Tempo de execução do processo Extrair-Carregar-Transformar (ELT) Análise e Otimização

Aleksei E. Zvonarev1 , Dmitriy S. Gudilin2 , Dmitriy A. Lychagin3 , Boris S. Goryachkin4
Bauman Moscow State Technical University
Moscou, *Federação Russa*
defygee@yandex.ru1 , dima.gudilin@mail.ru2 , lychagin.dmitry@gmail.com3 , bsgor@mail.ru4

***Resumo—O* artigo discute algoritmos para otimização da etapa de transformação do processo ELT, construídos com base em procedures do SGBD PostgreSQL e no mecanismo de paralelização implementado pelas ferramentas da linguagem de programação Python. Como base de comparação, foi tomada a versão mais trivial do processo de conversão de dados, que consiste em uma conexão sequencial de cada procedimento individual. O primeiro algoritmo proposto utiliza o princípio da paralelização mais simples de procedimentos, o que permite realizar procedimentos independentes em paralelo. O segundo algoritmo é uma versão melhorada do primeiro. Ele usa o princípio de otimização passo a passo com paralelização adicional de blocos de cadeia de procedimentos dependentes. Como principal critério de avaliação, tomou-se o tempo de execução de toda a cadeia de procedimentos. Como resultado do estudo, foi determinado que a versão melhorada do algoritmo de paralelização do procedimento apresenta o menor tempo de execução de toda a cadeia da etapa de transformação de dados.**

Palavras-chave— otimização, ELT, Python, PostgreSQL, fluxo, procedimento, DBMS

I. INTRODUÇÃO

Dentro das organizações, os dados geralmente estão espalhados por diferentes sistemas [1] e se tornam inúteis por não serem armazenados centralmente no sistema e formato corretos [2]. É difícil descobrir [3] todos os insights úteis desses dados.
Extração, Carregamento e Transformação (ELT) é um método usado atualmente para transferir dados do sistema de origem para o armazenamento de dados. Envolve recuperar dados de muitas fontes externas, carregá-los em um banco de dados de destino (DB) ou armazenamento de dados [4] e, em seguida, convertê-los para atender às necessidades operacionais (às vezes usando tabelas intermediárias) [5]. Durante o processo de extração, os dados são extraídos de várias fontes de dados para posterior processamento ou armazenamento. A Transformação de Dados é responsável por converter um conjunto de valores de dados do formato do sistema de origem para o formato do sistema de destino de acordo com a lógica de negócios fornecida pelo cliente na forma de um documento de requisitos de negócios [6].

Este artigo discutirá a abordagem analítica da construção da arquitetura da etapa de transformação do processo ELT usando o SGBD PostgreSQL e a linguagem de programação Python e construiu um algoritmo para resolver o problema de sua otimização sequencial.

II. REVISÃO DA LITERATURA

Não há tantos métodos propostos na literatura para otimizar processos ELT quanto para processos ETL, mas muitas abordagens e métodos para a fase de transformação de dados usando paralelização podem ser reutilizados em processos ELT. Em [7] o autor considera várias opções diferentes para otimizar processos ETL, tais como: Abordagem de espaço de estado, Grafo de dependência, Estratégias de escalonamento, Padrões reutilizáveis, Paralelismo, Métricas de qualidade, Estatísticas, uso de ferramentas ETL comerciais.

Em [8] também é apresentada uma metodologia para paralelismo de processos no fluxo de trabalho ELT, mas este trabalho também carece de algoritmos de otimização específicos.

Além disso, existem várias estruturas semânticas ETL [9] [10] na literatura que se concentram em fornecer semântica para vários campos de dados, facilitando assim uma integração de dados mais rica. Essa abordagem pode ser útil, mas não pode produzir arquitetura ELT.

Nosso trabalho revela com mais detalhes as variantes de algoritmos para paralelização passo a passo de procedimentos relacionados à parte Paralelismo em [7] e a metodologia em [8], utilizando como exemplo o SGBD PostgreSQL e ferramentas escritas na linguagem de programação Python .

III. DESCRIÇÃO DO PROCESSO ELT

O esquema de processo ELT padrão inclui vários sistemas de origem dos quais os dados devem ser transferidos para um único data warehouse [9] continuamente. Depois que os dados são coletados em um único repositório, eles podem ser transformados para resolver vários tipos de negócios ou problemas técnicos. O diagrama do processo ELT é mostrado na Figura 1.

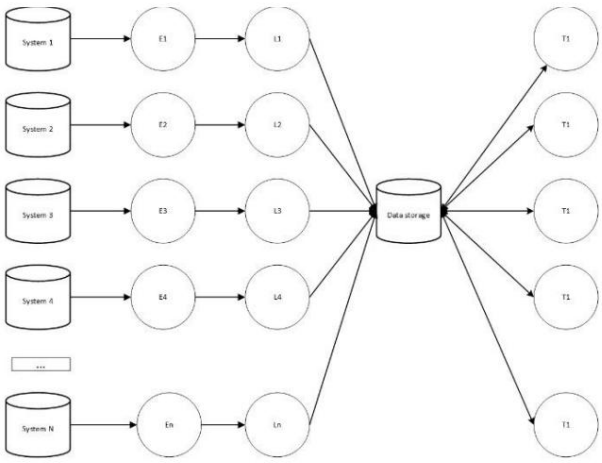

Fig. 1. Diagrama do processo ELT.

Como exemplo de processos, que ocorrem durante a fase de transformação de dados, toma-se parte da etapa de transformação final do processo ELT, desenvolvida para a construção do reporting analítico [11] do banco [12]. Esta etapa consiste em 14 procedimentos do PostgreSQL [13].

A solução inicial e mais trivial será realizar [14] este conjunto de procedimentos sequencialmente. Este gráfico é mostrado na Figura 2.

979-8-3503-9952-3/23/$31,00 ©2023 IEEE

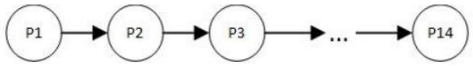

Fig. 2. Conjunto de procedimentos.

Para estimar o tempo de execução sequencial deste conjunto de procedimentos, precisamos usar a seguinte fórmula:

$$= \ddot{y}_{=1} = tp1 + tp2 + tp3 + \dots + tpn \quad \ddot{y}\ddot{y}\ddot{y}$$

O tempo experimental de cada procedimento tpn é mostrado é o Tabela I:

TABELA I. TEMPO DE EXECUÇÃO DO PROCEDIMENTO

Procedimento	Tempo de execução
1	
P2	
P3	
P4	
P5	
P6	
P7	
P8	
P9	
P10	
P11	**(segundos)** 30 120 90 110 60 70 100 76 194 181 531
P12	11
P13	189
P14	300

Usando a fórmula obtida anteriormente, o tempo total de todo o processo será:

$$= \ddot{y}_{=1}^{14} = \ddot{y}_{=1} = 2062$$

4. OTIMIZAÇÃO DE PROCESSO ELT POR PARALELIZAÇÃO

Em primeiro lugar, para otimizar o processo ELT, é necessário paralelizar a execução de cadeias de processos individuais. Por exemplo, propõe-se considerar duas opções para realizar encadeamentos consecutivos: sequencial e paralelizado.

Como exemplo, toma-se o seguinte conjunto de procedimentos apresentados na Tabela II. As dependências entre os procedimentos são mostradas na Figura 3.

TABELA II. EXEMPLO DE PROCEDIMENTOS POSTGRESQL

Procedimento	***Tempo de funcionamento***
P1	
P2	
P3	
P4	
P5	
P6	***(segundos)*** 60 42 66 66 70 20

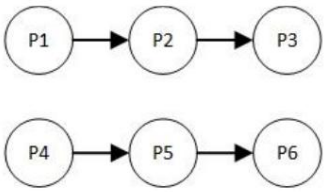

Fig. 3. Exemplo de dependências entre procedimentos.

Quando executado sequencialmente, o tempo total será tomado pela fórmula (1) e será igual a:

$$\text{Marca} = \ddot{y}_{6} = tp1 + tp2 + tp3 + tp4 + tp5 + tp6 = 60 + 42 + 66 + 66 + 70 + 20 = 324 \text{ segundos}$$

Ao paralelizar duas cadeias independentes, o tempo de execução será calculado usando a seguinte fórmula:

$$= \max(\ddot{y}_{=1}, \ddot{y}^{1}_{2=1+1}) \quad \ddot{y}\ddot{y}\ddot{y}\ddot{y}$$

O cálculo do tempo de execução para este exemplo será igual a 128 segundos.

Nesse caso, o tempo de execução paralela da requisição é quase 3 vezes menor que o tempo sequencial, o que mostra que, para otimizar o tempo de execução dos processos ELT, é necessário utilizar o método de paralelização de cadeias de processos independentes.

V. ALGORITMO DE OTIMIZAÇÃO DE TEMPO DE PROCESSO ELT

O PostgreSQL não possui a funcionalidade necessária para executar [15] esta tarefa e, portanto, é necessário o uso de software adicional.

Como opção mais simples para resolver este problema, propõe-se a utilização das ferramentas da linguagem de programação Python de alto nível e sua biblioteca Threading.

Threading é um módulo padrão que vem com um interpretador. O programador não precisa instalar, basta conectar o módulo usando o comando: import threading.

É possível trabalhar com threads criando instâncias da classe Thread. Para criar uma thread separada, é necessário criar uma instância da classe e aplicar nela o método start().

O uso dessas ferramentas permite paralelizar procedimentos [16] escritos em PostgreSQL.

VI. ALGORITMO DE PARALELIZAÇÃO DE PROCEDIMENTO

Passo 1. Crie uma matriz de interação entre procedimentos e tabelas.

Cada um dos procedimentos executa um conjunto específico de algoritmos, enquanto interage com várias tabelas. Todos os relacionamentos existentes entre as tabelas devem ser identificados. Para definir um conjunto de relacionamentos, é necessário criar uma matriz de mapeamento entre as tabelas utilizadas e as instruções SQL que nelas são executadas.

Etapa 2. Compilando conjuntos de dependências entre tabelas.

Esta etapa é preparatória para a etapa 3. É necessário desenhar as dependências entre as tabelas no formato: Ta -> Tb Tc Td.

É possível obter um conjunto de dados de dependência da matriz de interação construída no passo 1. A dependência é determinada pela seguinte regra:

Regra 1. Se no procedimento Pa for realizada a ação inserir, atualizar ou excluir na tabela Ta e uma ação selecionar para a tabela Tb, então a tabela Ta será uma tabela dependente em Tb e este link será registrado da seguinte forma: Tb -> Ta.

Etapa 3. Compilando conjuntos de dependências entre procedimentos.

Com base no conjunto de dependências entre as tabelas obtidas no passo 2 e nas matrizes de interação obtidas no passo 1, é necessário criar conjuntos de dependências entre os procedimentos. Isso é feito de acordo com a seguinte regra:

Regra 2. Se no procedimento Pa a ação inserir, atualizar ou excluir for executada na tabela Ta, então todos os procedimentos nos quais para a tabela Ta usar a ação selecionar serão dependentes do procedimento Pa.

Etapa 4. Otimização iterativa do processo ELT criando threads.

Após obter um conjunto de dependências entre procedimentos, é necessário compor threads iterativamente. Para fazer isso, é necessário a cada passo escrever aqueles procedimentos que não são dependentes de nenhum dos conjuntos obtidos no passo 3. Todos os procedimentos escritos por tal mecanismo irão para um fluxo cujo número corresponde ao número da iteração . Todos os procedimentos no mesmo thread serão executados em paralelo.

Exemplo de dependências entre procedimentos é mostrado na Tabela III:

TABELA III. TEMPO DE EXECUÇÃO DO PROCEDIMENTO

Procedimento	***inserir***	***atualizar***	***excluir***	***selecionar***
P1	ÿ2			ÿ1
P2	ÿ3, ÿ4	ÿ3	ÿ3	ÿ2
P3	ÿ5			ÿ3, ÿ4
P4	ÿ6			ÿ2
P5	ÿ7			ÿ5
P6	ÿ8			ÿ1, ÿ2, ÿ5, ÿ6 ÿ3,
P7	ÿ9			ÿ4, ÿ7, ÿ8 ÿ6 ÿ12,
P8	ÿ11		ÿ11	
P9	ÿ10			ÿ13 ÿ14
P10	ÿ13	ÿ13		ÿ15,
P11	ÿ9			ÿ10, ÿ11 ÿ11 ÿ10
P12	ÿ9		ÿ11	ÿ7
P13	ÿ9			
P14	ÿ15			

A instrução select para o data mart final deve ser executada após a conclusão da inserção, atualização e exclusão.
Além disso, as instruções de inserção, atualização e exclusão não podem ser executadas em paralelo devido à probabilidade de violações da integridade dos dados. Os relacionamentos para tabelas são:

ÿ1 -> T2, ÿ8; T2 -> T3, T4, ÿ6, ÿ8; T3 -> ÿ5, ÿ9; T4 -> T5, ÿ9; T5 -> T7, T8; T6 -> T8, ÿ11; ÿ7 -> ÿ9, ÿ15; ÿ8 -> ÿ9; ÿ9 -> {}; ÿ10 -> ÿ9; ÿ11 -> ÿ9; ÿ12 -> ÿ10; ÿ13 -> ÿ10; ÿ14 -> ÿ13; ÿ15 -> ÿ9

Com base nesses dados, é possível construir links entre os procedimentos:

P1-> P2, P4, P6; P2-> P3, P7; P3 -> P5, P6; P4-> P6, P8; P5-> P7, P14; P6-> P7; P7-> {}; P8-> P11, P12; P9-> P11, P13; P10-> P9; P11-> P7; P12-> P7; P13-> P7; P14-> P11

Para encontrar os procedimentos a serem executados na primeira thread, é necessário selecionar aqueles que não estão do lado direito da relação entre os procedimentos, tais procedimentos são P1 e P10. Esses procedimentos irão para o primeiro fluxo.

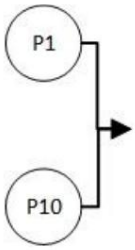

Fig. 4. Primeiro fluxo.

Removê-los do conjunto levará ao seguinte:

P2-> P3, P7; P3 -> P5, P6; P4-> P6, P8; P5-> P7, P14; P6-> P7; P7-> {}; P8-> P11, P12; P9-> P11, P13; P11-> P7; P12-> P7; P13-> P7; P14-> P11

Da mesma forma, é necessário selecionar aqueles que não estão do lado direito para selecionar os procedimentos no segundo thread. Tais procedimentos são P2, P4 e P9. Eles serão adicionados ao segundo fluxo.

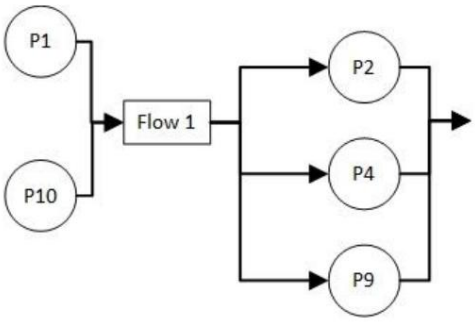

Fig. 5. Segundo fluxo.

Depois de removê-los do conjunto, obtém-se tal resultado:

P3 -> P5, P6; P5-> P7, P14; P6-> P7; P7-> {}; P8-> P11, P12; P11-> P7; P12-> P7; P13-> P7; P14-> P11

P3, P8 e P13 não estão do lado direito. Eles serão adicionados ao terceiro fluxo.

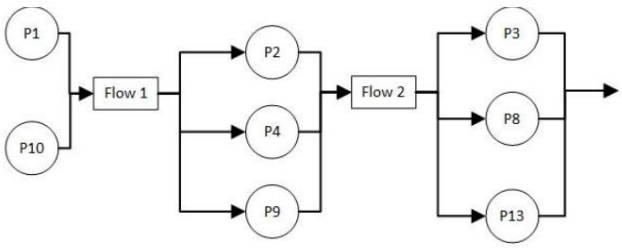

Fig. 6. Terceiro fluxo.

Próxima iteração:

P5-> P7, P14; P6-> P7; P7-> {}; P11-> P7; P12-> P7; P14-> P11

P12, P5 e P6 não estão do lado direito. Eles serão adicionado ao fluxo 4.

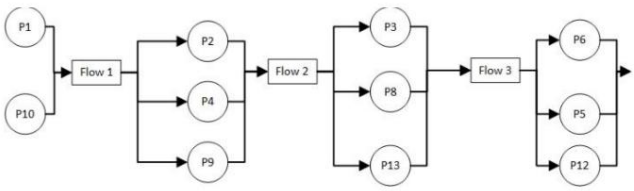

Fig. 7. Quinto fluxo.

Próxima iteração:

P7-> {}; P11-> P7

P7 depende da execução P11. P11 será adicionado ao sexto fluxo e P7 ao sétimo. A cadeia final é mostrada na Figura 8.

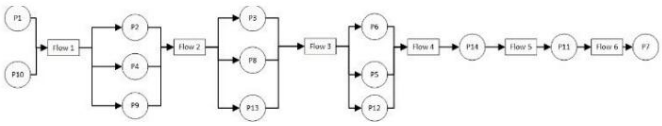

Fig. 8. Cadeia final.

Substituindo na fórmula 1, obtém-se o tempo de execução antes da otimização e depois:

$$= \ddot{y}_{=1} \quad = \ddot{y}_{=1}^{14} \quad = 2062 \text{ segundos}$$

O tempo total de operação paralela e sequencial também será considerado pela fórmula 1, mas ao invés do tempo de execução de um procedimento separado, é necessário tomar o tempo de execução da thread. Para encontrar o tempo de operação de cada fluxo, use a fórmula 2. Como resultado, será igual a 1376 segundos

VII. ALGORITMO DE PARALELIZAÇÃO DE PROCEDIMENTO APRIMORADO

Para cadeias de dependências mais complexas, terá maior relevância um algoritmo, no qual, além da paralelização dos procedimentos, é utilizada a paralelização passo a passo do processo ELT.

Passo 1. Crie uma matriz de interação entre procedimentos e tabelas.

Cada um dos procedimentos executa um conjunto específico de algoritmos, enquanto interage com várias tabelas. Todos os relacionamentos existentes entre as tabelas devem ser identificados. Para definir um conjunto de relacionamentos, é necessário criar uma matriz de mapeamento entre as tabelas utilizadas e as instruções SQL que nelas são executadas.

Etapa 2. Compilando conjuntos de dependências entre tabelas.

Esta etapa é preparatória para a etapa 3. É necessário desenhar as dependências entre as tabelas no formato: Ta -> Tb Tc Td.
É possível obter um conjunto de dados de dependência a partir de uma matriz de interação construída no passo 1. A dependência é determinada pela seguinte regra:

Regra 1. Se no procedimento Pa for realizada a ação inserir, atualizar ou excluir na tabela Ta e uma ação selecionar para a tabela Tb, então a tabela Ta será uma tabela dependente em Tb e este link será registrado da seguinte forma: Tb -> Ta.

Etapa 3. Compilando conjuntos de dependências entre procedimentos.

Com base no conjunto de dependências entre as tabelas obtido no passo 2 e nas matrizes de interação obtidas no passo 1 é necessário criar conjuntos de dependências entre os procedimentos. Isso é feito de acordo com a seguinte regra:

Regra 2. Se no procedimento Pa a ação inserir, atualizar ou excluir for executada na tabela Ta, então todos os procedimentos nos quais para a tabela Ta usar a ação selecionar serão dependentes do procedimento Pa.

Etapa 4. Otimização iterativa do processo ELT por paralelização passo a passo.

Após obter um conjunto de dependências entre procedimentos, é necessário compor iterativamente conjuntos de cadeias paralelas combinadas em um processo ELT. Este método introduz o conceito de dependência condicional, que indica o início de um procedimento dependente deste tipo não sequencialmente, mas após a execução de um determinado conjunto de procedimentos. Este tipo de link é denotado da seguinte forma: -> Pa Pb Pc e significa que o procedimento Pc será iniciado após os procedimentos Pa e Pb, não depende do tempo de procedimentos posteriores

Para uma versão melhorada do algoritmo anterior, será utilizado o princípio de paralelização passo a passo do processo ELT. Para fazer isso, crie blocos adicionais para procedimentos que não dependam da execução de subseqüentes.

Relacionamento de tabelas:

ÿ1 -> T2, ÿ8; T2 -> T3, T4, ÿ6, ÿ8; T3 -> ÿ5, ÿ9; T4 -> T5, ÿ9; T5 -> T7, T8; T6 -> T8, ÿ11; ÿ7 -> ÿ9, ÿ15; ÿ8 -> ÿ9; ÿ9 -> {}; ÿ10 -> ÿ9; ÿ11 -> ÿ9; ÿ12 -> ÿ10; ÿ13 -> ÿ10; ÿ14 -> ÿ13; ÿ15 -> ÿ9

Com base nesses dados, links entre os procedimentos:

P1-> P2, P4, P6; P2-> P3, P7; P3 -> P5, P6; P4-> P6, P8; P5-> P7, P14; P6-> P7; P7-> {}; P8-> P11, P12; P9-> P11, P13; P10-> P9; P11-> P7; P12-> P7; P13-> P7; P14-> P11

Semelhante ao método anterior, os procedimentos que não estão no lado direito são selecionados. Tais procedimentos são P1 e P10. Eles podem ser distribuídos em 2 unidades independentes diferentes.

Cadeias restantes:

P2-> P3, P7; P3 -> P5, P6; P4-> P6, P8; P5-> P7, P14; P6-> P7; P7-> {}; P8-> P11, P12; P9-> P11, P13; P11-> P7; P12-> P7; P13-> P7; P14-> P11

P2, P4 e P9 não estão do lado direito. Eles são adicionados aos blocos, com base em suas dependências, e os procedimentos que emanam deles para o lado direito da próxima etapa também são adicionados, portanto, serão considerados controversos:

Bloco 1: P1 -> (P2 ?-> P3, P7) || (P4 ?-> P6, P8)

Bloco 2: P10 -> P9 ?-> P11, P13

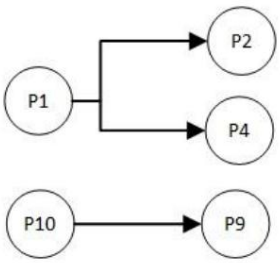

Fig. 9. Segundo passo das iterações.

Procedimentos restantes:

P3 -> P5, P6; P5-> P7, P14; P6-> P7; P7-> {}; P8-> P11, P12; P11-> P7; P12-> P7; P13-> P7; P14-> P11

P3, P8, P13 não estão do lado direito. Esses procedimentos são colocados na próxima iteração e os procedimentos controversos restantes na próxima etapa (se os procedimentos forem duplicados, deixamos apenas uma variação) são executados:

Bloco 1: P1 -> ((P2 -> P3) -> P6, P5, P7) || ((P4 -> P8) -> (P11, P12))

Bloco 2: P10 -> P9 -> P13 ?-> P11, P7

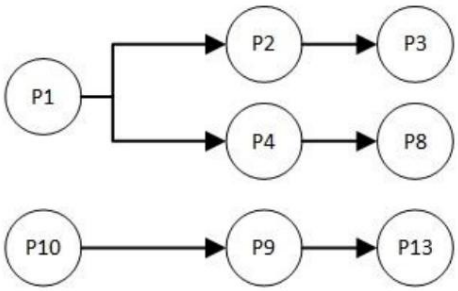

Fig. 10. Terceira etapa das iterações.

Conjunto restante:

P5-> P7, P14; P6-> P7; P7-> {}; P11-> P7; P12-> P7; P14-> P11

P6, P5 e P12 serão adicionados à nossa cadeia por analogia com as etapas anteriores. Como P6 e P5 procedem do mesmo procedimento, o cálculo paralelo será separado:

Bloco 1: P1 -> (((P2 -> P3) -> (P6 || P5 -> P14)) ?-> P7) || (((P4 -> P8) -> (P12 ?-> P7)) -> P11)

Bloco 2: P10 -> P9 -> P13 ?-> P11, P7

Observe que no bloco 1 o P7 aparece tanto na primeira parte do circuito paralelo quanto na segunda. Depois que toda a cadeia estiver concluída:

Bloco 1: P1 -> ((((P2 -> P3) -> (P6 || P5 ?-> P14))) || (((P4 - > P8) -> (P12)) ?-> P11)) ?-> P7

Bloco 2: P10 -> P9 -> P13 ?-> P11, P7

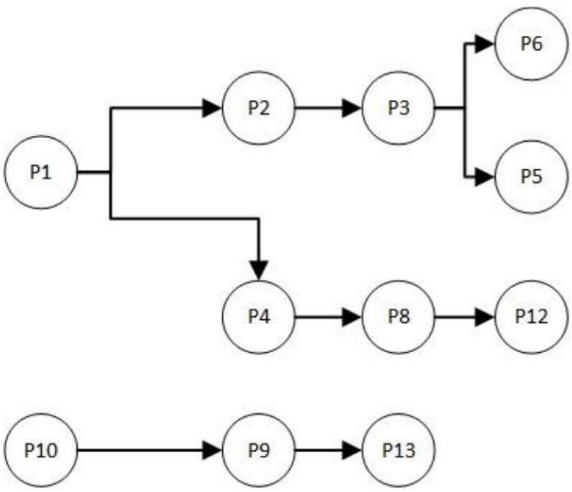

Fig. 11. Quarto passo de iterações.

Conjunto restante:

P7-> {}; P11-> P7; P14-> P11

Adicionando P14 aos nossos blocos:

Bloco 1: P1 -> ((((P2 -> P3) -> (P6 || P5-> P14 ?-> P11))) || (((P4 -> P8) -> (P12) ?|| P11))) ?-> P7

Bloco 2: P10 -> P9 -> P13 ?-> P7

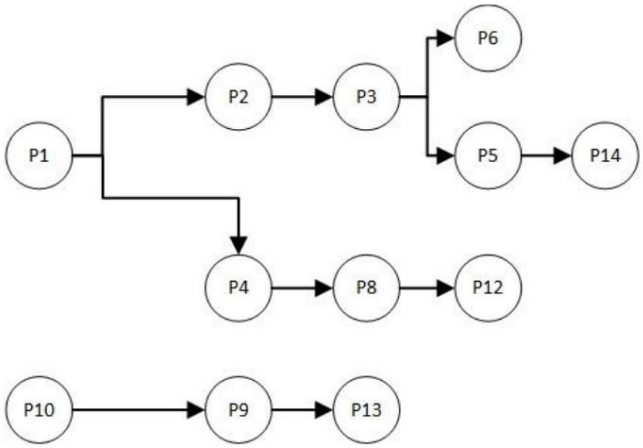

Fig. 12. Etapa cinco das iterações.

Conjunto restante:

P7-> {}; P11-> P7

Adicione P11 aos nossos blocos:

Bloco 1: P1 -> ((((P2 -> P3) -> (P6 || P5-> P14 ?-> P11))) || (((P4 -> P8) -> (P12) || P11))) ?-> P7

Bloco 2: P10 -> P9 -> P13 ?-> P7

Observe que o P11 apareceu duas vezes no primeiro bloco. No primeiro caso, é realizado em série e no segundo em paralelo. Nesse sentido, é retirado após a execução de blocos paralelos e indica a execução sequencial condicional como um tipo de conexão. No índice inferior deste tipo de relacionamento, serão registrados os procedimentos após os quais serão registrados os subsequentes.

Bloco 1: P1 -> ((((P2 -> P3) -> (P6 || P5-> P14))) || (((P4 - > P8) -> (P12))))) -> P8P14 P11 ?-> P7

Bloco 2: P10 -> P9 -> P13 ?-> P7

No conjunto restante, apenas o procedimento P7 é observado:

Bloco 1: P1 -> (((((P2 -> P3) -> (P6 || P5-> P14))) || (((P4 - > P8) -> (P12))))) ->P8P14 P11 -> P7

Bloco 2: P10 -> P9 -> P13 -> P7

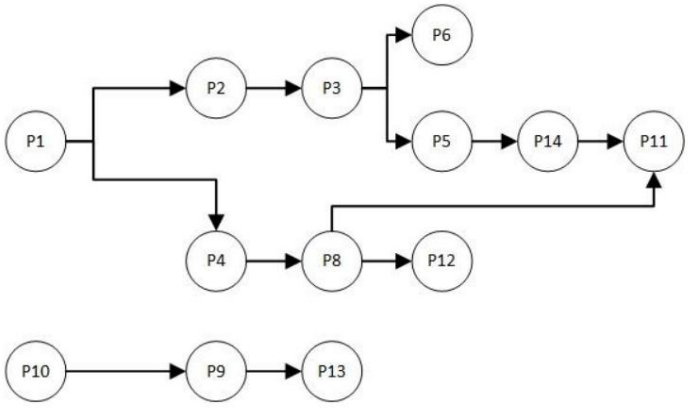

Fig. 13. Etapa seis das iterações.

O procedimento P7 aparece em ambos os blocos. Nesse sentido, é possível combinar dois blocos em um:

(P1 -> ((((P2 -> P3) -> (P6 || P5-> P14))) || (((P4 -> P8) -> (P12))))) ->P8P14 P11) || (P10 -> P9 -> P13) -> P7

Diagrama final do processo ELT:

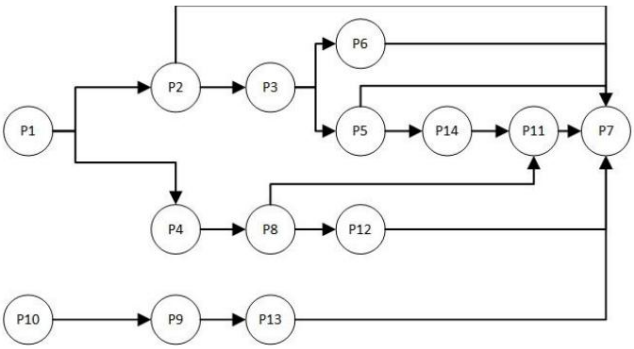

Fig. 14. Esquema final.

Para calcular o tempo, são utilizadas as fórmulas 1 e 2, contando as partes consecutivas e paralelas do esquema, respectivamente.

A dependência condicional será considerada pela fórmula:

$$Tagg = B1 + P1 - Tseq \quad (1)$$

onde B1 – o bloco do qual o procedimento P1 convencionalmente depende,

P1 – um procedimento que depende condicionalmente de P1,

Tseq – tempo de execução do bloco de procedimentos, que segue sequencialmente após a execução dos procedimentos, dos quais P1 depende condicionalmente

Como resultado, obtém-se o tempo de operação desta cadeia igual a 1131 segundos.

VIII. COMPARAÇÃO DO PROCESSO ELT

O diagrama de barras comparando a execução de vários tipos de algoritmos é mostrado na Figura 15:

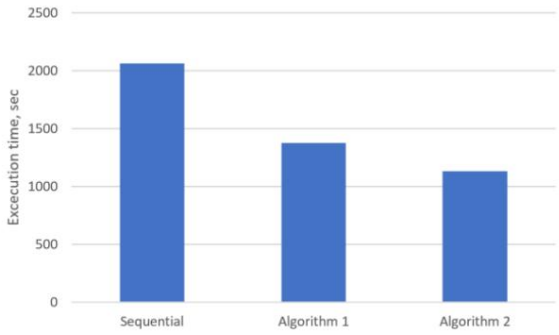

Fig. 15. Comparação de algoritmos ELT.

Como resultado da comparação, verificou-se que o algoritmo 1 e o algoritmo 2 permitem um aumento de desempenho de mais de 30 por cento em comparação com um algoritmo sequencial. O melhor foi o algoritmo número 2, cujo tempo de operação foi de 1131 segundos.

IX. CONCLUSÃO

Este artigo discutiu várias maneiras de construir um processo ELT. São considerados algoritmos para execução de procedimentos PostgreSQL: sequencial, algoritmo de paralelização e algoritmo de paralelização aprimorada. Como resultado do estudo, foi revelado que um algoritmo de paralelização aprimorado apresenta o melhor resultado. Pode ser usado em processos como trend identificação [17], controle biônico [18], assinaturas multipartidárias quânticas [19] ou abordagens de IA [20] onde o tempo de execução é crítico, com recursos suficientes para realizar cálculos paralelos. Mas o método sequencial também pode ser útil quando você precisa obter resultados em um tempo limitado porque é fácil de configurar.

REFERÊNCIAS

[1] EA Eliseeva, BS Goryachkin, MV Vinogradova e MV Chernenkiy, “Estimando o tempo de execução de consultas de pesquisa em bancos de dados NoSQL e relacionais de objetos”, revista científica internacional "Dynamics of complex systems - XXI century": Radiotekhnika Publishing House, Moscou, 2022, nº 2, pp. 44-51.

[2] DA Lychagin, MV Vinogradova, DS Gudilin e AE Zvonarev, “Organização do armazenamento de dados nas condições de substituição de importações,” Inteligência Artificial em Sistemas Automatizados de Gerenciamento e Processamento de Dados, vol.1, pp.364-370, 2022 . [em russo]

[3] M. Madhikerrni e K. Främling, “Método de descoberta de dados para Extract-Transform-Load”, em 2019 IEEE 10ª Conferência Internacional sobre Tecnologias de Manufatura Mecânicas e Inteligentes (ICMIMT), 2019, 205-212, doi: 10.1109/ICMIMT .2019.8712027.pp.

[4] Q. Hanlin, J. Xianzhen e Z. Xianrong, "Pesquisa sobre Extração, Transformação e Carregamento (ETL) em Land and Resources Star Schema Data Warehouse", em 2012 Quinto Simpósio Internacional de Inteligência Computacional e Design, 2012, pp. 120-123, doi: 10.1109/ ISCID.2012.38.

[5] A. Wibowo, “Problemas e soluções disponíveis no estágio de extração, transformação e carregamento em armazenamento de dados quase em tempo real (um estudo de literatura)”, em 2015 Seminário Internacional sobre Tecnologia Inteligente e Suas Aplicações (ISITIA), 2015, pp. 345-350, doi: 10.1109/ISITIA.2015.7220004.

[6] H. Morris et al., "Trazendo objetos de negócios para a tecnologia Extract-Transform Load (ETL)", em 2008 IEEE International Conference on e-Business Engineering, 2008, pp. 709-714, doi: 10.1109/ICEBE. 2008.72.

[7] Kumari Deepika e M. Tech, “Otimização do Processo ETL Usando Técnicas de Particionamento e Paralelização,” Jornal Internacional de Pesquisa Avançada em Ciência da Computação, vol. 8, não. 3, 2017, doi: 10.26483/ jarcs.v8i3.3093.

[8] SMF Ali e R. Wrembel, “Do projeto conceitual à otimização de desempenho de fluxos de trabalho ETL: estado atual da pesquisa e problemas em aberto,” The VLDB Journal, vol. 26, pp. 777-801, 2017.

[9] H. Homayouni, “Testing Extract-Transform-Load Process in Data Warehouse Systems,” 2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), 2018, pp. 158-161, doi: 10.1109/ ISSREW.2018.000- 6.

[10] SK Bansal, “Rumo a uma extração-transformação-carga semântica (ETL) Framework for Big Data Integration,” em 2014 IEEE International Congress on Big Data, 2014, pp. 522-529, doi: 10.1109/ BigData.Congress.2014.82.

[11] Munawar, “Extract Transform Loading (ETL) Based Data Quality for Data Warehouse Development,” em 2021 1st International Conference on Computer Science and Artificial Intelligence (ICCSAI), 2021, pp. 373-378, doi: 10.1109/ICCSAI53272. 2021.9609770.

[12] AE Zvonarev, MV Vinogradova, DS Gudilin e DA Lychagin, “Uso de análise de processo no setor bancário,” Inteligência Artificial em Sistemas Automatizados de Gerenciamento e Processamento de Dados, vol.1, pp.47-52, 2022.

[13] YA Grigoriev, “Estimativa do tempo de execução de consultas SQL a bancos de dados”, Engenharia Mecânica e Tecnologias de Computação, no. 01, 2012, http://technomag.edu.ru/doc/296143.html (acessado em 7 de fevereiro de 2031). [em russo]

[14] Adnan, AA Ilham e S. Usman, “Análise de desempenho de extrair, transformar, carregar (ETL) no apache Hadoop no topo do armazenamento NAS usando ISCSI”, em 2017 4ª Conferência Internacional sobre Aplicativos de Computador e Tecnologia de Processamento de Informação (CAIPT), 2017, pp. 1-5, doi: 10.1109/CAIPT.2017.8320716.

[15] EA Eliseeva, BS Goryachkin e MV Vinogradova, “Investigação do desempenho do DBMS ao trabalhar com cluster

bancos de dados baseados em análise ergonômica," Jornal científico e educacional para estudantes e professores STUDNET: Editora LLC "Ciência Eletrônica", Moscou, vol.5, no. 4, pp. 2889-2910, 2022.

[16] S. Sakulin, A. Alfimtsev, K. Kvitchenko, L. Dobkacz, Y. Kalgin e I. Lychkov, "Abordagem de detecção de anomalias de rede baseada em votação ponderada", Jornal internacional de segurança e privacidade da informação, vol. 16. pp.1-17, 2022. doi: 10.4018/IJISP.2022010105.

[17] Iu Butenko, I. Telnova e V. Garazha, "Métodos de Identificação de Tendências de Pesquisa Científica Prospectiva (Baseado na Análise de Publicações Relacionadas a Combustível Gás)", Documentação Automática e Linguística Matemática, vol. 56, pp. 11-25, 2022. doi: 10.3103/S0005105522010034.

[18] A. Briko, V. Kapravchuk, A. Kobelev, A. Hammoud, S. Leonhardt, C. Ngo, Y. Gulyaev e S. Shchukin, "Uma forma de controle biônico baseado em EI, EMG e FMG Signals", Sensors, vol. 22, não. 1, pág. 152, 2021, doi: 10.3390/s22010152.

[19] E. Kiktenko, A. Zelenetsky e A. Fedorov, "Assinaturas multipartidárias quânticas práticas usando redes de distribuição de chaves quânticas," Física Rev. A, vol. 105, pág. 012408, 2021. See More

[20] A. Masalimova, M. Khvatova, L. Chikileva, E. Zvyagintseva, V. Stepanova e M. Melnik, "ensino à distância no ensino superior durante a Covid-19," Frontiers in Education, vol. 7, 2022, doi:10.3389/feduc.2022.822958.