**Universidade de Brasília**

**Faculdade de Ciências e Tecnologias em Engenharia**

**Sistemas de Banco de Dados 2**

**Estudando *Tuning* em Banco de Dados Relacional**

(Oportunidade Letiva de Discentes Experientes)

Na disciplina de Sistemas de Banco de Dados 2 (SBD2), oferecida no curso de Engenharia de Software da UnB, foram compartilhados estudos sobre o tema de *Tuning* em Bancos de Dados Relacionais, focando em consultas SQL, por um dos monitores da disciplina, que lecionou parte de uma aula regular para a turma atual de SBD2.

Nesta atividade foram explicados conceitos relevantes ao tema e demonstradas operações simples de serem realizadas pelo Workbench MySQL. Assim, está sendo apresentado abaixo como exemplo e modelo a consulta trabalhada em sala de aula envolvendo uma base de dados utilizada pela turma em exercícios solicitado pelo docente em atividade anterior (base disponível na Área de Compartilhamento - **/aulas/basesDados/projetoBaseDados\_Jogos\_2020.zip**).

Em seguida, são apresentadas três demandas solicitadas por seu cliente no qual você deverá elaborar uma consulta SQL que resolva cada problema demandado e depois apresentar o resultado da explicação fornecida pelo MySQL (*explain*) sobre sua consulta proposta inicialmente.

Após a explicação de sua solução (*Result Grid*) você deverá solicitar a análise do MySQL (*analyze*) para otimizar a sua proposta inicial, sendo ao final apresentado o resultado e a consulta que atenda a demanda otimizada.

Dessa forma, a apresentação realizada em sala de aula está detalhada a seguir com a indicação de primeira demanda (1) do cliente e você deverá elaborar as outras TRÊS propostas de soluções seguindo EXATAMENTE o mesmo padrão realizado para atender a primeira demanda.

**1 ) Primeira Demanda** Elabore uma consulta SQL que retorne informações detalhadas sobre jogos, incluindo: ID do jogo, nome, plataforma de lançamento, ano de lançamento e o total de vendas globais (calculado pela soma das vendas na América do Norte, Europa, Japão e outras regiões). A consulta deve permitir a busca por um jogo específico, seja pelo seu identificador (id\_game = 1000), seja por nomes que começam com determinada palavra, utilizando como exemplo o prefixo “max”.

Proposta de solução inicial:  
  
  **SELECT**

**id\_game as ID\_do\_Jogo,**

**name as Nome\_do\_Jogo,**

**platform as Plataforma,**

**year\_of\_release as Eh\_de,**

**(na\_sales + eu\_Sales + jp\_sales + other\_sales) as Total\_de\_vendas**

**FROM**

**GAME**

**WHERE**

**id\_game = 1000 OR name like 'max%';**

Na consulta acima foi inicialmente definido os atributos que seriam projetados (apresentados) para o usuário que gerou a demanda como o resultado solicitado para a equipe de **TI** (ou de BD em específico). Em seguida, foi identificada a tabela principal para esta consulta (**GAME**). Na cláusula WHERE desta consulta foi elaborada a condição para a filtragem para o ID do jogo ou nome do jogo

Solicitando explicação sobre a solução inicial (*explain*):

**EXPLAIN SELECT**

**id\_game as ID\_do\_Jogo,**

**name as Nome\_do\_Jogo,**

**platform as Plataforma,**

**year\_of\_release as Eh\_de,**

**(na\_sales + eu\_Sales + jp\_sales + other\_sales) as Total\_de\_vendas**

**FROM**

**GAME**

**WHERE**

**id\_game = 1000 OR name like 'max%';**

Analisando os resultados apresentados pelo MySQL Workbench no *Result Grid* são destacados alguns itens relacionados à consulta proposta inicialmente.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **id** | **select\_type** | **table** | **partitions** | **type** | **possible\_keys** | **key** | **key\_len** | **ref** | **rows** | **filtered** | **Extra** |
| 1 | SIMPLE | GAME | NULL | ALL | PRIMARY | NULL | NULL | NULL | 11957 | 11.12 | Using where |

## Resultado mostrado no *Visual Explain* do MySQL**.**

## 

Interpretando o *Result Grid* e *Visual Explain*:

## O comando EXPLAIN mostra que a consulta está fazendo uma varredura completa na tabela GAME (indicada pelo tipo de acesso ALL), ou seja, não está utilizando um índice. Embora a chave primária (PRIMARY) esteja listada em possible\_keys, ela não foi usada (key está vazia), porque a cláusula WHERE utiliza um OR, o que dificulta o uso de índices.

Solicitando a análise da solução inicial (*analyze*):

**EXPLAIN ANALYZE SELECT**

**id\_game as ID\_do\_Jogo,**

**name as Nome\_do\_Jogo,**

**platform as Plataforma,**

**year\_of\_release as Eh\_de,**

**(na\_sales+eu\_Sales + jp\_sales + other\_sales) as Total\_de\_vendas**

**FROM**

**GAME**

**WHERE**

**id\_game = 1000 OR name like 'max%';**

Apresentando o resultado da nova execução do *ANALYZE:*

-> Filter: ((GAME.id\_game = 1000) or (GAME.`name` like 'max%'))

(cost=1220 rows=1329) (actual time=5.46..47.8 rows=14 loops=1)

-> Table scan on GAME (cost=1220 rows=11957)

(actual time=0.421..26.8 rows=12043 loops=1)

A leitura do resultado do EXPLAIN ANALYZE deve ser feita de forma hierárquica, do nó mais interno para o mais externo. Isso significa que as operações listadas mais abaixo na estrutura do plano são executadas primeiro, e seus resultados servem de entrada para as operações que aparecem acima.

Observa-se que a consulta realiza um table scan completo na tabela GAME, ou seja, percorre todas as linhas da tabela para aplicar o filtro. Isso é evidenciado pela linha "Table scan on GAME", que indica a leitura de aproximadamente 12.043 linhas em tempo real. O filtro da cláusula WHERE (com id\_game = 1000 OR name LIKE 'max%') foi aplicado após a leitura dos dados, resultando em 14 linhas retornadas. O uso de OR e o operador LIKE tem um o custo da operação (cost=1220) e o tempo total da execução (de cerca de 47.8 milissegundos). Com base nisso, e na ausência de detalhes mais específicos sobre o custo de cada cláusula OR, opta-se por facilitar a análise adotando uma abordagem alternativa na escrita da consulta, utilizando o operador UNION ALL.

Nova proposta de solução:

**SELECT**

**id\_game as ID\_do\_Jogo,**

**name as Nome\_do\_Jogo,**

**platform as Plataforma,**

**year\_of\_release as Eh\_de,**

**(na\_sales + eu\_Sales + jp\_sales + other\_sales) as Total\_de\_vendas**

**FROM**

**GAME**

**WHERE**

**id\_game = 1000**

**UNION ALL**

**SELECT**

**id\_game as ID\_do\_Jogo,**

**name as Nome\_do\_Jogo,**

**platform as Plataforma,**

**year\_of\_release as Eh\_de,**

**(na\_sales + eu\_Sales + jp\_sales + other\_sales) as Total\_de\_vendas**

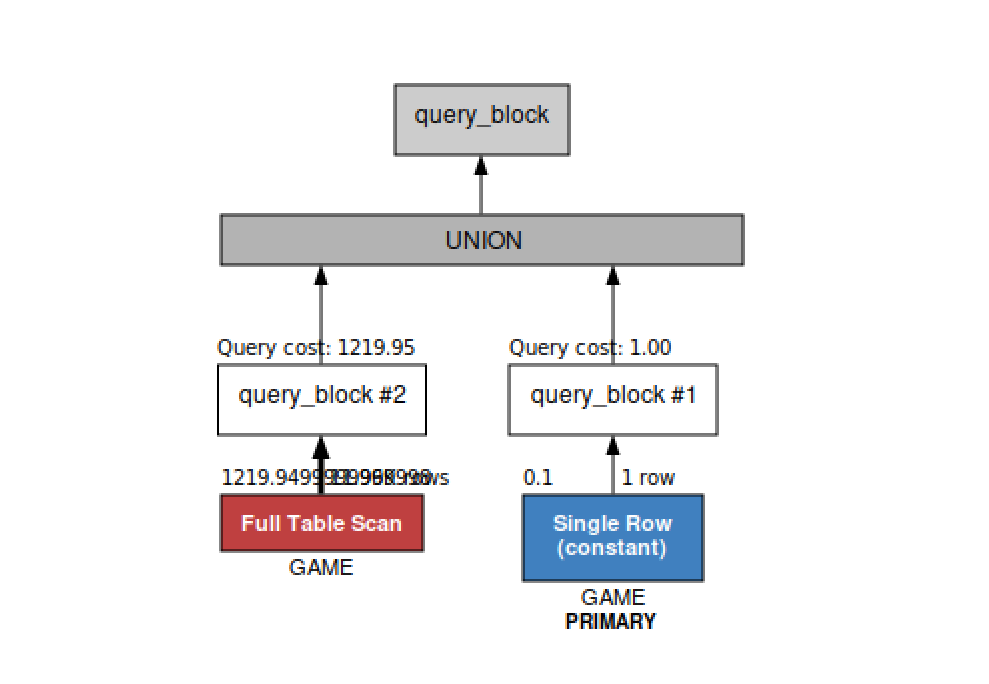
**FROM**

**GAME**

**WHERE**

**name like 'max%';**

Apresentando o resultado da nova execução do *ANALYZE e pelo Visual Explain*:



**->** Append (cost=1220 rows=1329) (actual time=0.031..43.2 rows=14 loops=1)

-> Stream results (cost=0..0 rows=1) (actual time=0.0226..0.0286

rows=1 loops=1)

-> Rows fetched before execution (cost=0..0 rows=1)

(actual time=0.00384..0.00566 rows=1 loops=1)

-> Stream results (cost=1220 rows=1328) (actual time=22.7..43.1

rows=13 loops=1)

-> Filter: (GAME.`name` like 'max%') (cost=1220 rows=1328)

(actual time=22.6..43.1 rows=13 loops=1)

-> Table scan on GAME (cost=1220 rows=11957)

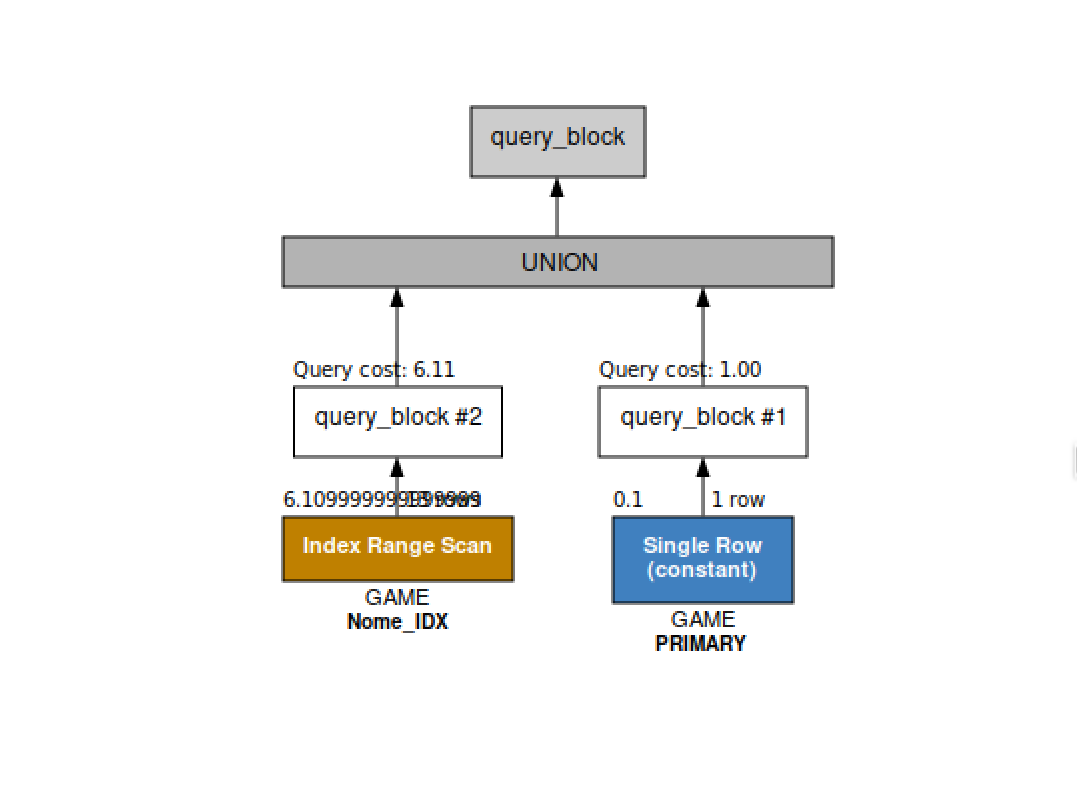
(actual time=0.398..23.8 rows=12043 loops=1)

Ao utilizar UNION ALL, a consulta foi dividida em duas partes distintas: uma que busca o jogo com id\_game = 1000 e outra que busca jogos com nomes que começam com "max". Isso permite que o MySQL processe cada filtro separadamente, o que é mais eficiente do que usar um único OR, pois o filtro exato do id\_game pode ser resolvido rapidamente, enquanto o filtro com LIKE ainda exige uma varredura completa da tabela. No entanto, o custo total e o tempo (actual) de execução permanecem muito próximos (com OR com 47.8 o UNION ALL 43.2) , com a parte do LIKE ainda sendo o principal gargalo. Então uma forma de melhorar agora esse gargalo seria criar um índice para o coluna name.

## **CREATE INDEX Nome\_IDX on GAME(name);**

Em seguida, executando o comando EXPLAIN ANALYZE agora com o índice criado.

Apresentando o resultado da nova execução do *ANALYZE e pelo Visual Explain*:



-> Append (cost=6.11 rows=14) (actual time=0.0295..0.284 rows=14 loops=1)

-> Stream results (cost=0..0 rows=1) (actual time=0.0214..0.027

rows=1 loops=1)

-> Rows fetched before execution

(cost=0..0 rows=1) (actual time=0.00328..0.0051 rows=1 loops=1)

-> Stream results (cost=6.11 rows=13)

(actual time=0.0963..0.193 rows=13 loops=1)

-> Index range scan on GAME using Nome\_IDX over ('max\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0' <= name <= 'max?????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????'), with index condition: (GAME.`name` like 'max%') (cost=6.11 rows=13) (actual time=0.0858..0.12 rows=13 loops=1)

No plano de execução apresentado, o operador Append indica que o MySQL está unindo os resultados de duas subconsultas devido ao uso do UNION ALL. O campo cost mostra a estimativa do otimizador, que calculou um custo total de 6.11 para a execução, prevendo o retorno de 14 linhas. Na primeira parte da consulta, é realizada uma busca direta (no índice primário) para localizar o registro com id\_game = 1000, conforme evidenciado pela linha Rows fetched before execution. Já na segunda parte, o MySQL realiza um Index range scan utilizando o índice Nome\_IDX, definido sobre a coluna name, para encontrar registros cujo nome começa com 'max'. O motor de busca delimita o intervalo de nomes possíveis com base nesse prefixo, otimizando o acesso ao índice. A presença da linha with index condition: (GAME.name like 'max%') é um bom sinal, pois mostra que a condição do LIKE está sendo aplicada diretamente sobre o índice, o que melhora significativamente a performance da consulta.

Dessa forma, a consulta inicialmente proposta foi modificada sua escrita, e um novo objeto no SGBD foi criado para tornar o desempenho dessa consulta SQL mais eficiente.

## **2 ) Segunda demanda**

Elaborar uma consulta que recupera informações sobre os jogos que possuem vendas na América do Norte superiores a 1 milhão de unidades. A consulta deve retornar o nome do jogo, o gênero, e as vendas nas regiões Norte-Americana, Européia e Japonesa, ordenadas pela venda na América do Norte em ordem decrescente.

Proposta de solução inicial (elabore a consulta que resolva esta nova demanda):

**SELECT**

**g.name AS game\_name,**

**ge.description AS genre\_description,**

**g.na\_sales,**

**g.eu\_sales,**

**g.jp\_sales**

**FROM**

**GAME g**

**JOIN**

**GENRE ge ON g.id\_genre = ge.id\_genre**

**WHERE**

**g.na\_sales > 1.0**

**ORDER BY**

**g.na\_sales DESC;**

Solicitando explicação sobre a solução inicial (*explain*):  
  
 **EXPLAIN SELECT**

**g.name AS game\_name,**

**ge.description AS genre\_description,**

**g.na\_sales,**

**g.eu\_sales,**

**g.jp\_sales**

**FROM**

**GAME g**

**JOIN**

**GENRE ge ON g.id\_genre = ge.id\_genre**

**WHERE**

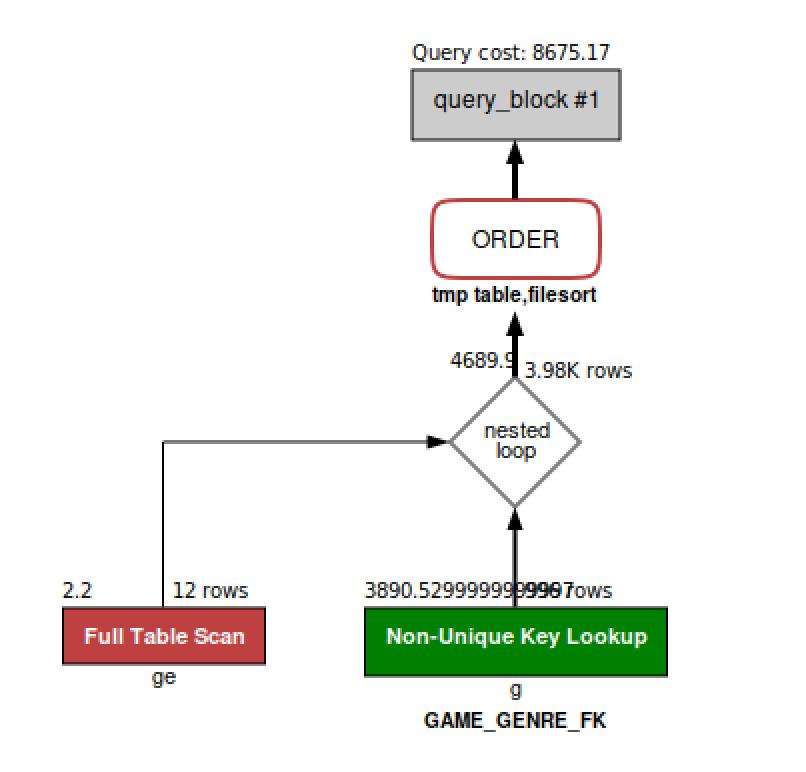
**g.na\_sales > 1.0**

**ORDER BY**

**g.na\_sales DESC;**

Apresentando a tabela e interpretando o *Result Grid* do *EXPLAIN* e *Visual Explain*:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **id** | **select\_type** | **table** | **partitions** | **type** | **possible\_keys** | **key** | **key\_len** | **ref** | **rows** | **filtered** | **Extra** |
| 1 | SIMPLE | ge | NULL | ALL | PRIMARY | NULL | NULL | NULL | 12 | 100.00 | Using temporary; Using filesort |
| 1 | SIMPLE | g | NULL | ref | GAME\_GENRE\_FK | GAME\_GENRE\_FK | 5 | t2\_jogos.ge.id\_genre | 996 | 33.33 | Using where |



Solicitando a análise da solução inicial (*analyze*):

**EXPLAIN ANALYZE SELECT**

Apresentando e interpretando o resultado do *ANALYZE*:

-**> Sort: g.na\_sales DESC**  (actual time=60.5..61.5 rows=757 loops=1)

**-> Stream results**  (cost=4690 rows=3985) (actual time=2.42..59.2

rows=757 loops=1)

**-> Nested loop inner join** (cost=4690 rows=3985) (actual

time=2.4..57 rows=757 loops=1)

**-> Table scan on ge** (cost=2.2 rows=12) (actual

time=1.09..1.11 rows=12 loops=1)

**-> Filter:** (g.na\_sales > 1.00) (cost=294 rows=332)

(actual time=0.385..4.49 rows=63.1 loops=12)

**-> Index lookup on g using**

GAME\_GENRE\_FK (id\_genre=ge.id\_genre) (cost=294

r ows=996) (actual time=0.285..3.01 rows=1004 loops=12)

Explicar os ajustes realizados para melhoria da consulta inicial da tarefa 2:

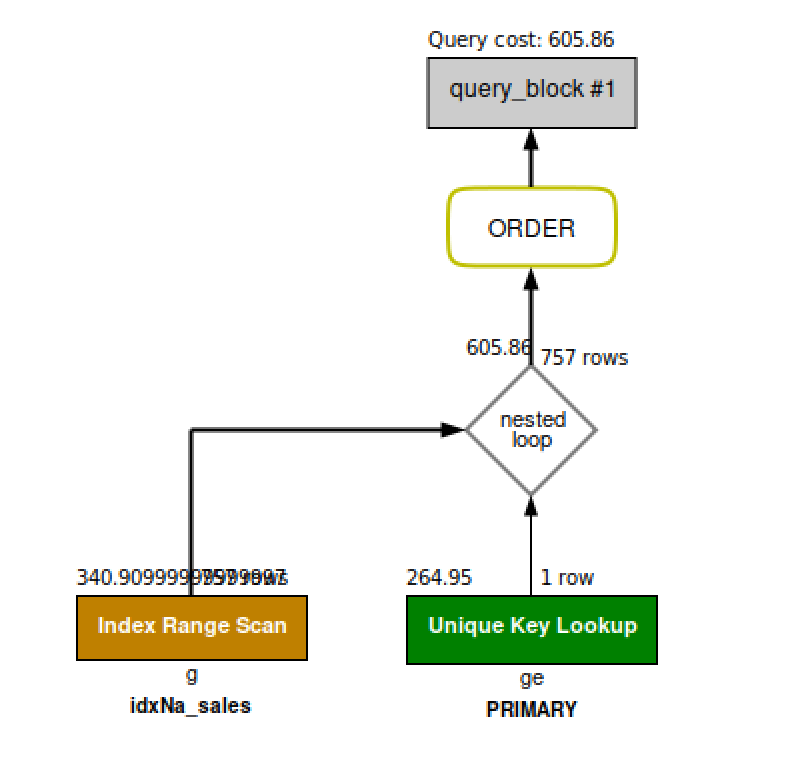
==> É importante esclarecer que além da explicação cada comando SQL usado ou alteração da consulta SQL proposta inicialmente também deverá fazer parte deste item da evolução da consulta SQL em questão que você propôs para esta demanda.

**CREATE INDEX idxNa\_sales ON GAME (na\_sales);**

Apresentando o *Result Grid* da nova execução do *EXPLAIN*:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **id** | **select\_type** | **table** | **partitions** | **type** | **possible\_keys** | **key** | **key\_len** | **ref** | **rows** | **filtered** | **Extra** |
| 1 | SIMPLE | ge | NULL | eq\_ref | PRIMARY | PRIMARY | 4 | 'base\_jogos.g.id\_genre' | 1 | 100.00 | NULL |
| 1 | SIMPLE | g | NULL | range | GAME\_GENRE\_FK, idxNa\_sales | idxNa\_sales | 3 | NULL | 757 | 100.00 | Using index condition; Using where; Backward index scan |

## Resultado mostrado no **Visual Explain.**



Apresentando o resultado da nova execução do *ANALYZE*:

-> Nested loop inner join (cost=606 rows=757) (actual time=0.399..14.8

rows=757 loops=1)

-> Filter: (g.id\_genre is not null) (cost=341 rows=757)

(actual time=0.37..5.18 rows=757 loops=1)

-> Index range scan on g using idxNa\_sales over (1.00 <

na\_sales) (reverse), with index condition: (g.na\_sales > 1.00) (

cost=341 rows=757) (actual time=0.361..3.05 rows=757 loops=1)

-> Single-row index lookup on ge using PRIMARY (id\_genre=

g.id\_genre) (cost=0.25 rows=1) (actual time=0.00411..0.00551

rows=1 loops=757)

Apresentação da análise final a partir da consulta inicial proposta e quais são os principais indicativos de que a consulta ficou mais eficiente:

## **3 ) Terceira Demanda.**

Liste o nome dos jogos e o nome de seus respectivos publicadores, para os jogos que possuem a pontuação de usuário (user\_score) igual a 9.

Proposta de solução inicial (elabore a consulta que resolva esta nova demanda):

**SELECT**

**g.name as Nome\_do\_jogo,**

**p.publisher\_name as NomeDaPublicadora**

**FROM**

**GAME g**

**JOIN**

**PUBLISHER p on g.id\_publisher = p.id\_publisher**

**WHERE**

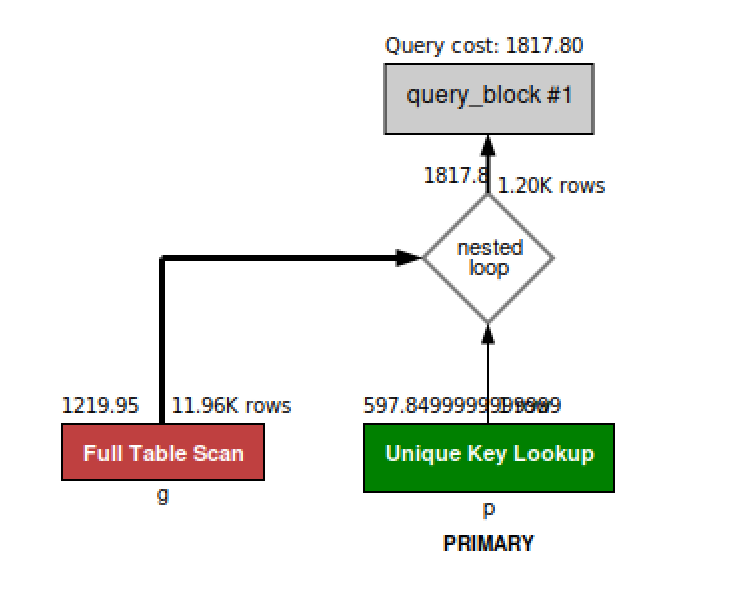
**g.user\_score = '9';**

Solicitando explicação sobre a solução inicial (*explain*):

**EXPLAIN SELECT**

Apresentando a tabela e interpretando o *Result Grid* do *EXPLAIN* e *Visual Explain*:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **id** | **select\_type** | **table** | **partitions** | **type** | **possible\_keys** | **key** | **key\_len** | **ref** | **rows** | **filtered** | **Extra** |
| 1 | SIMPLE | g | NULL | ALL | GAME\_PUBLISHER\_FK | null | NULL | NULL | 11957 | 10.00 | Using where; |
| 1 | SIMPLE | p | NULL | eq\_ref | PRIMARY | PRIMARY | 4 | 'base\_jogos.g.id\_publisher' | 1 | 100.00 | NULL |



Solicitando a análise da solução inicial (*analyze*):

**EXPLAIN ANALYZE SELECT**

Apresentando e interpretando o resultado do *ANALYZE*:

-> Nested loop inner join (cost=1818 rows=1196) (actual time=0.552..41.4 rows=85 loops=1)

-> Filter: (g.user\_score = '9') (cost=1220 rows=1196)

(actual time=0.509..39.8 rows=85 loops=1)

-> Table scan on g (cost=1220 rows=11957) (actual

time=0.293..22.1 rows=12043 loops=1)

-> Single-row index lookup on p using

PRIMARY (id\_publisher=g.id\_publisher) (cost=0.4 rows=1)

(actual time=0.00926..0.0107 rows=1 loops=85)

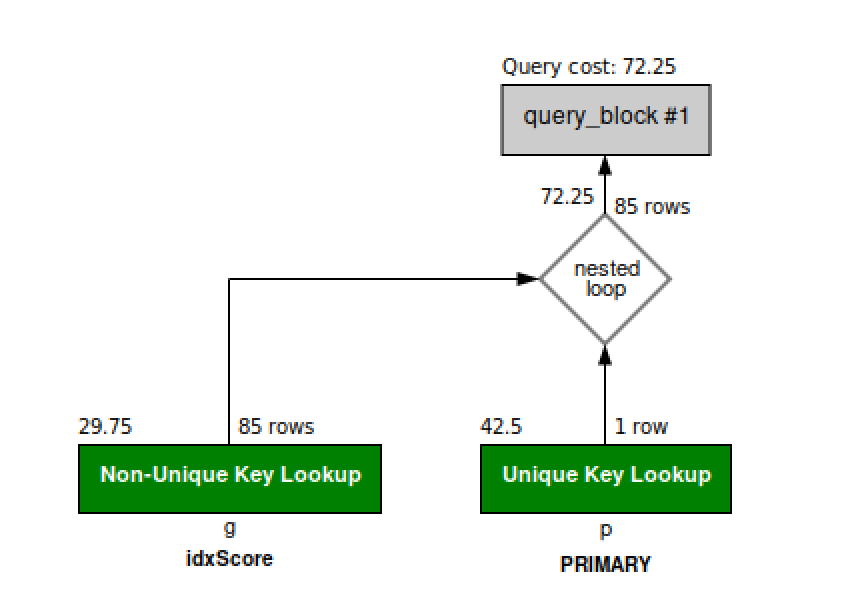
Explicar os ajustes realizados para melhoria da consulta inicial da tarefa 3:

==> É importante esclarecer que além da explicação cada comando SQL usado ou alteração da consulta SQL proposta inicialmente também deverá fazer parte deste item da evolução da consulta SQL em questão que você propôs para esta demanda.

**CREATE INDEX idxScore ON GAME(user\_score);**

Apresentando o *Result Grid* da nova execução do *EXPLAIN e Visual Explain:*

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **id** | **select\_type** | **table** | **partitions** | **type** | **possible\_keys** | **key** | **key\_len** | **ref** | **rows** | **filtered** | **Extra** |
| 1 | SIMPLE | g | NULL | ref | GAME\_PUBLISHER\_FK, idxScore | idxScore | 6 | const | 85 | 10.00 | NULL |
| 1 | SIMPLE | p | NULL | eq\_ref | PRIMARY | PRIMARY | 4 | 'base\_jogos.g.id\_publisher' | 1 | 100.00 | NULL |



-> Nested loop inner join (cost=72.2 rows=85) (actual time=0.192..3.1 rows=85 loops=1)

-> Index lookup on g using idxScore (user\_score='9') (cost=29.8 rows=85) (actual time=0.162..1.05 rows=85 loops=1)

-> Single-row index lookup on p using PRIMARY (id\_publisher=g.id\_publisher) (cost=0.401 rows=1) (actual time=0.0108..0.0129 rows=1 loops=85)

Apresentação da análise final a partir da consulta inicial proposta e quais são os principais indicativos de que a consulta ficou mais eficiente:

## **4 ) Quarta Demanda**

Listar o máximo do valor de venda de um jogo no Japão por categoria em que a nota dos críticos (*critic\_score*) seja maior que 9 e tenha mais de 50 avaliações de críticos (*critic\_count*).

Proposta de solução inicial (elabore a consulta que resolva esta nova demanda):

**SELECT**

**ge.description AS genre\_description,**

**MAX(g.jp\_sales) AS max\_japan\_sales**

**FROM**

**GAME g**

**JOIN**

**GENRE ge ON g.id\_genre = ge.id\_genre**

**WHERE**

**g.critic\_score > 9**

**AND g.critic\_count > 50**

**GROUP BY**

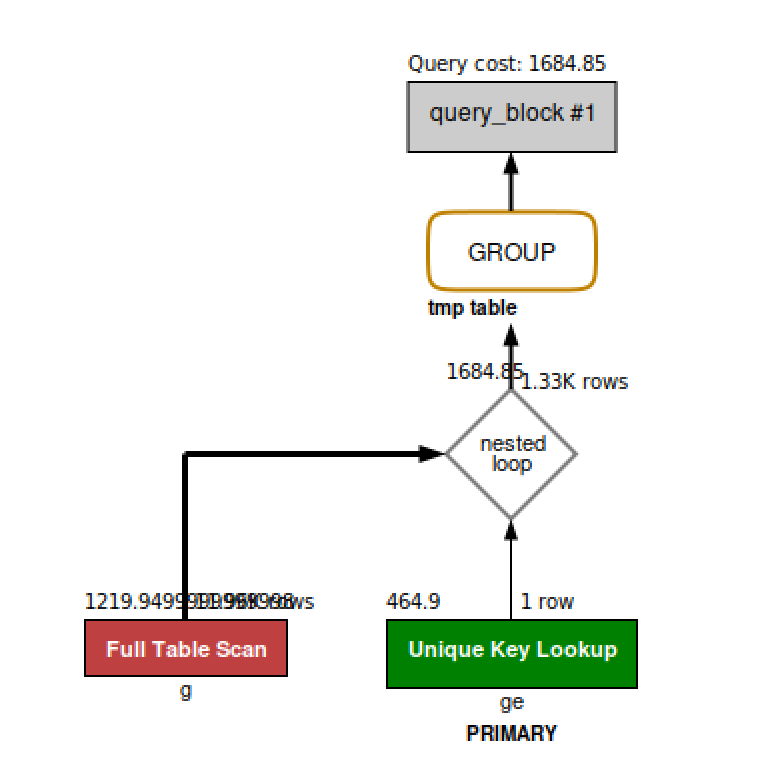
**ge.description;**

Solicitando explicação sobre a solução inicial (*explain*):

**EXPLAIN SELECT**

Apresentando a tabela e interpretando o *Result Grid* do *EXPLAIN e Visual Explain*:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **id** | **select\_type** | **table** | **partitions** | **type** | **possible\_keys** | **key** | **key\_len** | **ref** | **rows** | **filtered** | **Extra** |
| 1 | SIMPLE | g | NULL | ALL | GAME\_GENRE\_FK | NULL | NULL | NULL | 11957 | 11.11 | Using where; Using temporary |
| 1 | SIMPLE | ge | NULL | eq\_ref | PRIMARY | PRIMARY | 4 | base\_jogos.g.id\_genre | 1 | 100.00 | NULL |



Solicitando a análise da solução inicial (*analyze*):

**EXPLAIN ANALYZE SELECT**

Apresentando e interpretando o resultado do *ANALYZE*:

-> Table scan on <temporary> (actual time=52.7..52.7 rows=12 loops=1)

-> Aggregate using temporary table (actual time=52.7..52.7 rows=12 loops=1)

-> Nested loop inner join (cost=1685 rows=1328) (actual time=0.398..50.7

rows=830 loops=1)

-> Filter: ((g.critic\_score > 9) and (g.critic\_count > 50) and

(g.id\_genre is not null)) (cost=1220 rows=1328) (actual time=0.368..40.6

rows=830 loops=1)

-> Table scan on g (cost=1220 rows=11957) (actual

time=0.3..21.7 rows=12043 loops=1)

-> Single-row index lookup on ge using PRIMARY (id\_genre=g.id\_genre) (cost=0.25 rows=1) (actual time=0.00347..0.00486 rows=1 loops=830)

Explicar os ajustes realizados para melhoria da consulta inicial da tarefa 4:

==> É importante esclarecer que além da explicação cada comando SQL usado ou alteração da consulta SQL proposta inicialmente também deverá fazer parte deste item da evolução da consulta SQL em questão que você propôs para esta demanda.

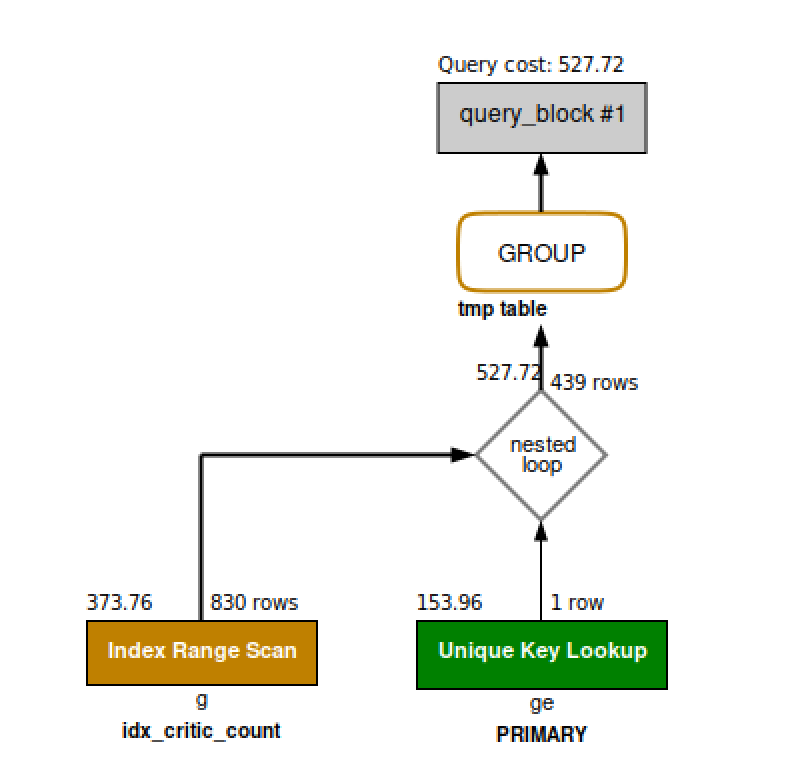
CREATE INDEX idx\_critic\_score ON GAME (critic\_score);

CREATE INDEX idx\_critic\_count ON GAME (critic\_count);

CREATE INDEX idx\_jp\_sales ON GAME (jp\_sales);

Apresentando o *Result Grid* da nova execução do *EXPLAIN* e *Visual Explai*

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **id** | **select\_type** | **table** | **partitions** | **type** | **possible\_keys** | **key** | **key\_len** | **ref** | **rows** | **filtered** | **Extra** |
| 1 | SIMPLE | g | NULL | range | GAME\_GENRE\_FK,idx\_critic\_score,idx\_critic\_count | idx\_critic\_count | 5 | NULL | 830 | 53.00 | Using index condition; Using where; Using temporary |
| 1 | SIMPLE | ge | NULL | eq\_ref | PRIMARY | PRIMARY | 4 | base\_jogos.g.id\_genre | 1 | 100.00 | NULL |



Apresentando o resultado da nova execução do *ANALYZE*:

-> Table scan on <temporary> (actual time=23.2..23.3 rows=12 loops=1)

-> Aggregate using temporary table (actual time=23.2..23.2 rows=12 loops=1)

-> Nested loop inner join (cost=528 rows=440) (actual time=0.311..20.4

rows=830 loops=1)

-> Filter: ((g.critic\_score > 9) and (g.id\_genre is not null)) (cost=374 rows=440)

(actual time=0.286..7.17 rows=830 loops=1)

-> Index range scan on g using idx\_critic\_count over (50 < critic\_count),

with index condition: (g.critic\_count > 50) (cost=374 rows=830) (actual time=0.276..

4.05 rows=830 loops=1)

-> Single-row index lookup on ge using PRIMARY (id\_genre=g.id\_genre) (cost=0.25 rows=1) (actual time=0.00526..0.00706 rows=1 loops=830)

Apresentação da análise final a partir da consulta inicial proposta e quais são os principais indicativos de que a consulta ficou mais eficiente: