Filtros

Introdução

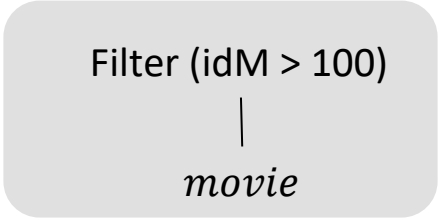
- Os filtros indicados na cláusula WHERE de uma consulta SQL são representados em um plano de execução por um operador especial
- No DBest, esse operador é chamado de Filter
- Nos próximos slides, veremos diversas possibilidades de uso desse operador

Sumário

- Operador Filter
- Uso do índice para resolver filtros
- Etada de complementação
- Filtro sobre o índice primário
- Filtro sobre o índice secundário
  - Com covering index
  - Sem covering index

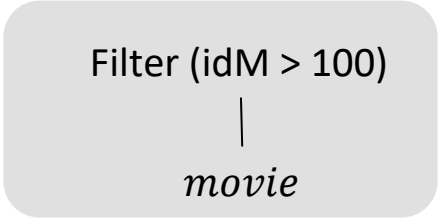
Operador Filter

- O operador Filter é unário: se conecta a um operador de entrada
- Ele define as condições de filtragem que precisam ser satisfeitas
- No exemplo abaixo, o filtro define que, dos filmes recuperados a partir de movie, apenas os que possuem idM maior do que 100 devem ser mantidos



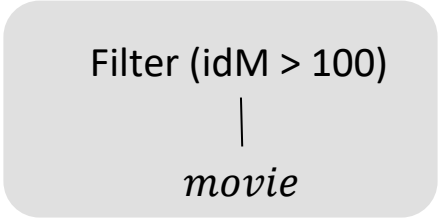
Operador Filter

- Se a entrada de um filtro for um operador indexado
  - o próprio operador conectado se encarrega de resolver o filtro



Operador Filter

- No exemplo abaixo
  - O operador de filtro está conectado ao índice primário movie
    - Por isso, cabe ao índice resolver o filtro e devolver apenas as tuplas solicitadas
  - Esse índice tem idM como chave de busca
    - Por isso, o próprio índice é capaz de localizar as entradas relevantes de forma eficiente

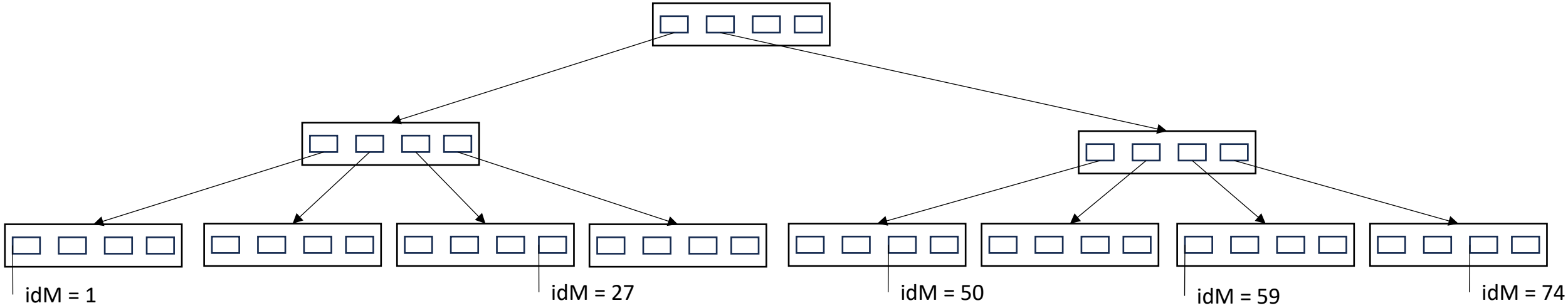


Sumário

- Operador Filter
- Uso do índice para resolver filtros
- Etada de complementação
- Filtro sobre o índice primário
- Filtro sobre o índice secundário
  - Com covering index
  - Sem covering index

Uso de índice para resolver filtros

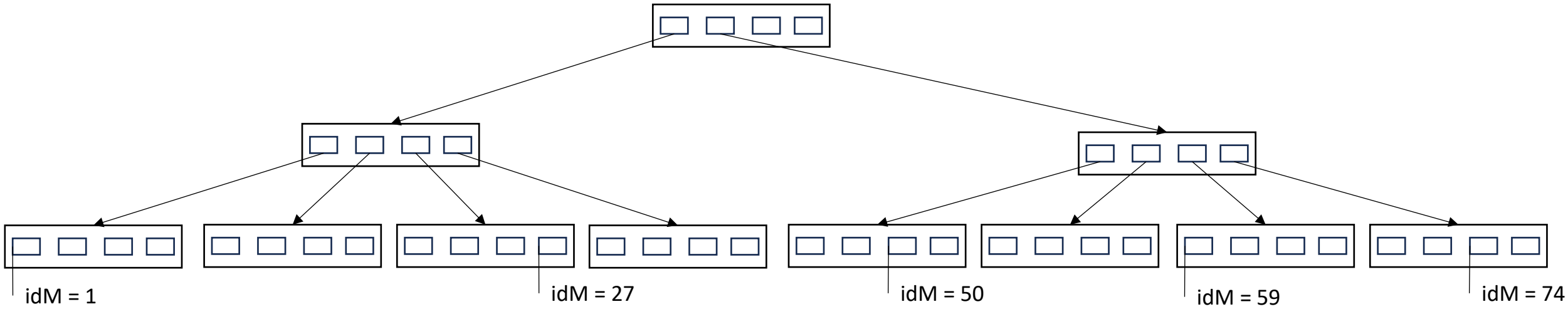
- A seguir veremos exemplos de buscas que podem ser feitas quando um operador Filter está conectado a um índice



Uso de índice para resolver filtros

- O índice abaixo tem chave de busca idM
  - O índice permite buscar registros com base em filtros sobre a coluna idM

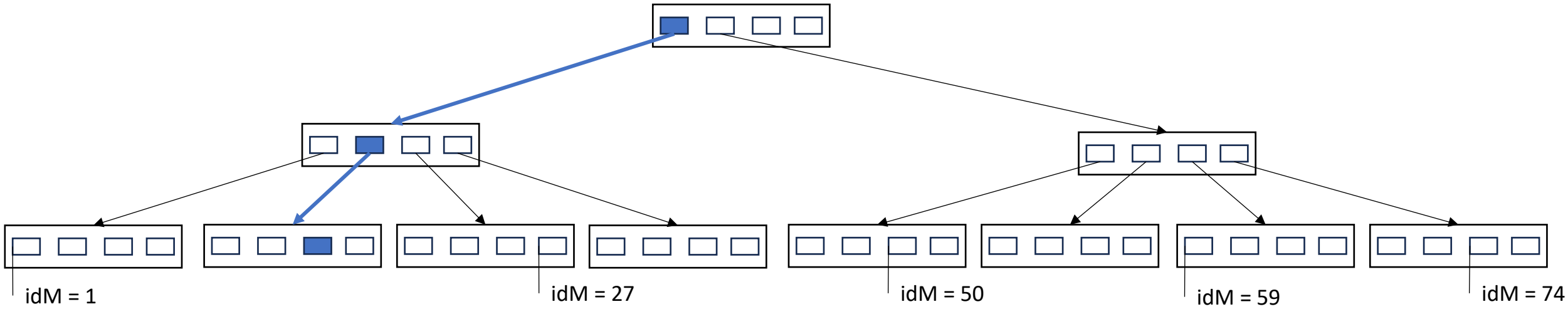
- O nível folha possui todas as chaves de busca, ordenadas por idM
  - Por ser um índice primário (clusterizado), o nível folha armazena o registro completo junto com a chave de busca



Uso de índice para resolver filtros

- Filtro: idM = 9

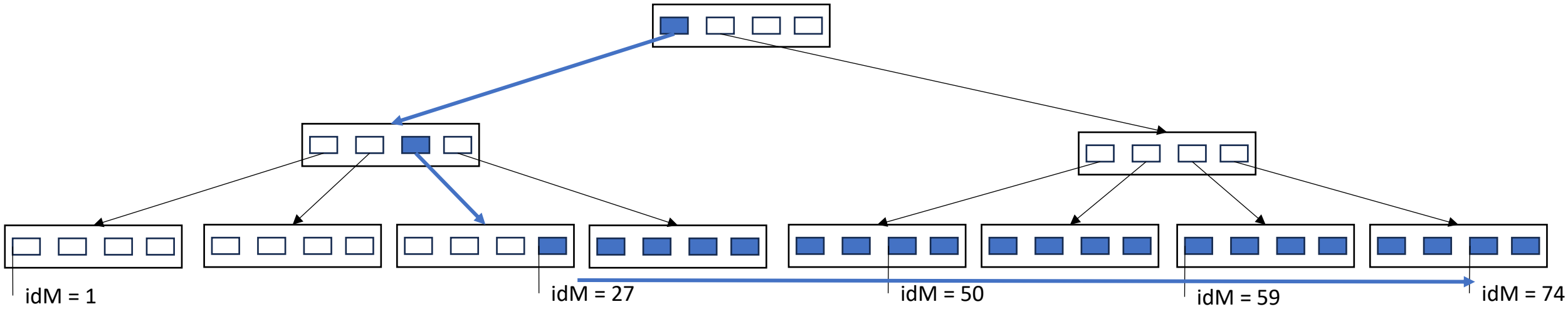
- Como o índice tem idM como chave de busca
  - É realizado um seek (uma busca que se inicia no nível raiz e termina em um nó folha)
  - O seek leva diretamente à entrada que satisfaz a busca



Uso de índice para resolver filtros

- Filtro: idM >= 27

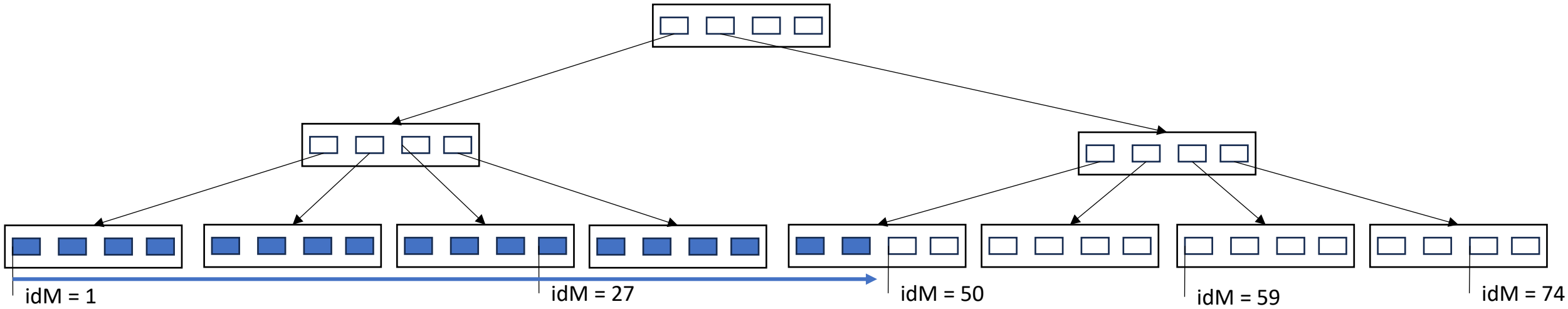
- Como o índice tem idM como chave de busca
  - É realizado um seek, que leva diretamente à primeira entrada que satisfaz o critério de busca
  - Todas as entradas a partir desse ponto são recuperadas



Uso de índice para resolver filtros

- Filtro: idM < 50

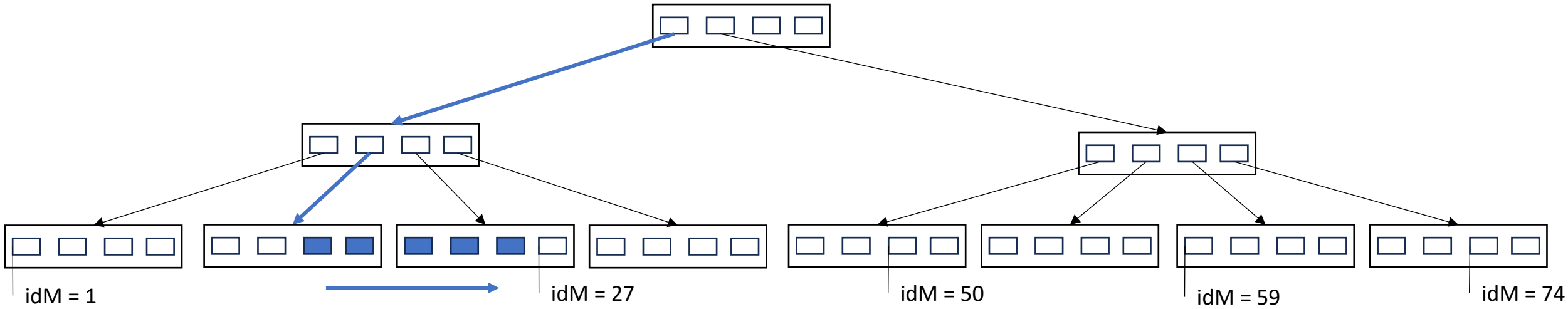
- Como o índice tem idM como chave de busca
  - É realizado um scan começa pelo nó folha mais à esquerda
  - O scan termina quando for encontrada a primeira entrada que não satisfaça o critério



Uso de índice para resolver filtros

- Filtro: idM >= 9 and idM < 27

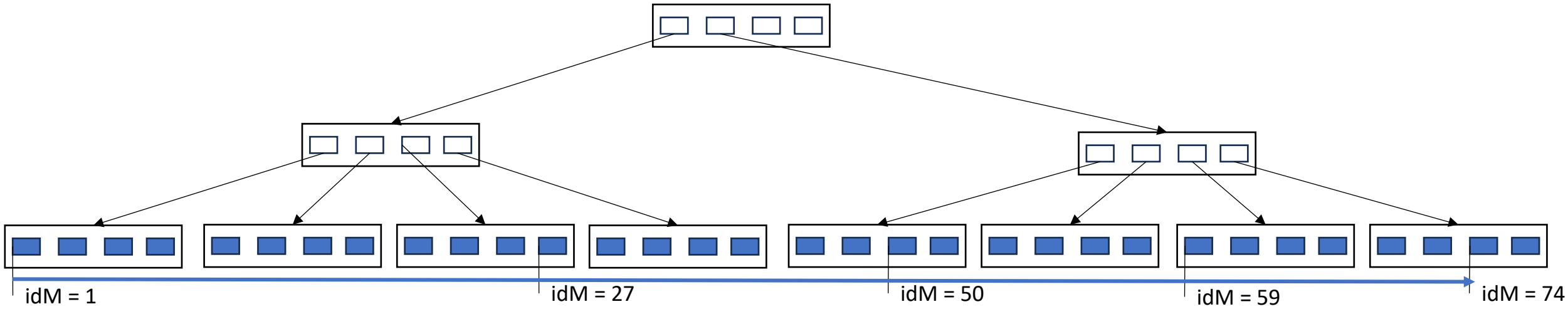
- Como o índice tem idM como chave de busca
  - Um seek leva diretamente à entrada que satisfaz o intervalo inferior.
  - A partir desse ponto, é necessário um scan sobre as entradas consequentes enquanto o intervalo superior for satisfeito



Uso de índice para resolver filtros

- Filtro: year > 2010

- Como o índice não tem year como chave de busca
  - será necessário varrer todos os registros e verificar a condição de filtragem um por um



Sumário

- Operador Filter
- Uso do índice para resolver filtros
- Etapa de complementação
- Filtro sobre o índice primário
- Filtro sobre o índice secundário
  - Com covering index
  - Sem covering index

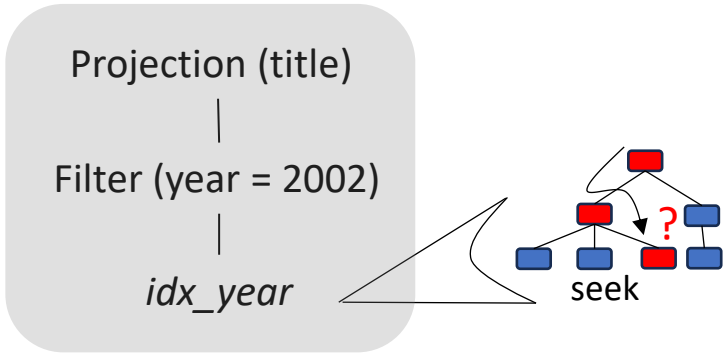
Etapa de Complementação

- Vimos que um índice ajuda a localizar as entradas que satisfazem as condições presentes em um operador Filter

- Mas e o que fazer quando o índice não possui todas as colunas exigidas pela consulta?

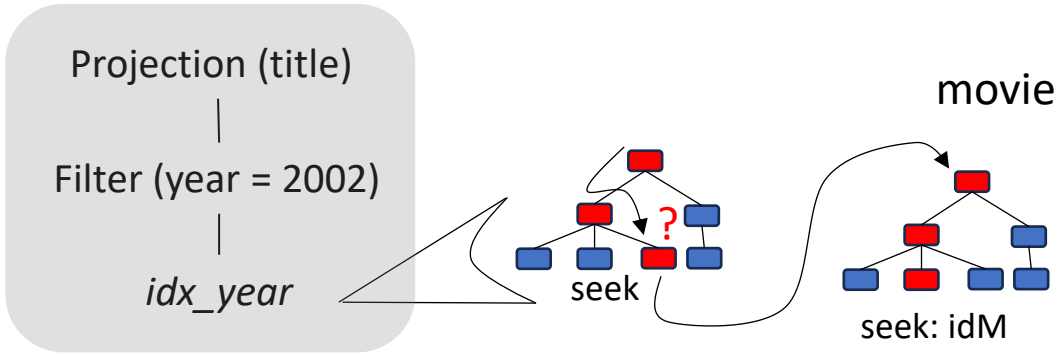
Etapa de Complementação

- No plano abaixo
  - O operador de filtro está conectado ao índice idx_year
  - Como esse índice tem year como chave de busca, é realizado um seek na B+tree para verificar a condição de filtro
  - No entanto, a coluna title não está disponível no índice
    - Para recuperá-la, é necessária uma etapa de complementação
      - Nessa etapa, a estrutura contendo os registros precisa ser acessada



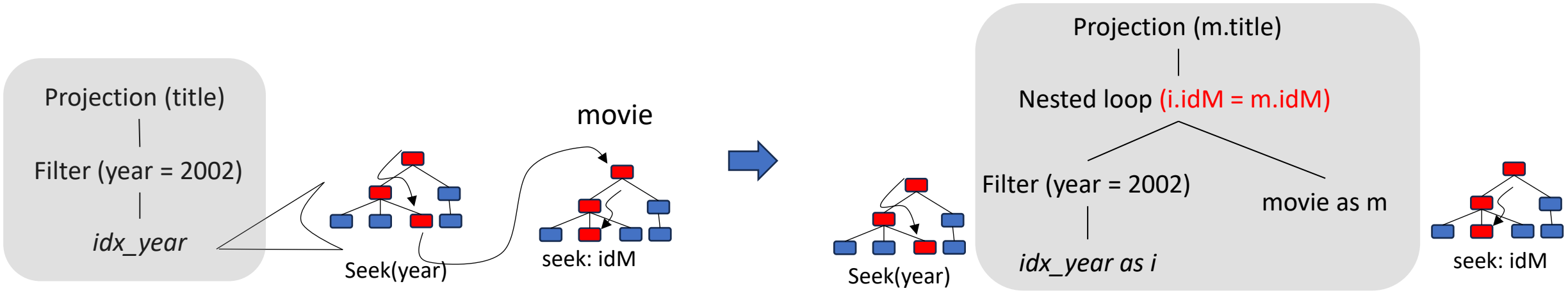
Etapa de Complementação

- A complementação depende de como as tabelas estão organizadas
- No DBest
  - O nível folha do índice idx_year guarda a chave primária de movie
  - Necessário acessar o índice primário de movie para obter o registro completo



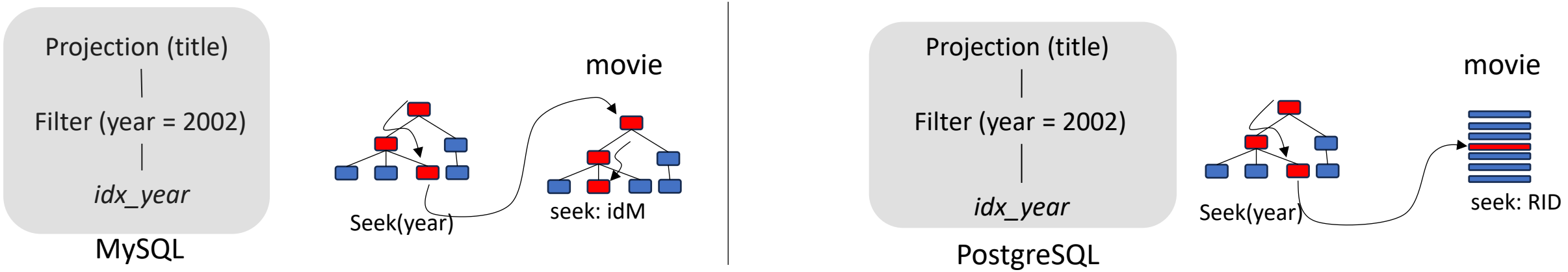
Etapa de Complementação

- Na prática, o DBest incorpora essa etapa no plano recorrendo ao algoritmo de junção Nested Loop Join



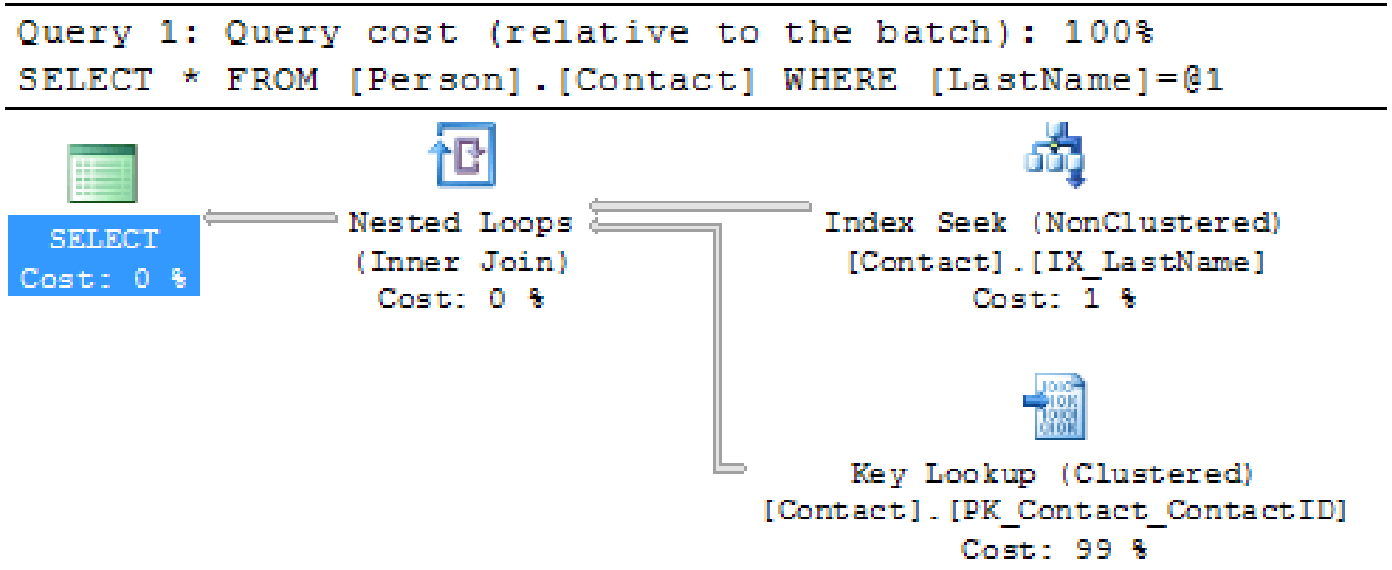
Etapa de Complementação

- O MySQL e o PostgreSQL também precisam acessar as estruturas que contem os registros para realizar a complementação
  - O índice primário, no MySQL
  - A heap, no PostgreSQL
- No entanto, essa etapa ocorre por baixo dos panos
  - O processo envolvido não aparece muito claramente no plano de execução



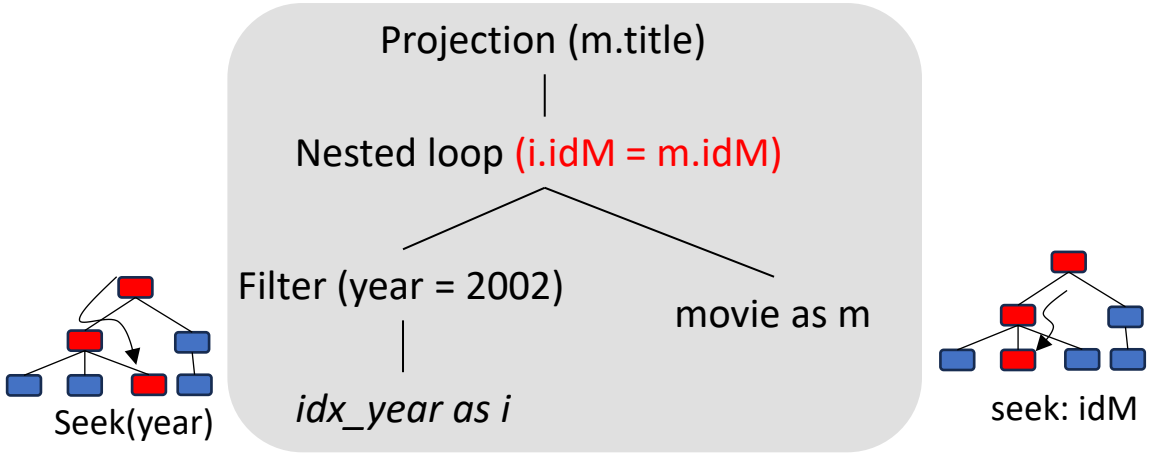
Etapa de Complementação

- Já o SQL Server usa o Nested Loop Join para fazer a complementação



Etapa de Complementação

- A etapa de complementação é executada para cada entrada que satisfizer o filtro
  - E ela leva a um acesso aleatório para recuperar a informação completa
- Por isso, quanto menos seletivo for o filtro
  - Mais custosa será a consulta, devido ao número elevado de acessos aleatórios



Sumário

- Operador Filter
- Uso do índice para resolver filtros
- Etapa de complementação
- Filtro sobre o índice primário
- Filtro sobre o índice secundário
  - Com covering index
  - Sem covering index

Filtro sobre o índice primário

- Exemplo: retornar filmes cujo id seja superior a 100

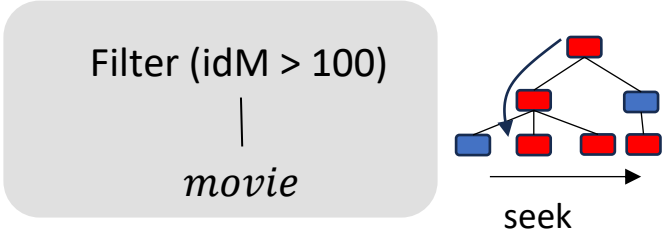
```
SELECT *
FROM movie
WHERE idM > 100
```

- O filtro é sobre o índice primário de movie (idM)

- Todas as colunas devem ser retornadas

Filtro sobre o índice primário

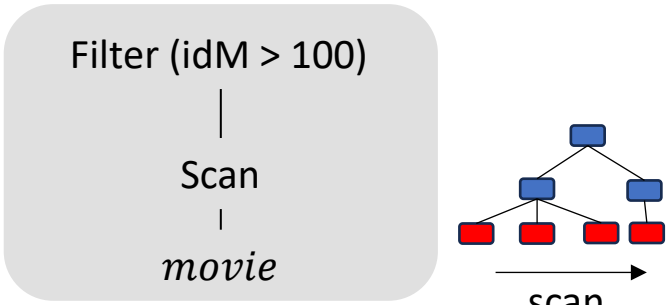
- Plano A
  - O operador de filtro está conectado ao índice primário movie
  - Como esse índice tem idM como chave de busca, é realizado um seek na B+tree para verificar a condição de filtro
  - O SELECT * não torna a consulta mais onerosa
    - No DBest, o índice primário guarda todos os registros no nível folha
    - Ou seja, não é necessário acessar nenhuma outra estrutura para recuperar o registro completo



Plano A

Filtro sobre o índice primário

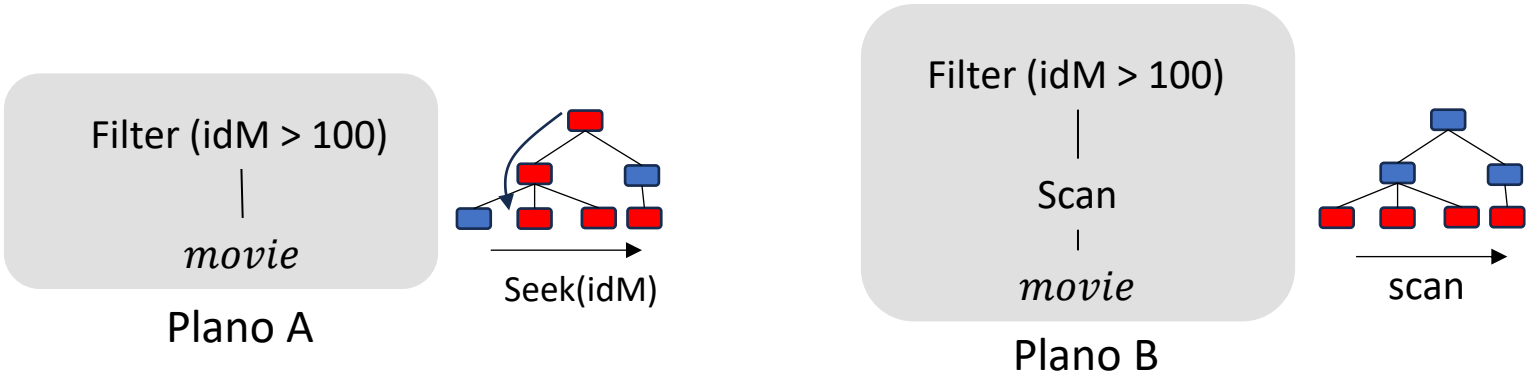
- Plano B
  - O operador de filtro está conectado a um operador scan
  - O scan é usado para evitar que um seek seja usado
  - Em vez do seek, é realizado um scan no nível folha da B+tree



Plano B

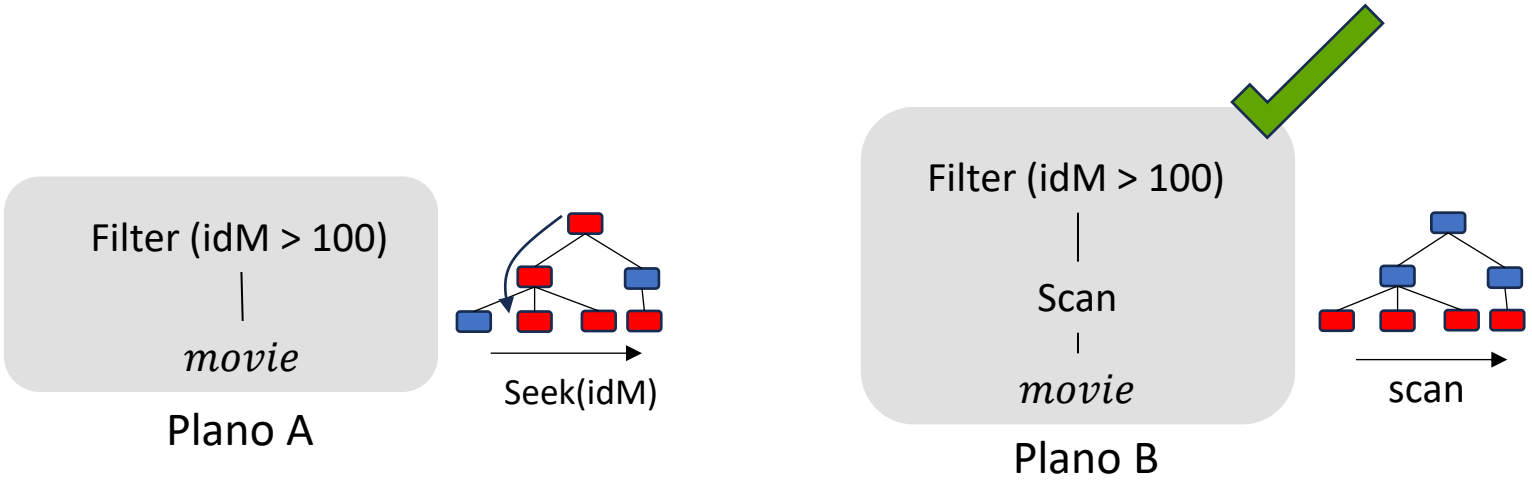
Filtro sobre o índice primário

- Qual é melhor?



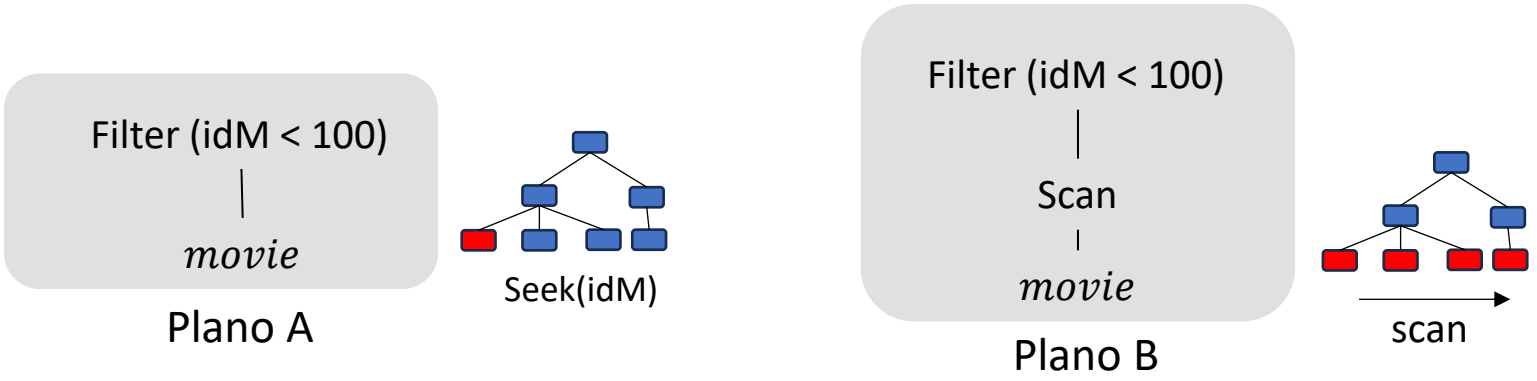
Filtro sobre o índice primário

- O plano B é melhor
  - No plano A, quase todos os nós no nível folha são retornados
  - Nesse caso, o acesso aos nós intermediários feitos pelo seek se tornam um overhead (pequeno, mas existente)
  - É melhor realizar um scan sobre a tabela



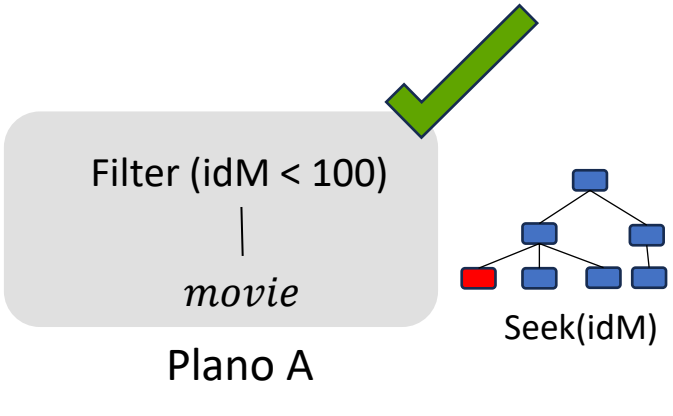
Filtro sobre o índice primário

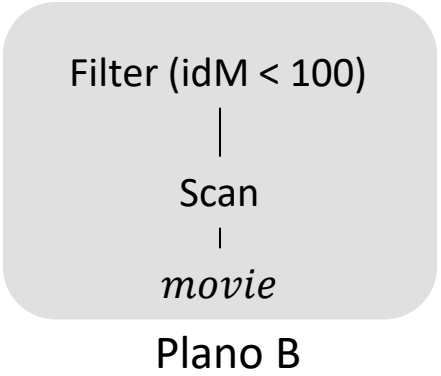
- E se o filtro for seletivo? (ex. idM < 100)

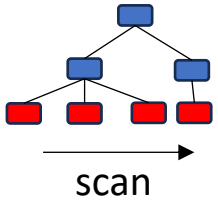


Filtro sobre o índice primário

- O plano A é melhor
  - O seek recupera poucos nós (páginas) enquanto o scan recuperaria todos







Filtro sobre o índice primário

- Em resumo
  - caso o filtro seja sobre o índice primário, compensa usá-lo.
  - Não há necessidade de etapa de complementação
    - Pode-se recuperar todas as colunas da tabela no nível folha do índice
  - Se o filtro for seletivo
    - será feito um index seek.
  - Se o filtro for pouco seletivo
    - será feito um index/table scan.

Sumário

- Operador Filter
- Uso do índice para resolver filtros
- Etapa de complementação
- Filtro sobre o índice primário
- Filtro sobre o índice secundário
  - Com covering index
  - Sem covering index

Filtro sobre o índice secundário – Covering Index

- Exemplo 1: retornar anos de filmes lançados após 2023

```
SELECT year
FROM movie
WHERE year > 2023
```

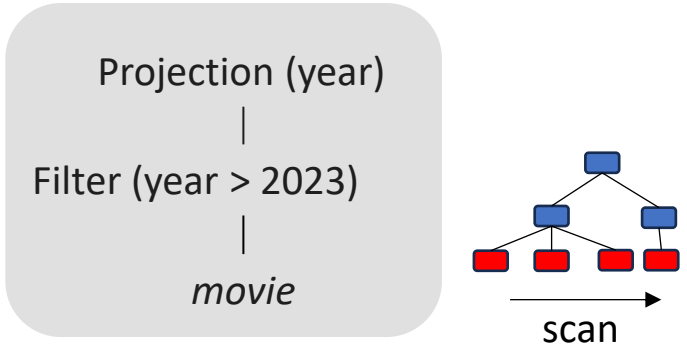
- O Filtro é bastante seletivo

- Existe um índice secundário sobre year (idx_year)

- A única coluna que interessa no SELECT é a que está indexada
  - Ou seja, é um covering index

Filtro sobre o índice secundário – Covering Index

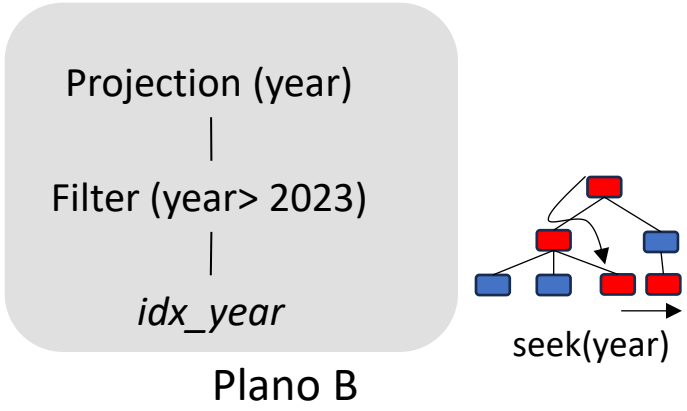
- Plano A
  - O operador de filtro está conectado ao índice primário movie
  - Como esse índice não tem year como chave de busca, é realizado um scan no nível folha da B+tree para verificar a condição de filtro



Plano A

Filtro sobre o índice secundário – Covering Index

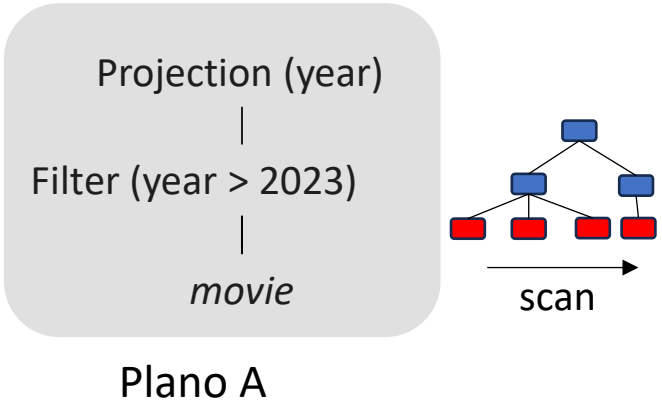
- Plano B
  - O operador de filtro está conectado ao índice secundário idx_year
  - Como esse índice tem year como chave de busca, é realizado um seek na B+tree para verificar a condição de filtro



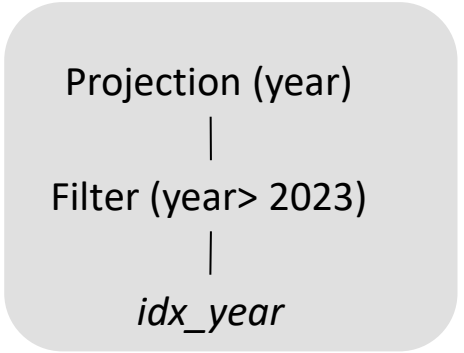
Plano B

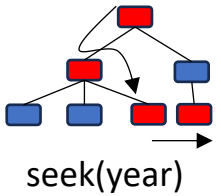
Filtro sobre o índice secundário – Covering Index

- Qual é melhor?



Plano A

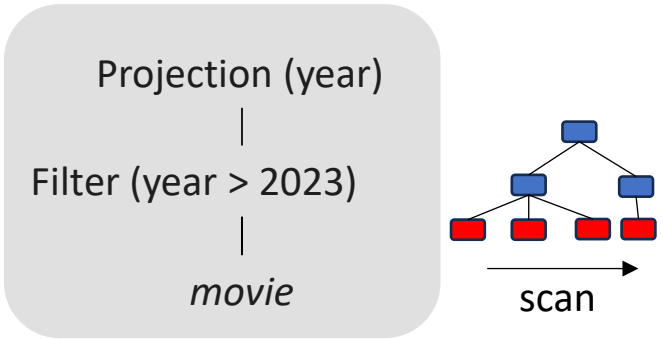




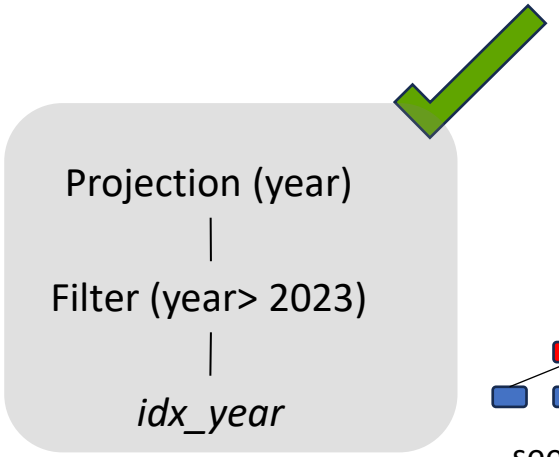
Plano B

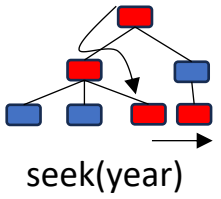
Filtro sobre o índice secundário – Covering Index

- É melhor usar o índice secundário.
- Além da busca ser seletiva, é possível encontrar a resposta usando apenas o índice



Plano A





Plano B

Filtro sobre o índice secundário – Covering Index

- Exemplo 2: retornar anos de filmes lançados após 1905

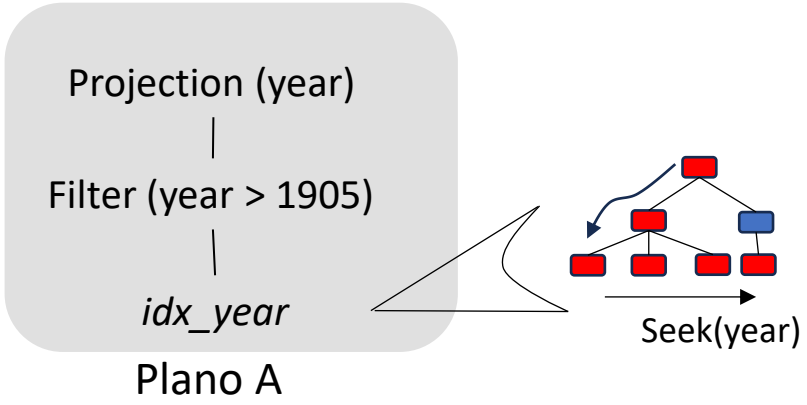
```
SELECT year
FROM movie
WHERE year > 1905
```

- Agora o filtro é pouco seletivo

- Vale a pena usar o índice sobre year?

Filtro sobre o índice secundário – Covering Index

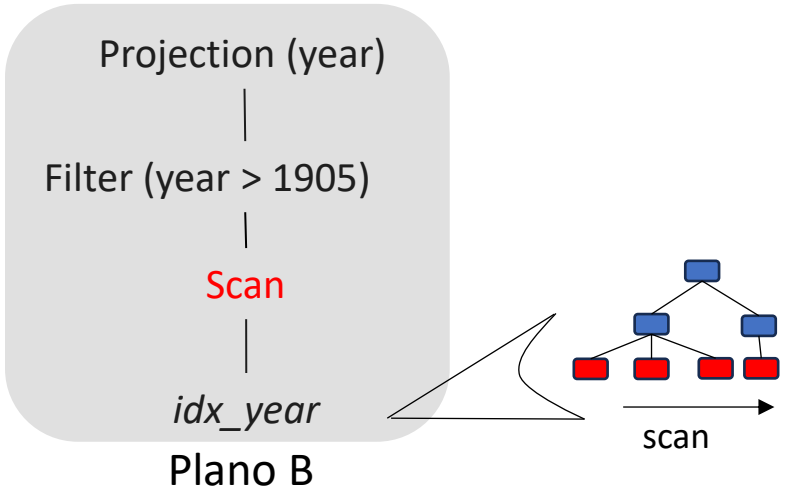
- Plano A
  - O operador de filtro está conectado ao índice secundário idx_year
  - Como o índice tem year como chave de busca, é realizado um seek sobre o índice
  - Porém, o seek localiza o limite inferior no nível folha
    - Todos os nós folha à direita precisam ser acessados



Plano A

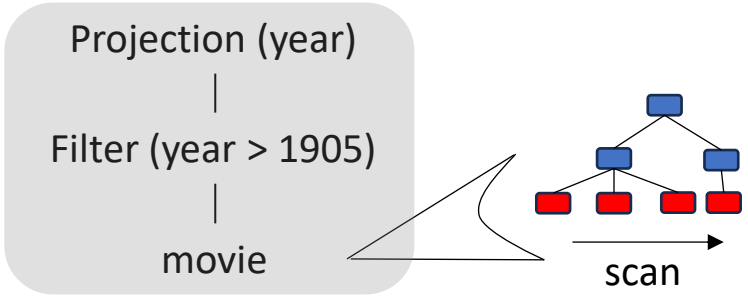
Filtro sobre o índice secundário – Covering Index

- Plano B
  - O operador de filtro está conectado a um operador scan
  - O scan é usado para evitar que um index seek seja usado
  - Em vez do seek, é realizado um index scan no nível folha da B+tree



Filtro sobre o índice secundário – Covering Index

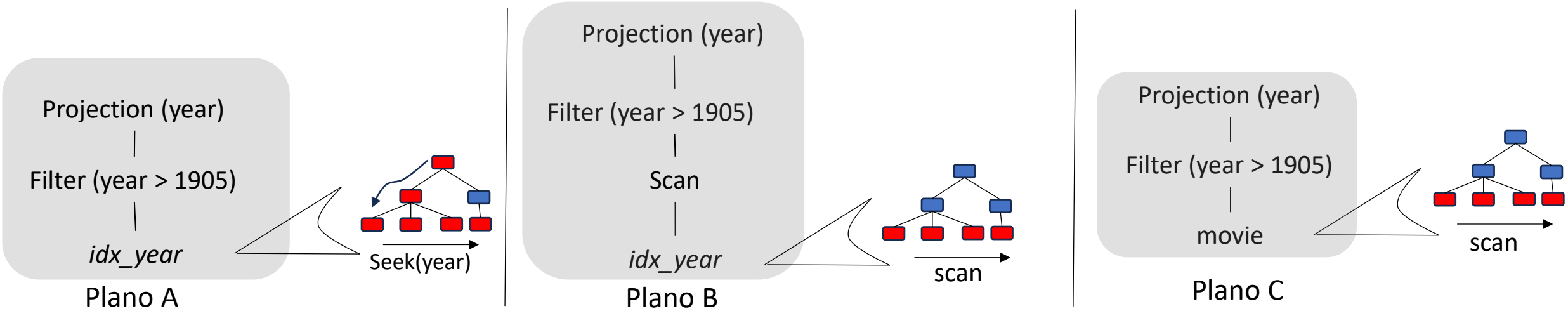
- Plano C
  - O operador de filtro está conectado ao índice primário
  - Como o índice não tem year como chave de busca, é realizado um scan



Plano C

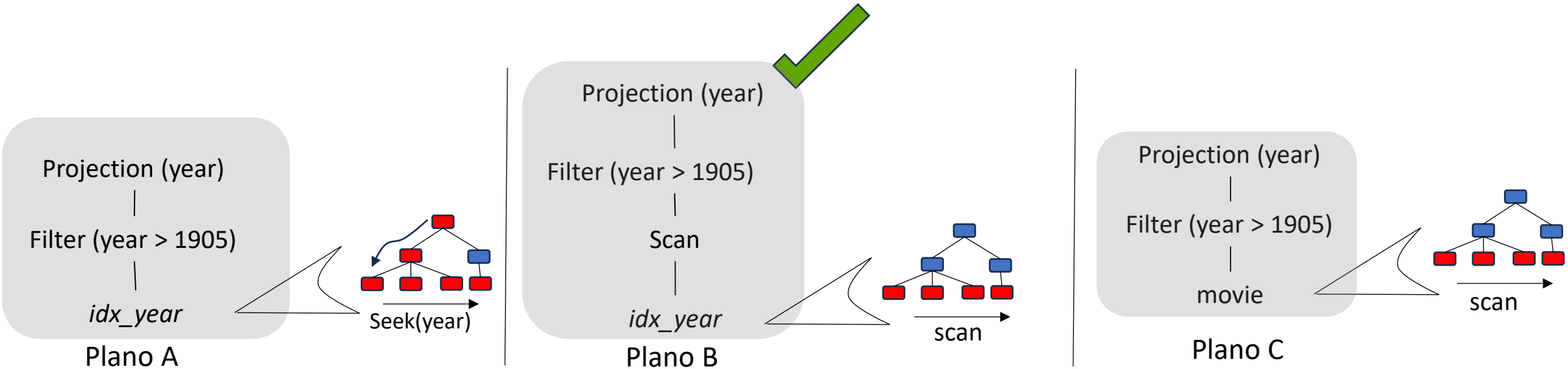
Filtro sobre o índice secundário – Covering Index

- Qual é melhor?



Filtro sobre o índice secundário – Covering Index

- Devido ao filtro pouco seletivo
  - O plano B, com scan, é melhor
  - Evita o overhead do plano A de navegação pela árvore B+ do seek
  - Evita o custo do plano C de varrer toda a tabela



Filtro sobre o índice secundário – Covering Index

- Em resumo
  - Na presença de um covering index, vale a pena usá-lo
  - Se o filtro for seletivo
    - Fazer um index seek
  - Se o filtro não for seletivo
    - Fazer um index scan

Sumário

- Operador Filter
- Uso do índice para resolver filtros
- Etapa de complementação
- Filtro sobre o índice primário
- Filtro sobre o índice secundário
  - Com covering index
  - Sem covering index

Filtro sobre o índice secundário – sem Covering Index

- Exemplo: título de filmes de 2002

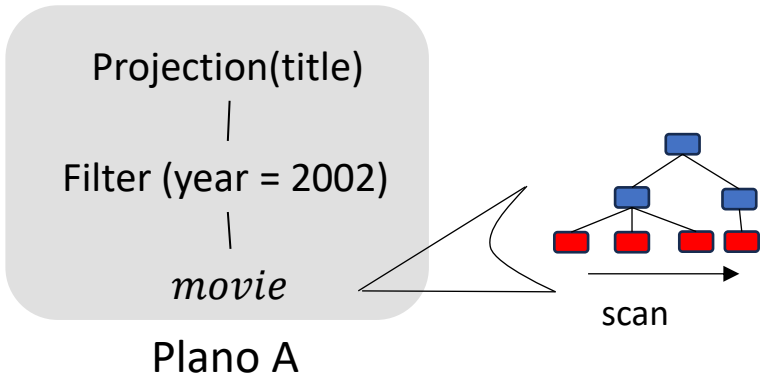
```
SELECT title
FROM movie
WHERE year = 2002
```

- O filtro é seletivo
- Existe um índice secundário sobre year
- A consulta solicita a coluna title, que não faz parte desse índice

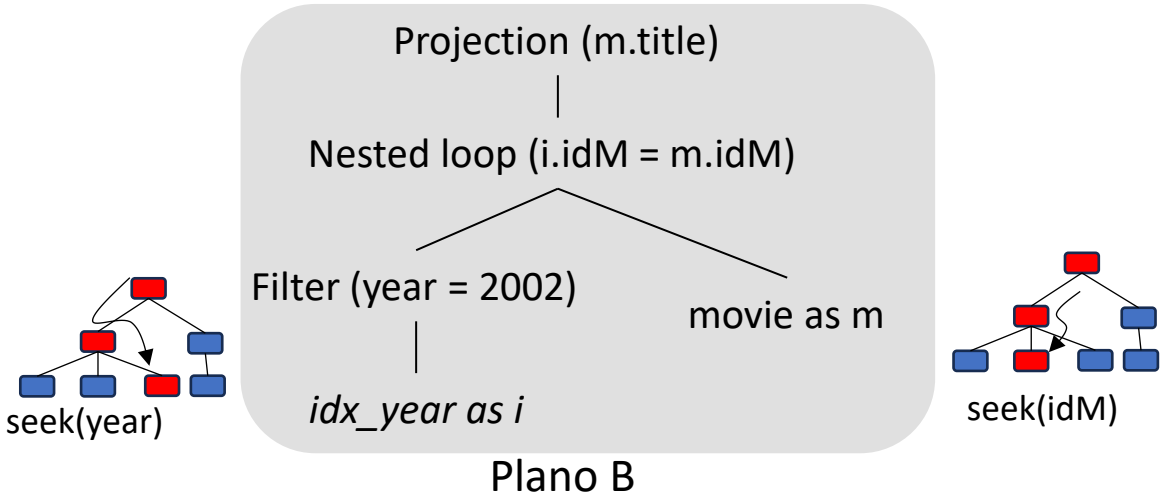
Filtro sobre o índice secundário – sem Covering Index

- Dois planos gerados
  - Plano A: Usa scan
  - Plano B: Usa seek, com complementação

- Qual é melhor?



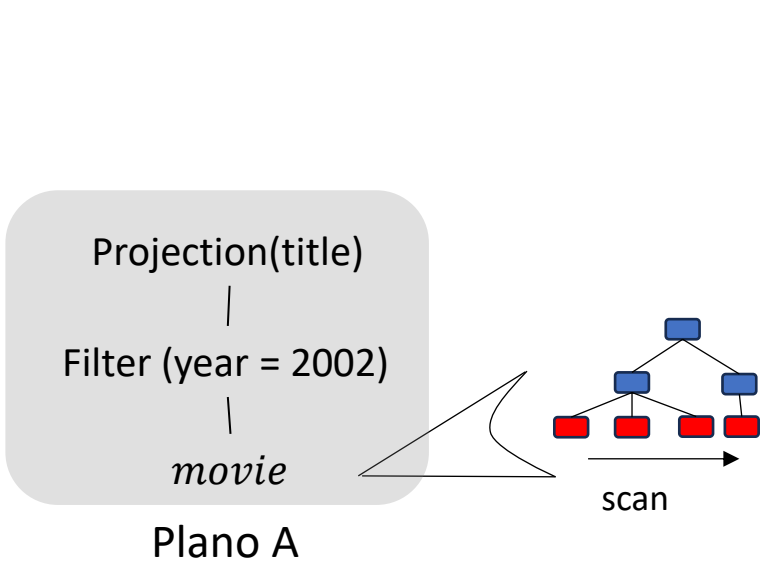
Plano A

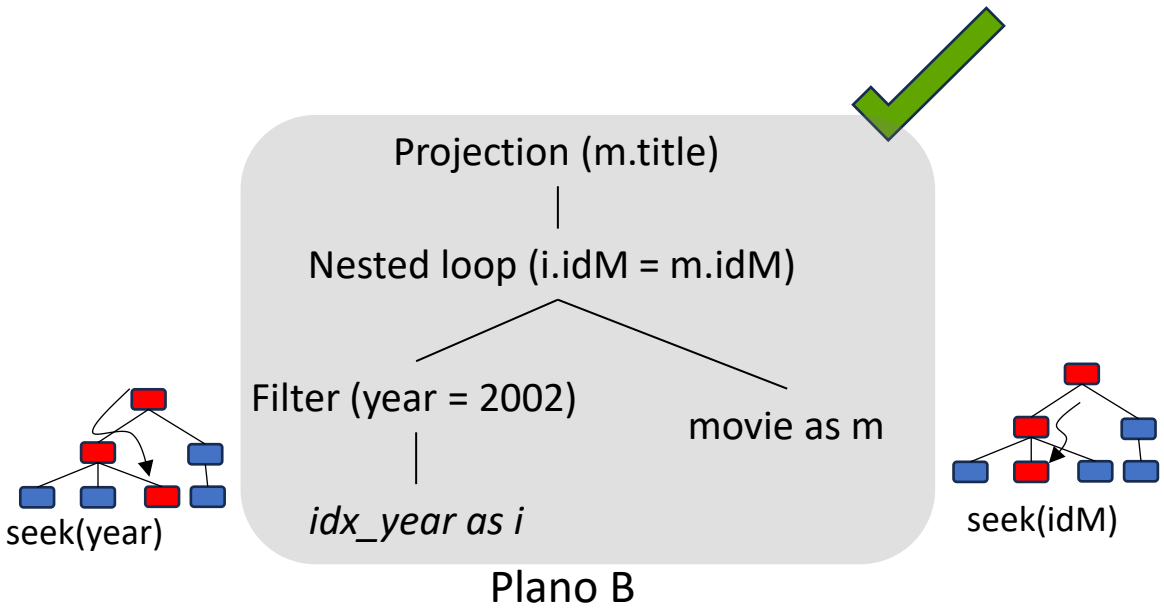


Plano B

Filtro sobre o índice secundário – sem Covering Index

- O plano com complementação é melhor
- Como o filtro é seletivo, a etapa de complementação será requisitada poucas vezes

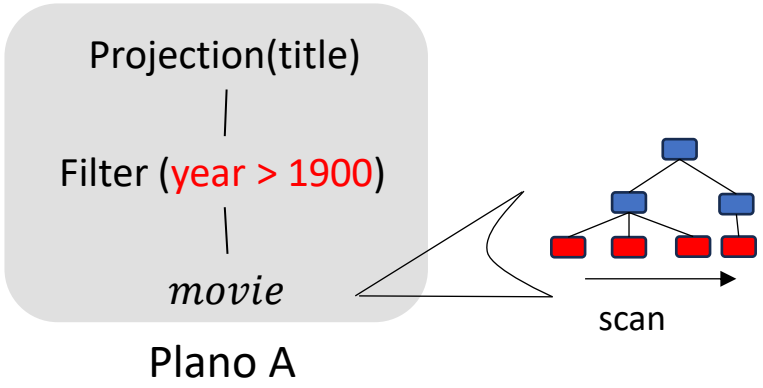




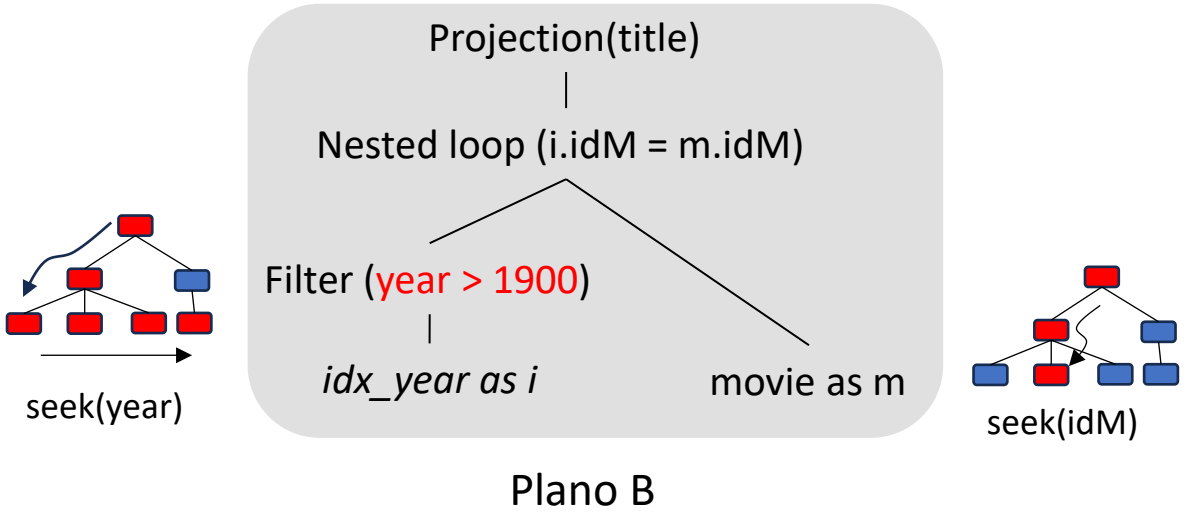
Filtro sobre o índice secundário – sem Covering Index

- E se o filtro for pouco seletivo (ex. year > 1900)
- Dois planos gerados
  - Plano A: Usa scan
  - Plano B: Usa seek, com complementação

- Qual é melhor?



Plano A

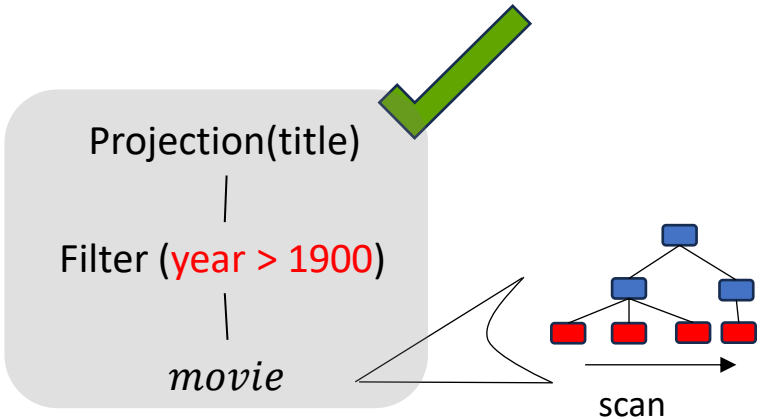


Plano B

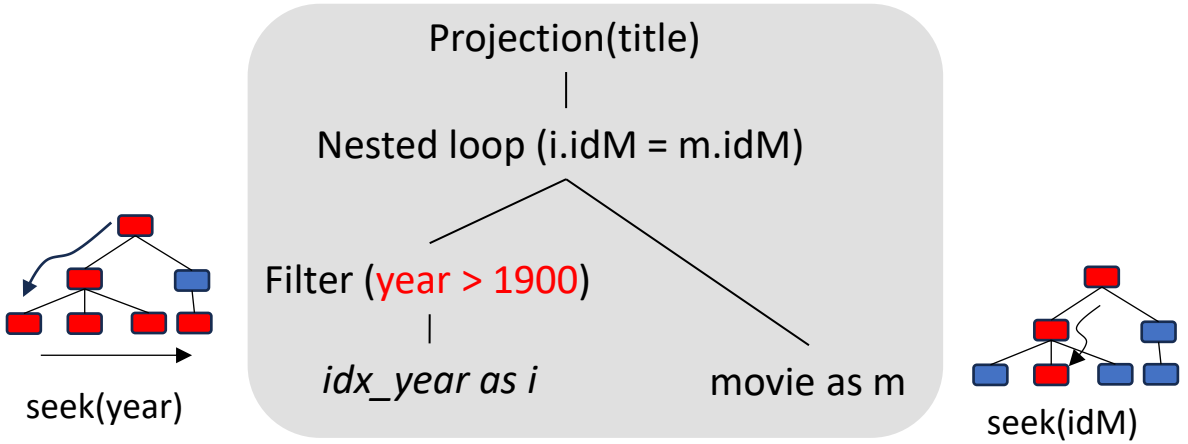
Filtro sobre o índice secundário – sem Covering Index

- Table Scan é melhor
  - O seek do plano B requer uma etapa de complementação
  - Para cada entrada de idx_year que satisfizer o filtro, uma busca em movie é necessária

- Importante:
  - Quanto menos seletivo for um filtro, mais cara é a complementação



Plano A



Plano B

Filtro sobre o índice secundário – sem Covering Index

- E se o filtro não for nem muito nem pouco seletivo? year > 1990
- Nesses casos o seek e o scan não são eficientes

- Com um seek
  - A quantidade de acessos aleatórios da complementação é muito alto
- Com um scan
  - O custo de varrer também é muito alto

Filtro sobre o índice secundário – sem Covering Index

- Para esses casos, os SGBDs costumam implementar técnicas de otimização no acesso

- Essas técnicas
  - recorrem ao índice
  - mas retiram a aleatoriedade no acesso às páginas

- Exemplos
  - MySQL: Multi Range Read (MRR)
  - PostreSQL: Bitmap Index

Filtro sobre o índice secundário – sem Covering Index

- Ex. O MRR ordena as chaves de busca antes de usá-las
  - Com a ordenação das chaves, o acesso passa a ser mais sequencial
    - Uma chave igual ou próxima à anterior possivelmente já esteja na memória (ex. 75, 101, 478)
      - Devido ao read ahead

Chaves de Busca
477
75
100
477
74
101
478

Acessos aleatórios

ordenação

Chaves de Busca
74
75
100
101
477
477
478

Acessos sequenciais

Encerrando

- Índices
  - aceleram a localização de registros
  - Mas implicam em uma sobrecarga
    - Na atualização
    - No espaço ocupado
- Não compensa indexar todos os atributos
  - Os índices seriam maiores do que as tabelas
  - A inserção de um registro seria muito mais demorada
- A escolha de quais atributos indexar deve ser criteriosa

Encerrando

- Critérios principais
  - Atributo usado em
    - filtros com alta seletividade
    - junções
- Critérios secundários
  - Atributo usado para
    - ordenação (ORDER BY)
    - agrupamento (GROUP BY)
    - ...
- Não indexar todas colunas usadas em consultas
  - Deve-se analisar o custo-benefício

Atividade Individual

- A consulta abaixo recupera nome do personagem e ordem de aparição para ordens superiores a algum valor

```
SELECT c_name, cast_order
FROM movie_cast
WHERE cast_order > ??
```

- No DBest
  - Crie um plano de execução que realize um table scan sobre movie_cast
  - Crie um plano de execução que realize um index seek sobre idx_cast_order

- Indique para que valor de cast_order uma estratégia é melhor do que a outra