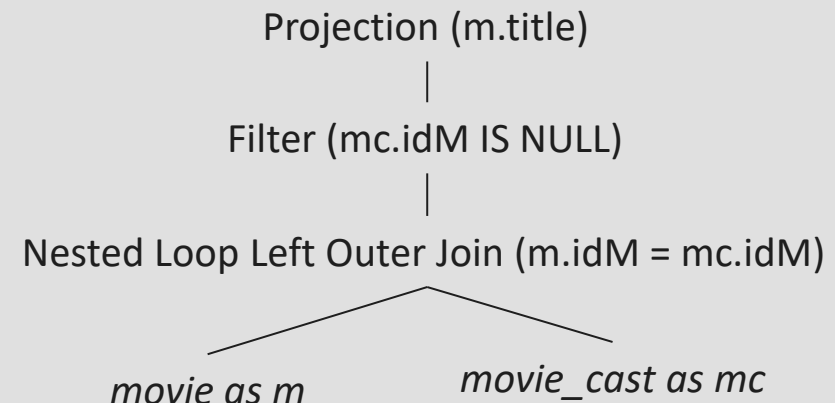
```
SELECT m.title  
FROM movie m LEFT JOIN movie_cast mc ON m.idM = mc.idM  
WHERE mc.movie_id IS NULL
```

- Consulta resolvida com junção externa e comparação com nulo
- Observe que colunas de movie_cast não são retornadas

Junção Externa

- No plano de execução
 - A junção externa mantém todos os filmes
 - O filtro remove os filmes que não tenham nenhum movie_cast

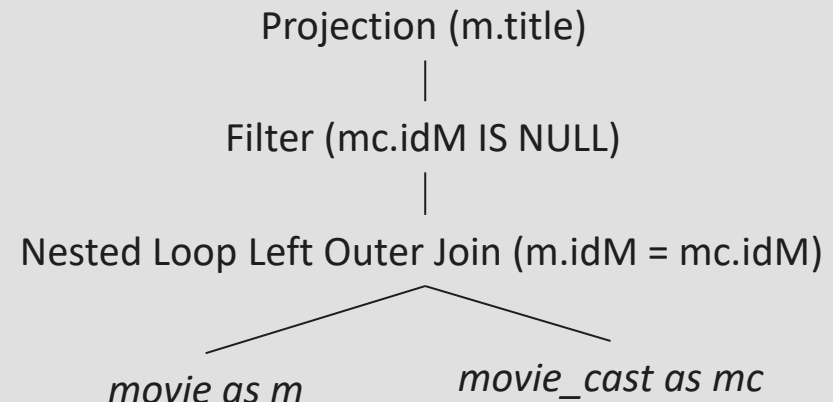
```
SELECT title  
FROM movie m  
LEFT JOIN movie_cast mc ON m.idM = mc.idM  
WHERE mc.idM IS NULL
```



Junção Externa

- Apesar de funcionar, é melhor usar um operador que foi especialmente projetado para casos assim
 - Como veremos mais adiante

```
SELECT title  
FROM movie m  
LEFT JOIN movie_cast mc ON m.idM = mc.idM  
WHERE mc.idM IS NULL
```



Junção Externa

- Junção externa é diferente da junção regular!
 - Seu processamento é um pouco mais custoso
 - Pode dificultar a troca do lado da junção
 - Ou exigir que seja usada uma solução materializada
 - Que aumenta o consumo de memória
- Por isso, certifique-se de que
 - as tuplas sem correspondência são realmente importantes para a aplicação
 - não seja possível eliminar a junção externa sem que isso altere o resultado

Sumário

- Junção externa
- **Semi-junção**
 - Semi-junção vs junção regular
- Anti-junção
 - anti-junção vs junção externa

Semi-junção

- Considerando que uma junção tem duas partes
 - parte principal
 - parte secundária
- Em uma semi-junção
 - Apenas tuplas da parte principal são retornadas
 - Cada tupla pode ser retornada apenas uma vez
 - Caso haja correspondências com a parte secundária

Semi-junção

- Em SQL, sem-junções são expressas na forma de subconsultas
 - A parte principal é a consulta externa
 - A parte secundária é a subconsulta
- Os exemplos abaixo mostram duas formas de uso (EXISTS e IN)
 - Parte principal: movie
 - Parte secundária: movie_cast

```
SELECT title
FROM movie m WHERE EXISTS
    (SELECT 1 FROM movie_cast mc
     WHERE m.idM = mc.idM )
```

```
SELECT title
FROM movie WHERE idM IN
    (SELECT idM FROM movie_cast)
```

Semi-junção

- A linguagem SQL fornece diversos operadores para subconsultas
- Ex.
 - IN
 - EXISTS
 - ALL
 - SOME
- O IN e o EXISTS são os mais comumente usados

Semi-junção

- EXISTS
 - a subconsulta contém uma coluna de correlação (**m.idM**)
 - Não importa o que a subconsulta retorna
 - Basta verificar que algo está sendo retornado
 - No exemplo, o retorno é uma constante (**1**)

```
SELECT title
FROM movie m WHERE EXISTS
    (SELECT 1 FROM movie_cast mc
     WHERE m.idM = mc.idM )
```

Semi-junção

- IN
 - A subconsulta é independente da consulta principal
 - Não possui coluna de correlação
 - As correspondências são feitas comparando colunas da consulta principal com o retorno da subconsulta

```
SELECT title  
FROM movie WHERE idM IN  
    (SELECT idM FROM movie_cast)
```

Semi-junção

- As estratégias de semi-junção podem ser classificadas em
 - Left Semi Join
 - A parte principal fica do lado externo da junção
 - Algoritmos
 - Nested Loop Left Semi Join
 - Hash Left Semi Join
 - Merge Left Semi Join
 - Right Semi Join
 - A parte principal fica do lado interno da junção
 - Algoritmos
 - Hash Right Semi Join
 - Merge Right Semi Join

Semi-junção

- **Exemplo 1:** título de filmes que contenham membros do elenco registrados

```
SELECT title
FROM movie m WHERE EXISTS
  (SELECT 1 FROM movie_cast mc
   WHERE m.idM = mc.idM )
```

- Como essa consulta é representada internamente?

Semi-junção

- A consulta é um exemplo de semi-junção
 - Retorna títulos de filme que contenham membros do elenco
 - Cada filme que satisfaça essa restrição é retornado uma única vez
 - Nenhuma coluna de elenco é retornada

```
SELECT title  
FROM movie m WHERE EXISTS  
    (SELECT 1 FROM movie_cast mc  
     WHERE m.idM = mc.idM )
```


Semi-junção

- A mesma consulta poderia ser representada usando IN

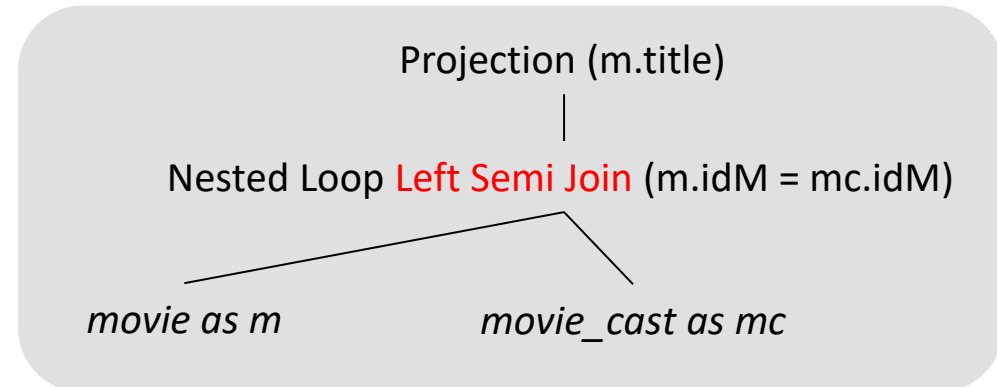
```
SELECT title
FROM movie m WHERE EXISTS
    (SELECT 1 FROM movie_cast mc
     WHERE m.idM = mc.idM )
```

```
SELECT title
FROM movie WHERE idM IN
    (SELECT idM FROM movie_cast)
```

Semi-junção

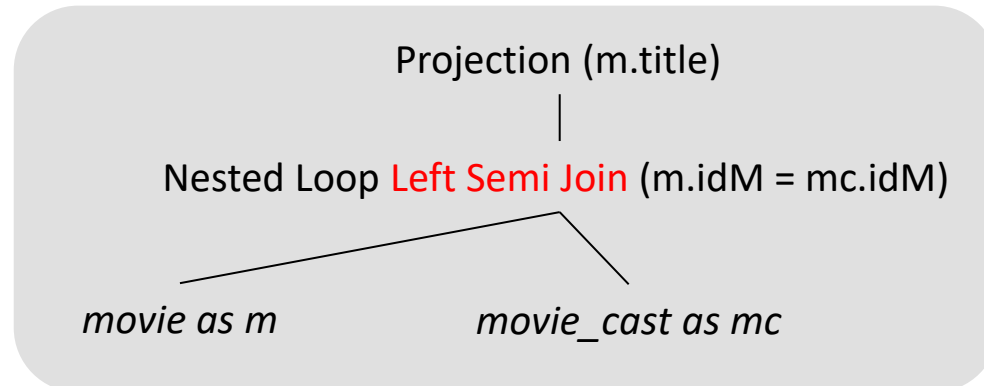
- O plano abaixo usa um Left Semi Join
 - A parte principal (movie) fica no lado externo da junção
 - Apenas registros da parte principal podem ser retornados

```
SELECT title
FROM movie m WHERE EXISTS
  (SELECT 1 FROM movie_cast mc
   WHERE m.idM = mc.idM )
```



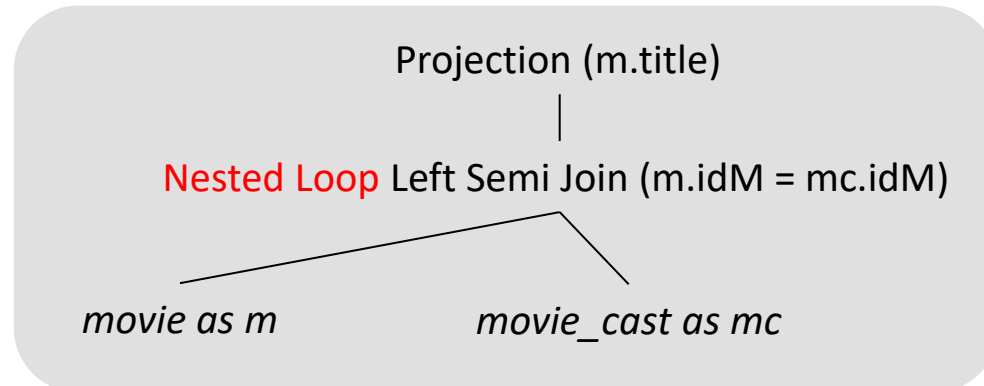
Semi-junção

- Funcionamento do Left Semi Join
 - A junção não busca correspondências
 - Apenas verifica se elas existem
 - Por isso, pouco importa o que foi usado no SELECT da subconsulta



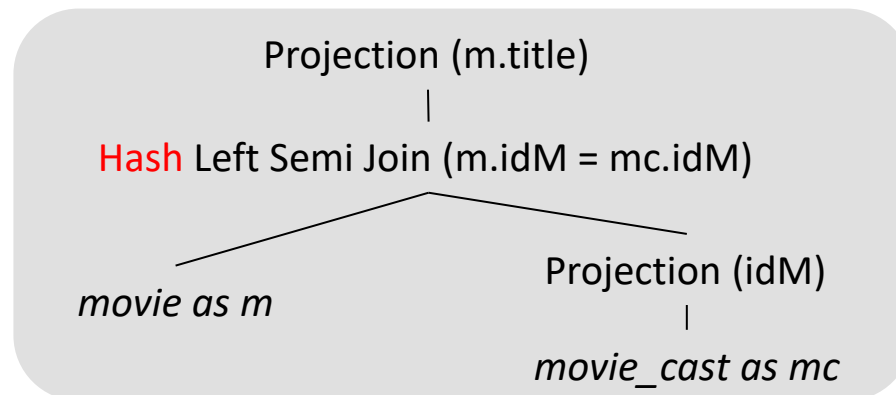
Semi-junção

- Existem vários algoritmos de Left Semi Join
- O plano abaixo usa um **Nested Loop**



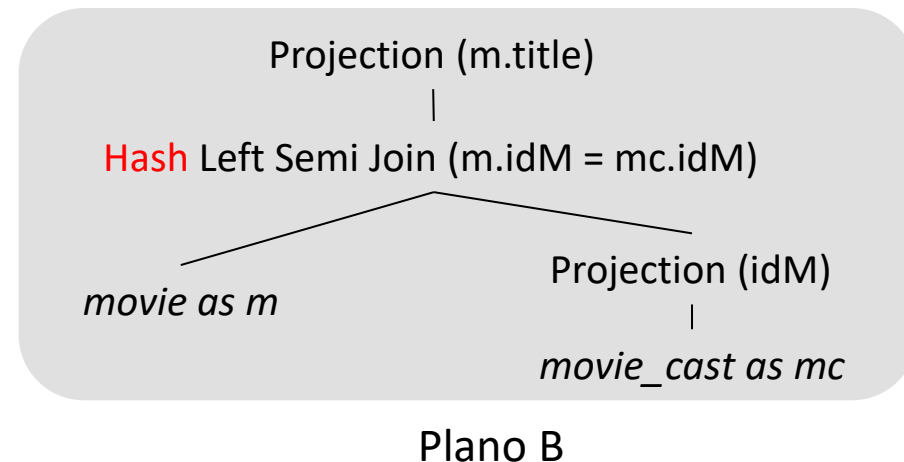
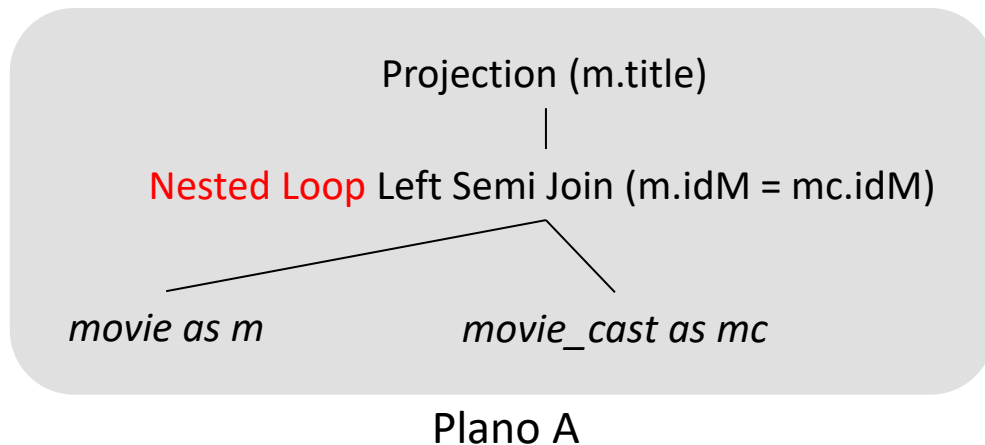
Semi-junção

- Já o plano abaixo usa um **Hash Join**
- O Hash Join usa uma tabela hash no lado interno
 - A verificação do semi Join é feita sobre a tabela hash



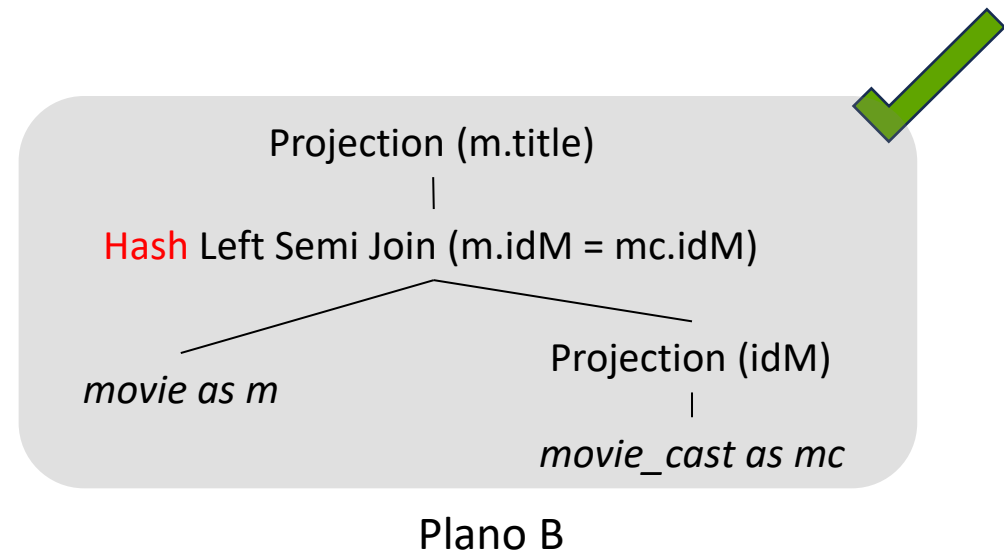
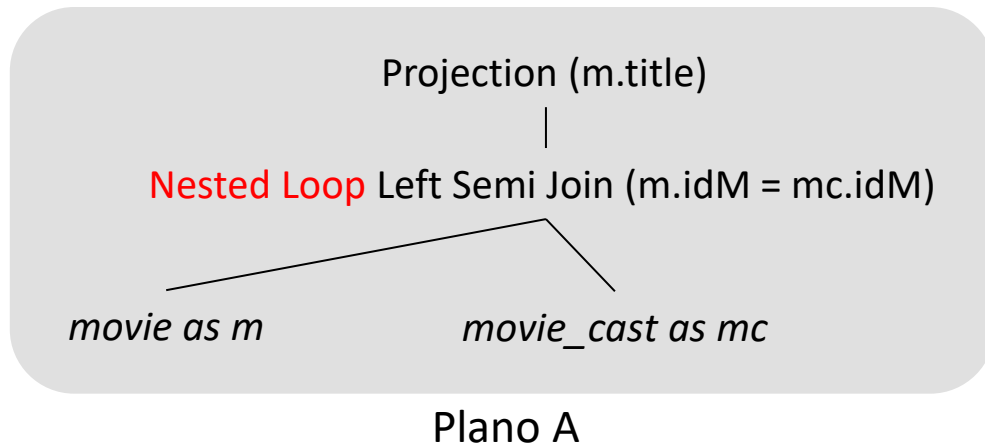
Semi-junção

- Os planos A e B mostram essas duas possibilidades de Left Semi Join
 - Plano A: com Nested Loop Left Semi Join
 - Plano B: com Hash Left Semi Join
- Qual deles é mais eficiente?



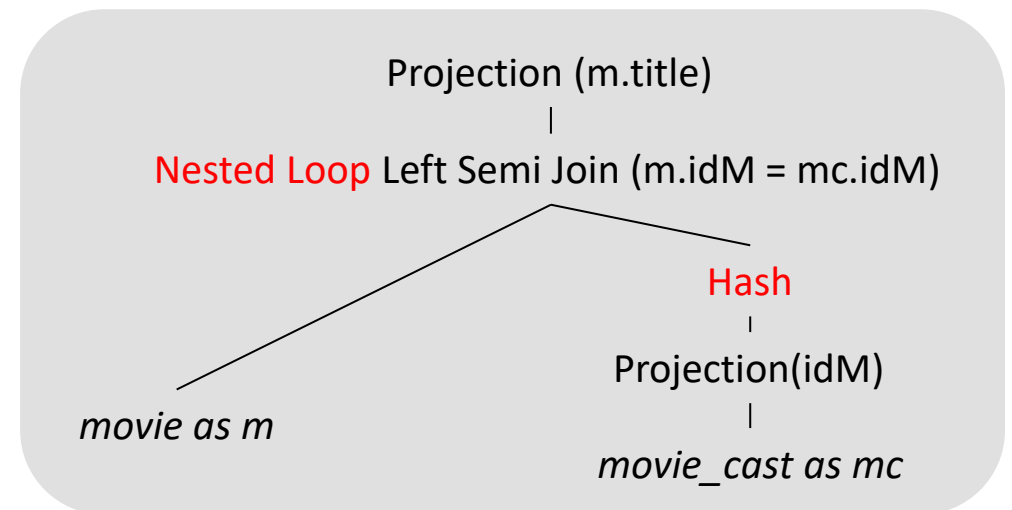
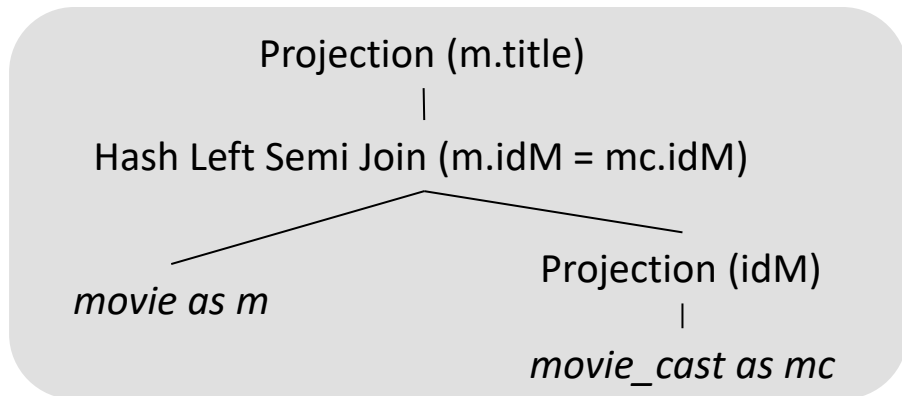
Semi-junção

- Plano B é mais eficiente
 - A busca é feita sobre uma tabela hash
 - No entanto, consome memória



Semi-junção

- Curiosidade: No **DBest**, o operador Hash Left Semi Join pode ser substituído por uma combinação de dois operadores
 - Nested Loop Left Semi Join
 - Hash



Semi-junção

- Limitação das estratégias Left Semi Join
 - a parte principal **sempre** fica no lado externo da junção
- Se a consulta tiver um filtro seletivo na parte secundária
 - Não será possível movê-lo para o lado externo
- Nesses casos, os otimizadores de consulta podem disponibilizar outros algoritmos de junção
 - Exemplos
 - **Nested Loop Join com remoção de duplicatas**
 - **Right Semi Join**

Semi-junção

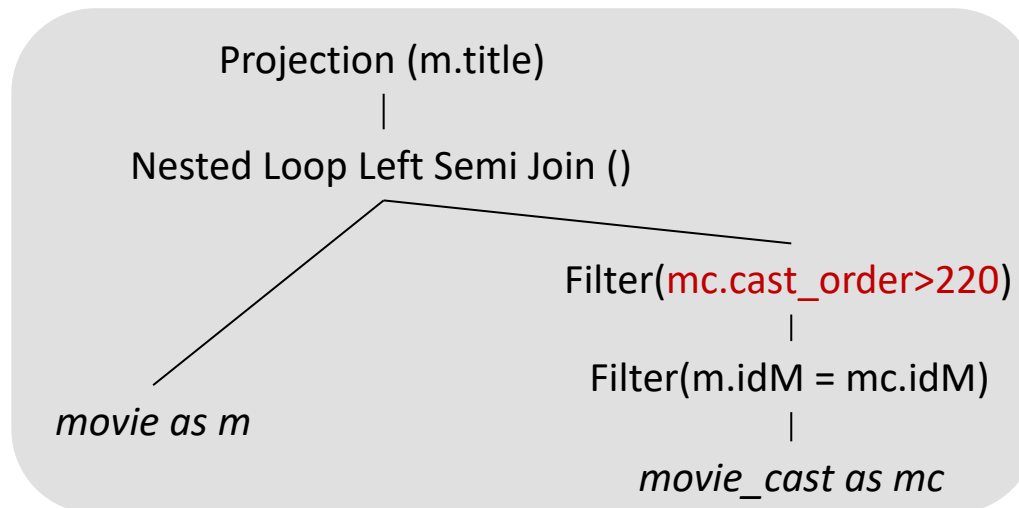
- **Exemplo 2:** título de filmes que contenham mais do que 220 personagens

```
SELECT title
FROM movie WHERE idM IN
    (SELECT idM FROM movie_cast
     WHERE cast_order > 220)
```

- Consulta escrita com IN, mas também poderia ser com EXISTS
- Existe um filtro seletivo sobre a parte secundária
 - cast_order > 220

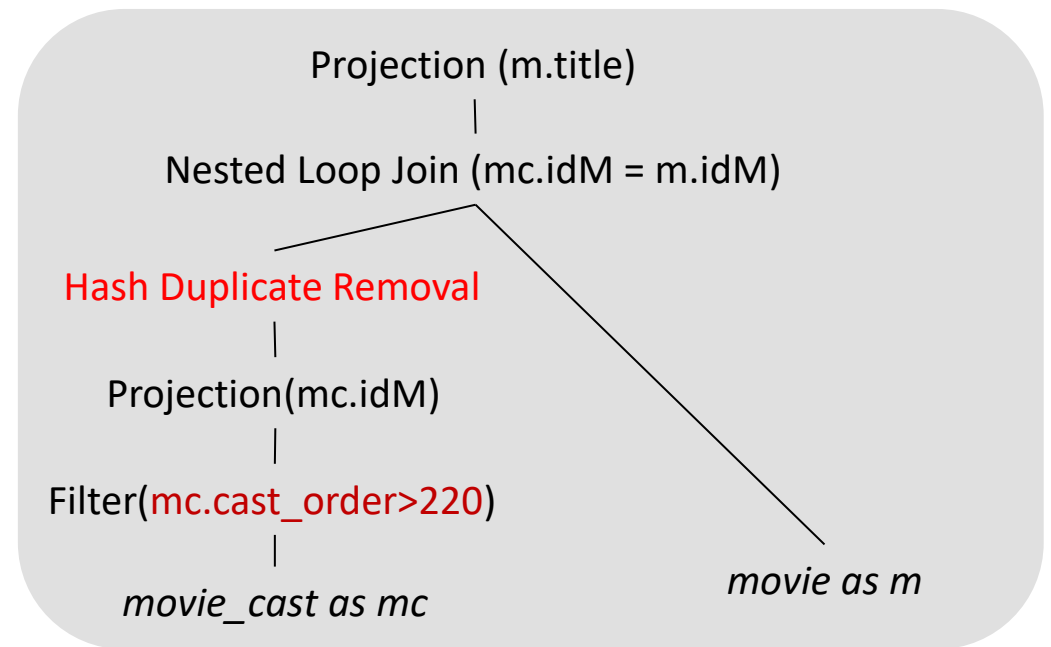
Semi-junção

- Com o Nested Loop Left Sem Join, o filtro fica no lado interno da junção
 - Seria interessante movê-lo para o lado externo, para reduzir o custo das junções
 - Mas o Nested Loop Left Semi Join não permite isso



Semi-junção

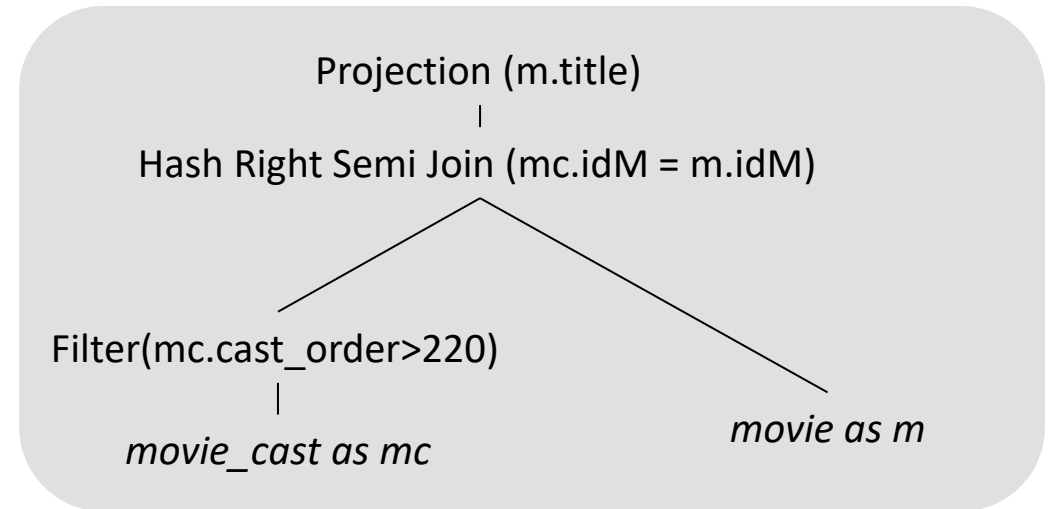
- **Alternativa 1:** Nested Loop normal com remoção de duplicatas
- Para cada `movie_cast` que satisfizer o filtro
 - A remoção de duplicatas preserva apenas idMs únicos
 - Isso garante que, na etapa de junção
 - cada filme tenha no máximo uma correspondência



Semi-junção

- **Alternativa 2: Hash Right Semi Join**

- Para cada movie_cast que satisfizer o filtro
 - o idM é verificado na tabela hash de filmes
- Se existir um filme correspondente
 - O filme é marcado na tabela hash
- No final
 - Só os filmes marcados são retornados



Sumário

- Junção externa
- Semi-junção
 - Semi-junção vs junção regular
- Anti-junção
 - anti-junção vs junção externa

Semi-junção vs junção regular

- Como vimos, uma semi-junção é expressa por meio de uma subconsulta

```
SELECT idM, title  
FROM movie WHERE idM IN  
      (SELECT idM FROM movie_cast)
```

Com semi-junção

Semi-junção vs junção regular

- Uma semi-junção também pode ser expressa como uma **junção regular**

```
SELECT idM, title  
FROM movie WHERE idM IN  
      (SELECT idM FROM movie_cast)
```

Com semi-junção



```
SELECT DISTINCT idM, title  
FROM movie m  
JOIN movie_cast mc ON m.idM = mc.idM
```

Sem semi-junção

Semi-junção vs junção regular

- Quando uma junção regular é equivalente a uma semi-junção?
 - O SELECT retorna apenas dados da parte principal(movie)
 - As colunas retornadas são únicas na parte principal (idM, title)
 - O DISTINCT remove múltiplas ocorrências para um mesmo filme
 - Para os filmes que tenham vários movie_casts

```
SELECT idM, title  
FROM movie WHERE idM IN  
      (SELECT idM FROM movie_cast)
```

Com semi-junção



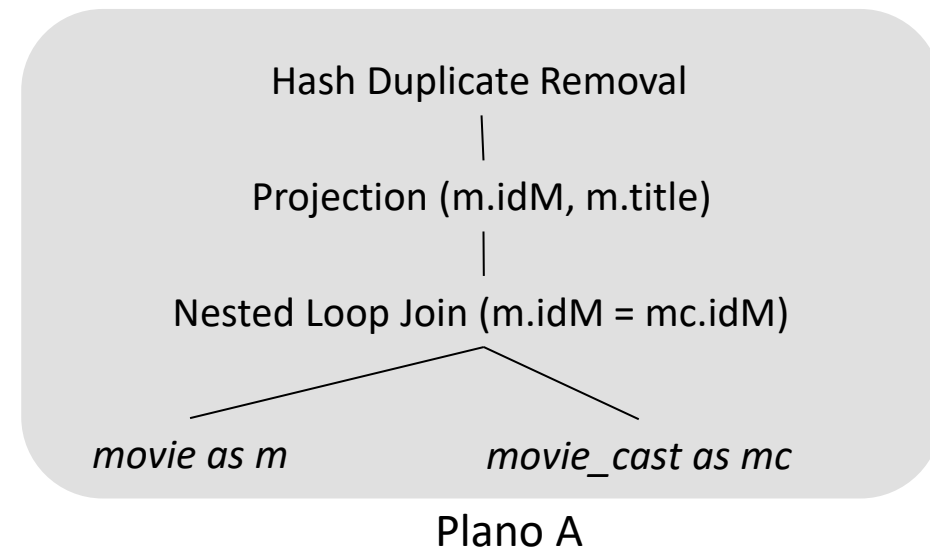
```
SELECT DISTINCT idM, title  
FROM movie m  
JOIN movie_cast mc ON m.idM = mc.idM
```

Sem semi-junção

Semi-junção vs junção regular

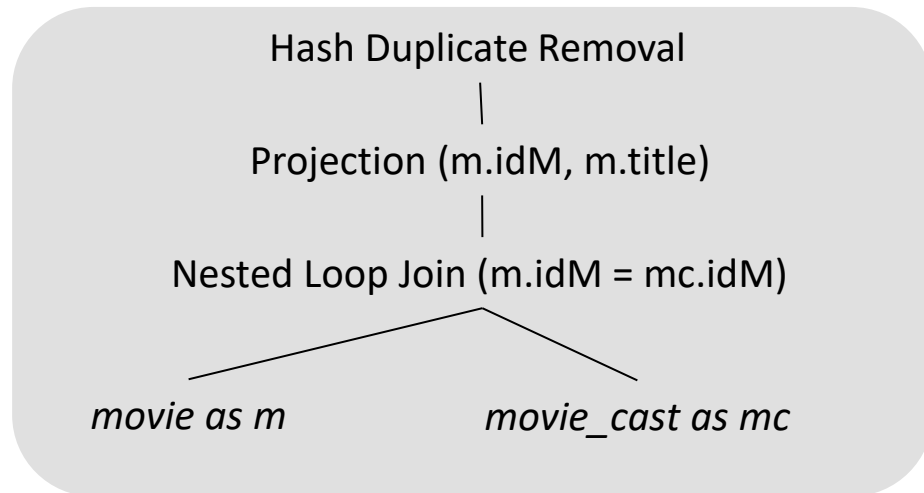
- Plano de execução usando junção regular
 - Um filme cruza com todos os seus movie_casts
 - Os cruzamentos sobressalentes são descartados
 - Com o operador Hash Duplicate Removal

```
SELECT DISTINCT idM, title  
FROM movie m JOIN movie_cast mc  
ON m.idM = mc.idM
```

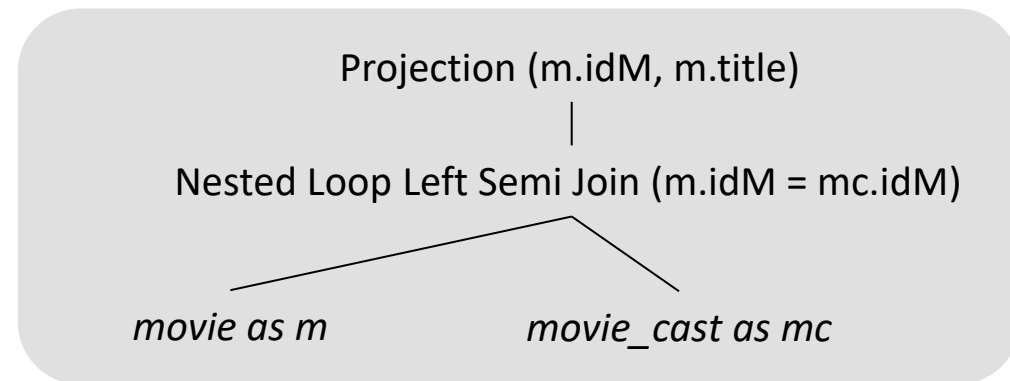


Semi-junção vs junção regular

- Qual estratégia é melhor?



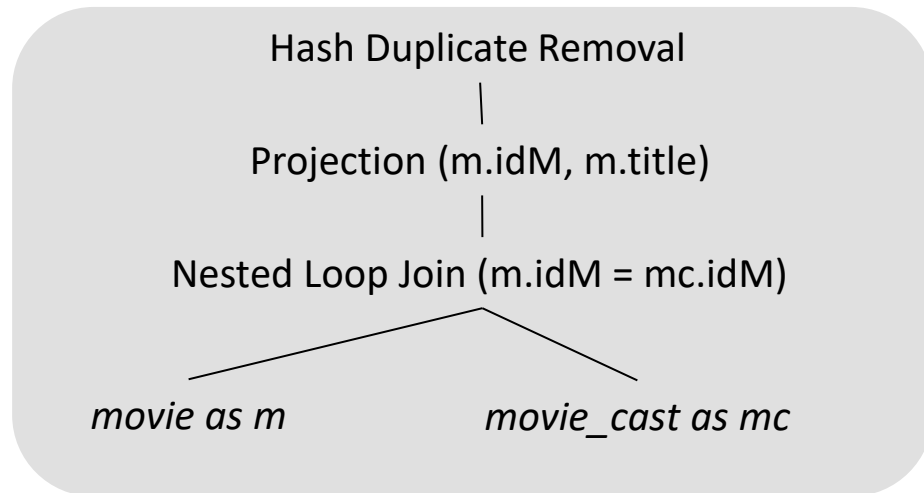
Plano A (usando junção regular)



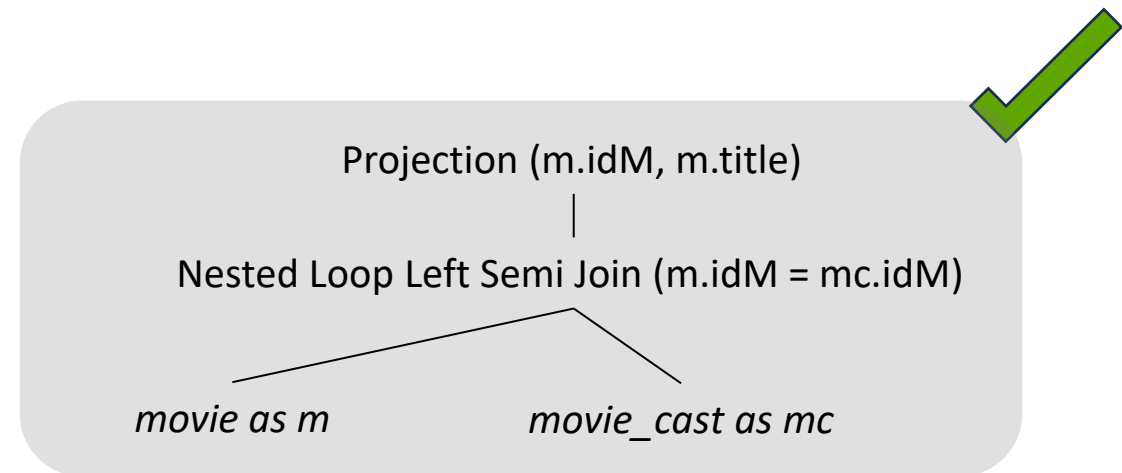
Plano B: usando semi-junção

Semi-junção vs junção regular

- O Plano B é melhor
 - Para cada filme, ocorre apenas uma verificação no lado interno
 - Em nenhum momento é realizado algum cruzamento
 - Isso evita o overhead de remoção de duplicatas



Plano A (usando junção regular)



Plano B: usando semi-junção

Semi-junção vs junção regular

- Na dúvida, use subconsulta
 - deixa claro para o SGBD que se trata de uma semi-junção
 - proporciona ao otimizador uma série de caminhos alternativos até a resposta
 - Quando uma semi-junção é especificada sem usar subconsulta
 - o SGBD pode não reconhecer que se trata de uma semi-junção e deixa de seguir caminhos que seriam interessantes
- Por via das dúvidas, convém testar o plano gerado pelo uso da junção regular

Sumário

- Junção externa
- Semi-junção
 - Semi-junção vs junção regular
- Anti-junção
 - anti-junção vs junção externa

Anti-junção

- Considerando que uma junção tem duas partes
 - parte principal
 - parte secundária
- Em uma anti-junção
 - Apenas tuplas da parte principal são retornadas
 - Cada tupla pode ser retornada apenas uma vez
 - Caso **não** haja correspondências com a parte secundária

Anti-junção

- Em SQL, **anti-junções** são expressas na forma de **subconsultas**
 - A parte principal é a consulta externa
 - A parte secundária é a subconsulta
- Os exemplos abaixo mostram duas formas de uso (**NOT EXISTS** e **NOT IN**)
 - Parte principal: movie
 - Parte secundária: movie_cast

```
SELECT title
FROM movie m WHERE NOT EXISTS
    (SELECT 1 FROM movie_cast mc
     WHERE m.idM = mc.idM )
```

```
SELECT title
FROM movie WHERE idM NOT IN
    (SELECT idM FROM movie_cast)
```


Anti-junção

- NOT EXISTS
 - a subconsulta contém uma coluna de correlação (**m.idM**)
 - Não importa o que a subconsulta retorna
 - Basta verificar se algo está sendo retornado
 - No exemplo, o retorno é uma constante (**1**)

```
SELECT title
FROM movie m WHERE NOT EXISTS
    (SELECT 1 FROM movie_cast mc
     WHERE m.idM = mc.idM )
```

Anti-junção

- NOT IN
 - A subconsulta é independente da consulta principal
 - Não possui coluna de correlação
 - As correspondências são feitas comparando colunas da consulta principal com o retorno da subconsulta

```
SELECT title  
FROM movie WHERE idM NOT IN  
    (SELECT idM FROM movie_cast)
```

Anti-junção

- Comparação
 - Operadores IN e EXISTS
 - Usados para expressar semi-junções
 - Operadores NOT IN e NOT EXISTS
 - Usados para expressar anti-junções
- Apesar de poderem ser usados para o mesmo fim
 - NOT IN e NOT EXISTS **se comportam de maneira diferente**

Anti-junção

- As estratégias de anti-junção podem ser classificadas em
 - Left Anti Join
 - A parte principal fica do lado externo da junção
 - Algoritmos
 - Nested Loop Left Anti Join
 - Hash Left Anti Join
 - Merge Left Anti Join
 - Right Anti Join
 - A parte principal fica do lado interno da junção
 - Algoritmos
 - Hash Right Anti Join
 - Merge Right Anti Join

Anti-junção

- **Exemplo 1:** título de filmes que **não** contenham membros do elenco registrados

```
SELECT title
FROM movie m WHERE NOT EXISTS
    (SELECT 1 FROM movie_cast mc
     WHERE m.idM = mc.idM )
```

- Como essa consulta é representada internamente?

Anti-junção

- A consulta é um exemplo de anti-junção
 - Retornar títulos de filme que **não** contenham membros do elenco
 - Cada filme que satisfaça essa restrição é retornado uma única vez
 - Nenhuma coluna de movie_cast é retornada

```
SELECT title
FROM movie m WHERE NOT EXISTS
    (SELECT 1 FROM movie_cast mc
     WHERE m.idM = mc.idM )
```

Anti-junção

- Neste caso, a consulta também poderia ser expressa usando **NOT IN**
- Obs. Nem sempre o NOT IN resultado em uma consulta equivalente
 - A presença de valores nulos para idM afeta o resultado

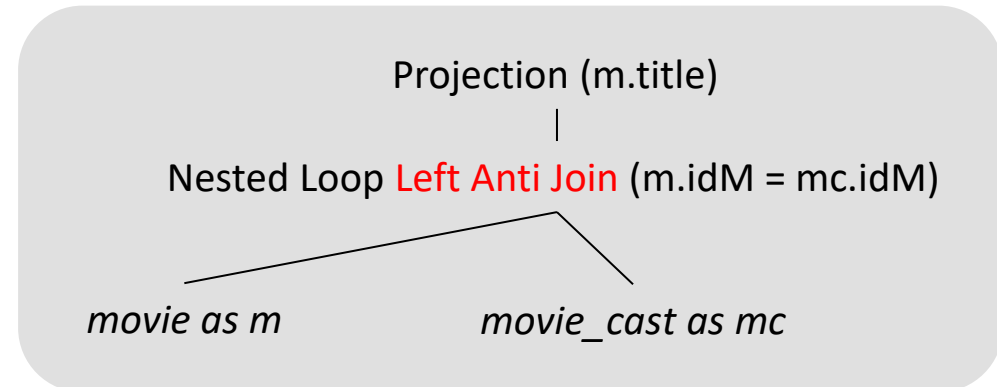
```
SELECT title
FROM movie m WHERE NOT EXISTS
    (SELECT 1 FROM movie_cast mc
     WHERE m.idM = mc.idM )
```

```
SELECT title
FROM movie WHERE idM NOT IN
    (SELECT idM FROM movie_cast)
```

Anti-junção

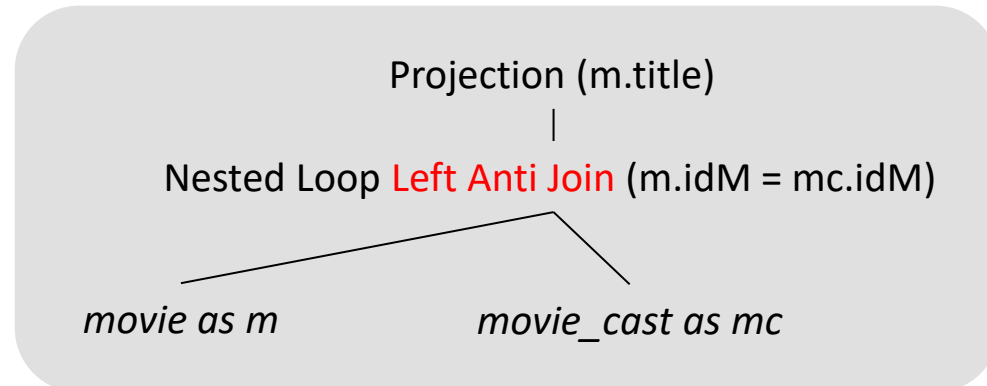
- O plano abaixo usa um **Left Anti Join**
 - A parte principal fica no lado externo da junção
 - Apenas registros da parte principal podem ser retornados

```
SELECT title
FROM movie m WHERE NOT EXISTS
  (SELECT 1 FROM movie_cast mc
   WHERE m.idM = mc.idM )
```



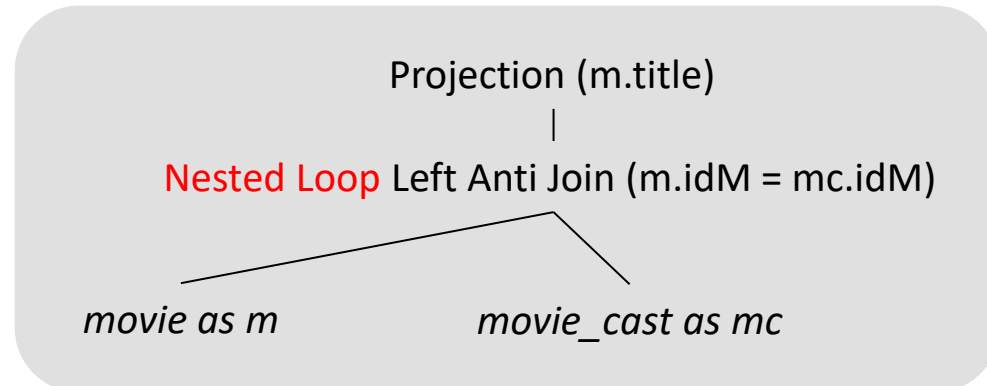
Anti-junção

- Funcionamento do **Left Anti Join**
 - A junção não busca correspondências
 - Apenas verifica se elas existem
 - Por isso pouco importa o que foi usado no SELECT da subconsulta



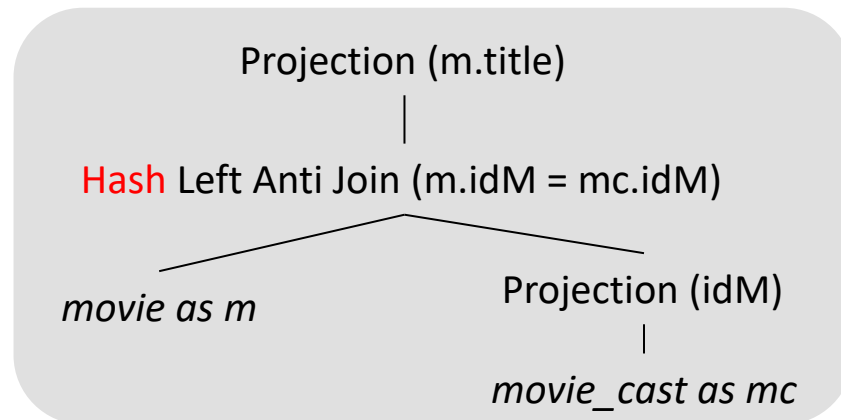
Anti-junção

- Existem vários algoritmos de Left Anti Join
 - O plano abaixo usa um **Nested Loop**



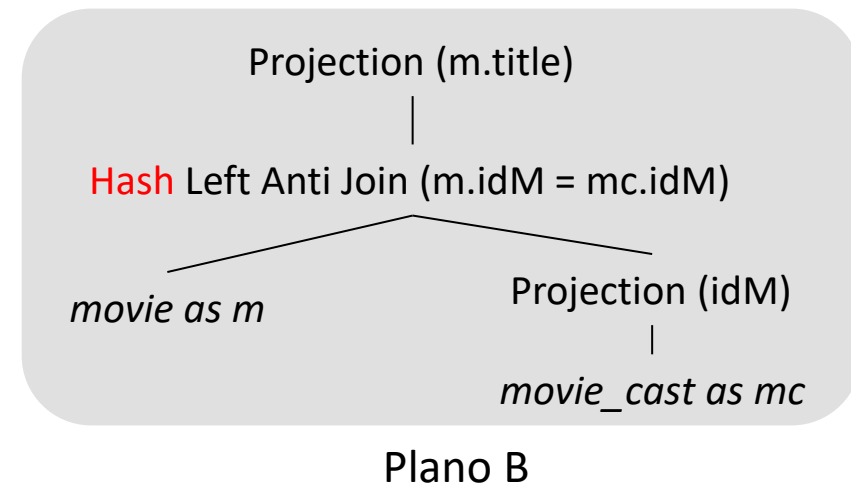
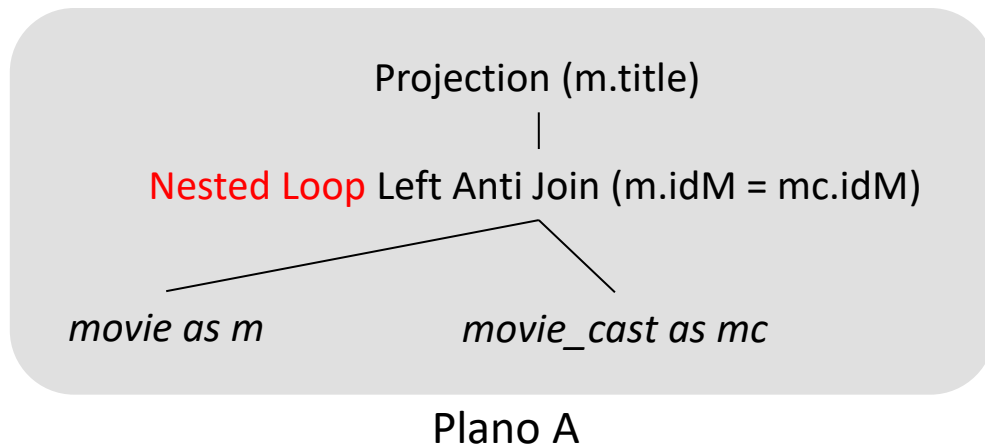
Anti-junção

- Já o plano abaixo usa um **Hash Join**
 - O Hash Join usa uma tabela hash no lado interno
 - A verificação do anti Join é feita sobre a tabela hash



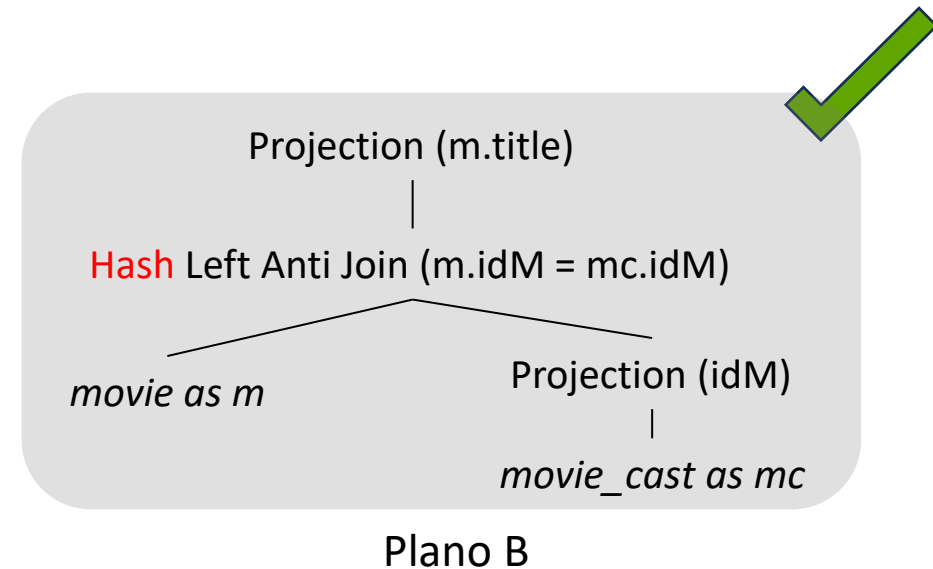
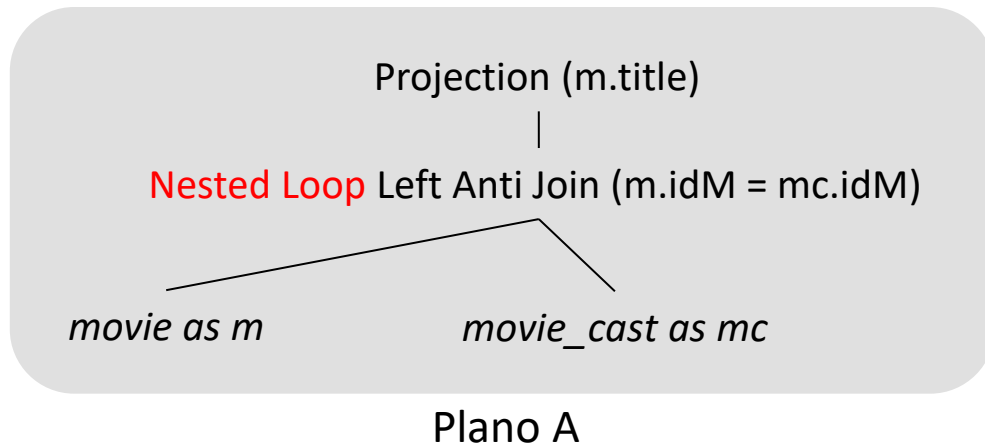
Anti-junção

- Os planos A e B mostram as duas possibilidades
 - Plano A: com Nested Loop Left Anti Join
 - Plano B: com Hash Left Anti Join
- Qual deles é mais eficiente?



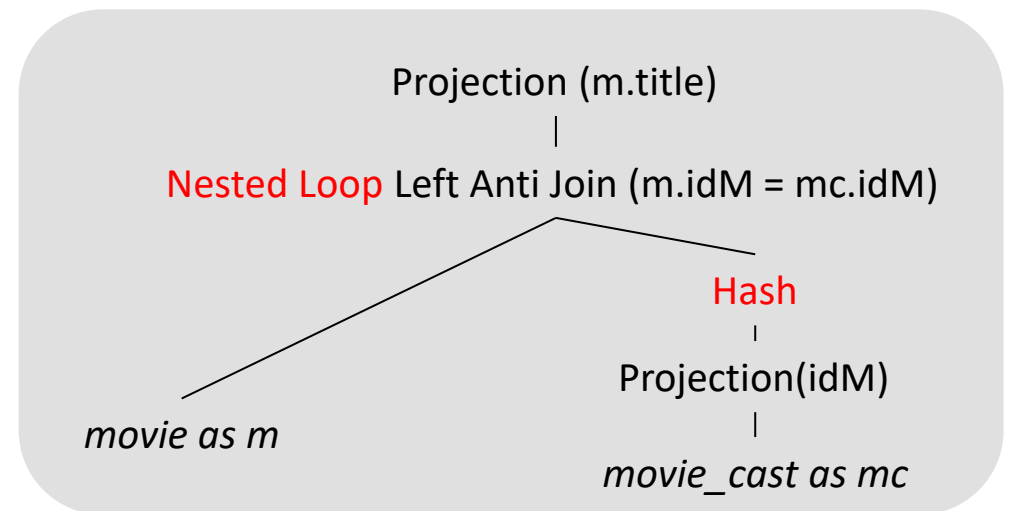
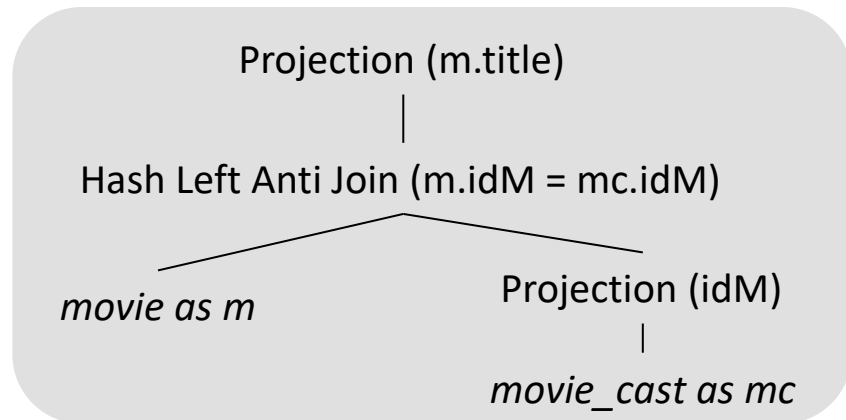
Anti-junção

- Plano B é mais eficiente
 - A busca da junção é feita em uma tabela hash
 - Contudo, consome memória



Anti-junção

- Curiosidade: No **DBest**, o operador Hash Left Anti Join pode ser substituído por uma combinação de dois operadores
 - Nested Loop Left Anti Join
 - Hash



Anti-junção

- Limitação das estratégias **Left Anti Join**
 - a parte principal **sempre** fica no lado externo da junção
- Se a consulta tiver um filtro seletivo na parte secundária
 - Não será possível movê-lo para o lado externo
- Nesses casos, os otimizadores de consulta podem disponibilizar outros algoritmos de junção
 - Exemplo
 - **Hash Right Anti Join**

Anti-junção

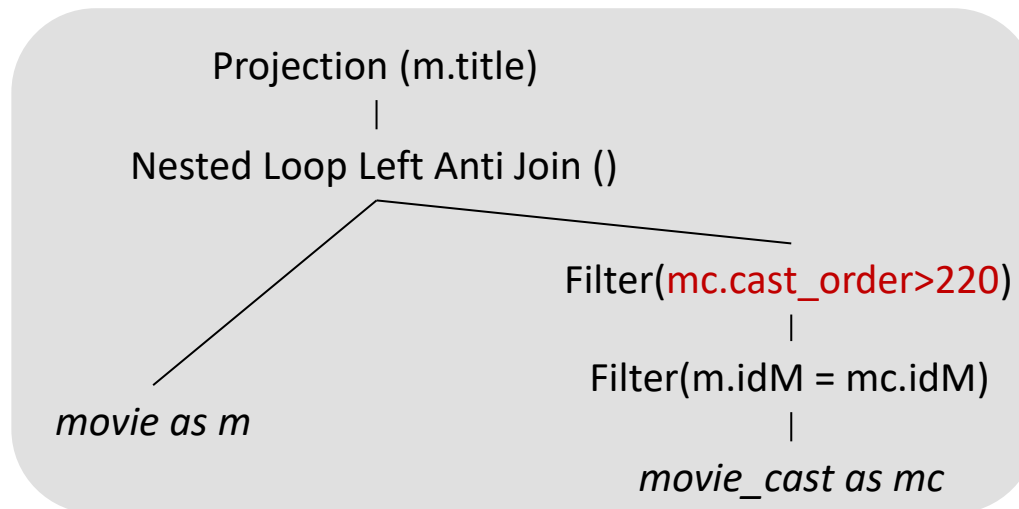
- **Exemplo 2:** título de filmes que **não** contenham mais do que 220 personagens

```
SELECT title  
FROM movie WHERE idM NOT IN  
    (SELECT idM FROM movie_cast  
     WHERE cast_order > 220)
```

- Consulta escrita com NOT IN, mas também poderia ser com NOT EXISTS
- Existe um filtro seletivo sobre a parte secundária

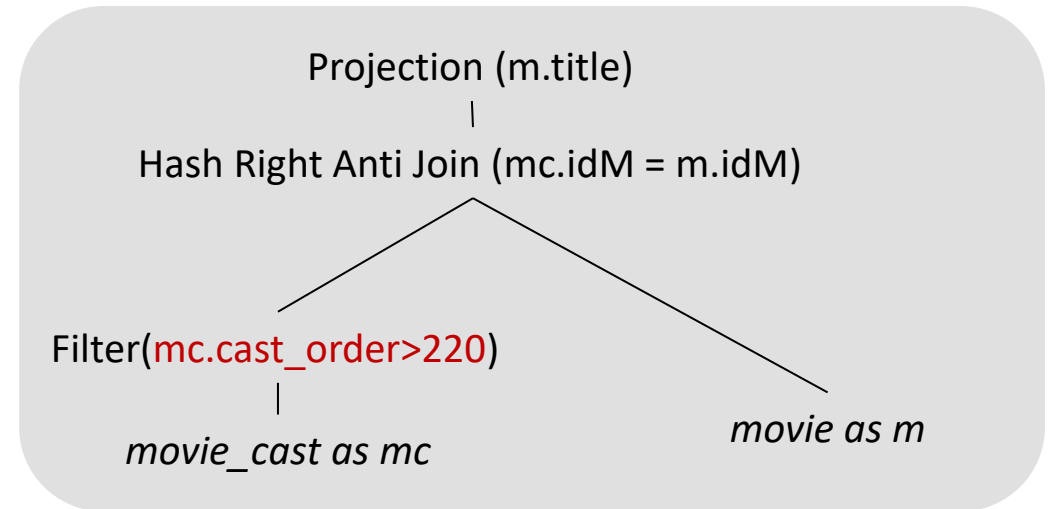
Anti-junção

- Com o Nested Loop Left Anti Join, o filtro ficará no lado interno da junção
 - Seria interessante movê-lo para o lado externo, para reduzir o custo das junções
 - Mas o left semi Join não permite isso



Anti-junção

- Alternativa: Hash Right Anti Join
 - Para cada movie_cast que satisfizer o filtro
 - o idM é verificado na tabela hash de filmes
 - Se existir um filme correspondente
 - O filme é marcado na tabela hash
 - No final
 - os filmes **não** marcados são retornados



Sumário

- Junção externa
- Semi-junção
 - Semi-junção vs junção regular
- Anti-junção
 - anti-junção vs junção externa

Anti-junção vs junção externa

- Como vimos, uma anti-junção é expressa por meio de uma subconsulta

```
SELECT title  
FROM movie WHERE title NOT IN  
    (SELECT c_name FROM movie_cast)
```

Com anti-junção

Anti-junção vs junção externa

- Uma anti-junção também pode ser expressa como uma junção externa

```
SELECT title  
FROM movie WHERE title NOT IN  
    (SELECT c_name FROM movie_cast)
```

Com anti-junção



```
SELECT title  
FROM movie m  
LEFT JOIN movie_cast mc ON m.idM = mc.idM  
WHERE mc.idM IS NULL
```

Sem anti-junção

Anti-junção vs junção externa

- Quando uma junção externa é equivalente a uma anti-junção?
 - O SELECT retorna apenas dados do lado principal(movie)
 - A cláusula WHERE remove todos os filmes que tenham associação com movie_cast
 - mc.idM is NULL

```
SELECT title  
FROM movie WHERE title NOT IN  
  (SELECT c_name FROM movie_cast)
```

Com anti-junção



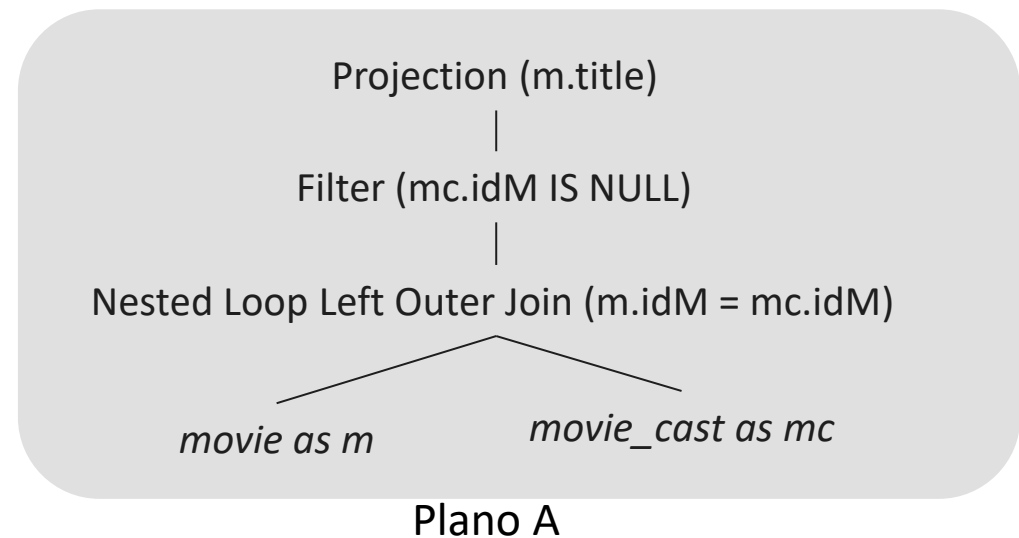
```
SELECT title  
FROM movie m  
LEFT JOIN movie_cast mc ON m.idM = mc.idM  
WHERE mc.idM IS NULL
```

Sem anti-junção

Anti-junção vs junção externa

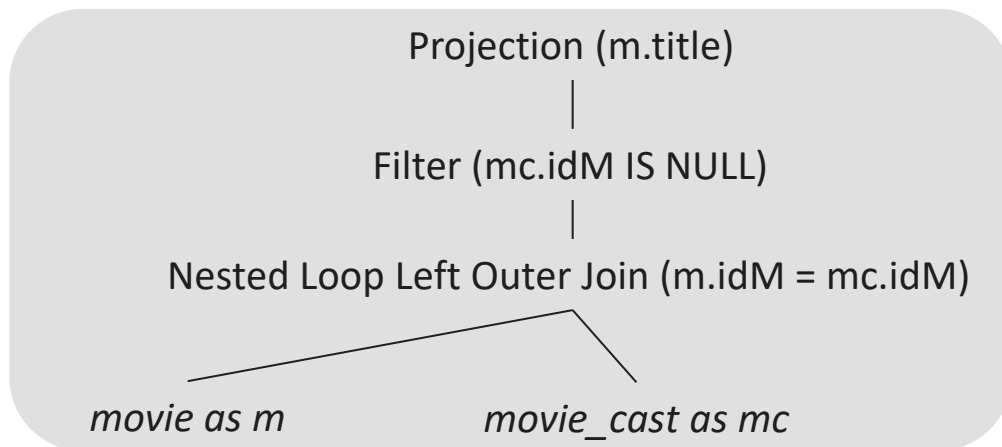
- No plano de execução usando junção externa
 - Um filme precisa cruzar com todos os seus movie_casts correspondentes
 - Filmes sem correspondência tem colunas preenchidas com nulo
 - Em seguida, todas as correspondências identificadas são descartadas
 - Ou seja, todo o trabalho de localização das correspondências é desfeito

```
SELECT title
FROM movie m
LEFT JOIN movie_cast mc ON m.idM = mc.idM
WHERE mc.idM IS NULL
```

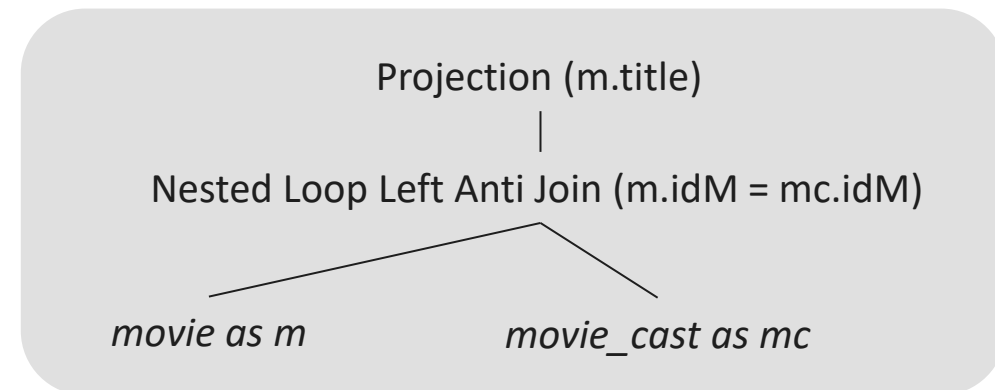


Anti-junção vs junção externa

- Qual estratégia é melhor?



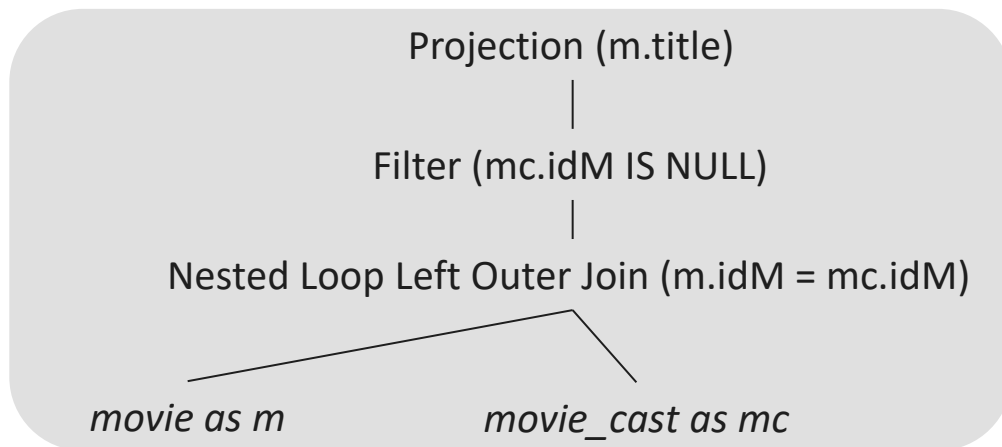
Plano A (com junção externa)



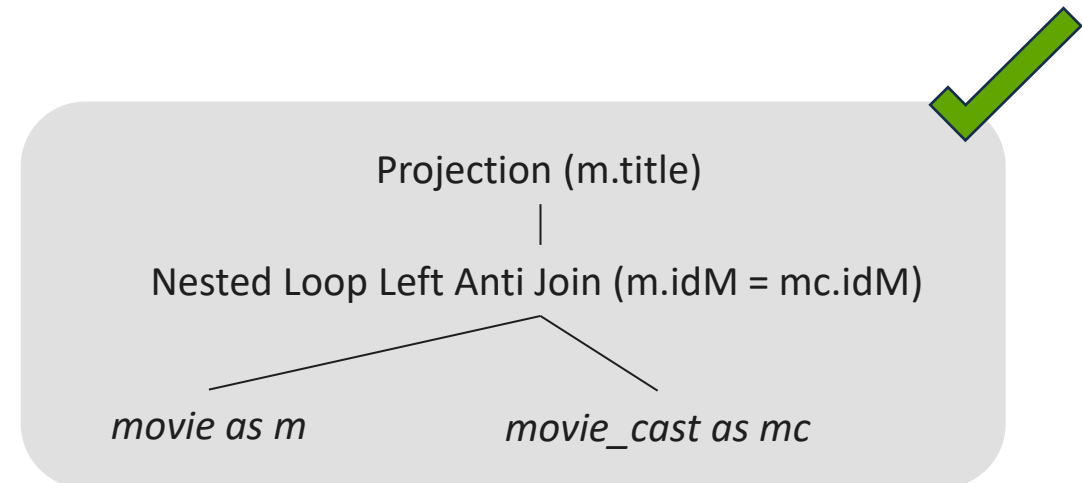
Plano B (com subconsulta)

Anti-junção vs junção externa

- Plano B é melhor
 - Para cada filme, ocorre uma verificação no lado interno (exists)
 - Se não existir correspondência, o filme é retornado
 - Em nenhum momento é realizado algum cruzamento
 - Isso evita o overhead de localização e posterior remoção via filtro



Plano A (com junção externa)



Plano B (com subconsulta)

Anti-junção vs junção externa

- Na dúvida, use subconsulta
 - deixa claro para o SGBD que se trata de uma anti-junção
 - proporciona ao otimizador uma série de caminhos alternativos até a resposta
 - Quando uma anti-junção é especificada sem usar subconsulta
 - o SGBD pode não reconhecer que se trata de uma semi-junção e deixa de seguir caminhos que seriam interessantes
- Mesmo assim, convém testar o plano gerado pelo uso da junção externa

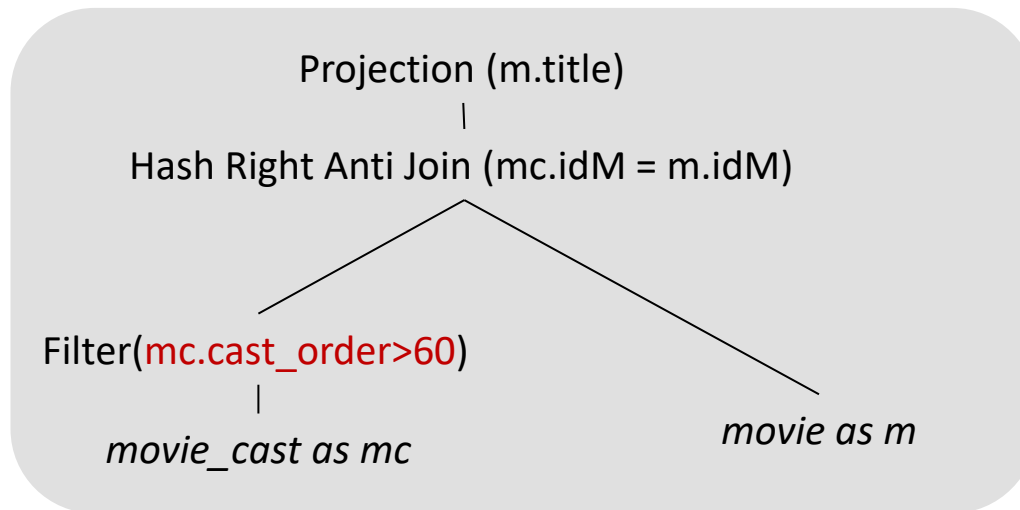
Atividade Individual

- A consulta abaixo pede títulos de filmes que não possuam mais do que 60 membros do elenco.

```
SELECT title
FROM movie WHERE idM NOT IN
    (SELECT idM FROM movie_cast
     WHERE cast_order > 60)
```

Atividade Individual


- O plano abaixo pode ser usado para encontrar filmes que não tenham membros de elenco com mais do que 60 personagens



Plano A

Atividade Individual

- O objetivo é encontrar um plano alternativo (plano B)
 - Esse plano não deve acessar mais páginas do que o plano A
 - E deve apresentar um consumo de memória menor



???????

Plano B