ConSQL: Consentimentos em SQL para o Processamento de Consultas Orientado a Propósitos

Ítalo C. de Abreu[1], **Francisco D. B. S. Praciano**[1], **Paulo R. P. Amora**[1], **Javam C. Machado**[1]

[1]Laboratório de Sistemas e Bancos de Dados (LSBD)
DC/UFC – CEP 60440-900 – Fortaleza – CE – Brasil

{italo.abreu, daniel.praciano, paulo.amora, javam.machado}@lsbd.ufc.br

Resumo. *Em vista da crescente necessidade de garantir a proteção de dados pessoais e a privacidade dos indivíduos em um mundo cada vez mais informatizado é crucial que os SGBDs deem suporte a técnicas que facilitem a aplicação dessas garantias quando envolvem dados sob sua gestão, como o direito ao consentimento, onde o indivíduo define os propósitos de acesso e de processamento dos seus dados. Neste trabalho, apresentamos uma extensão da gramática SQL, chamada ConSQL, que permite a definição e a manutenção de propósitos de acesso a relações, tuplas e atributos de um banco de dados relacional. ConSQL, completamente integrada ao processo de reconhecimento de SQL no PostgreSQL, gera estruturas de dados associadas ao esquema de um banco de dados para posterior utilização na verificação de propósitos em tempo de execução de consultas. O artigo também descreve, por meio de exemplos expressos em ConSQL, consentimentos de usuários e mostra a utilidade deles no processamento de consultas que assegura a aplicação de propósitos.*

1. Introdução

A quantidade de dados dos usuários em posse das empresas possibilita a construção de vários tipos de sistemas, como os sistemas de aprendizagem de máquina, porém traz consigo o seguinte problema: As pessoas têm realmente controle sobre seus dados pessoais? [Solove and Citron 2017]. Tendo em vista esse cenário, os governos de diversos países ao redor do mundo levantaram preocupação com esses fatos e, consequentemente, começaram a realizar estudos focados em atualizar as legislações responsáveis por normatizar a maneira pela qual os dados dos seus cidadãos devem ser coletados e armazenados e, ainda, quais os direitos e deveres que os diversos atores, interessados em acessar ou processar esses dados, têm sobre eles. Para exemplificar, podemos citar algumas legislações: *General Data Protection Regulation (GDPR)* [General Data Protection Regulation 2016], *California Consumer Privacy Act (CCPA)* [California 2018] e Lei Geral de Proteção de Dados Pessoais (LGPD) [Brasil 2018].

No Brasil, a LGPD guarda semelhanças à GDPR. Sinteticamente, ambas as leis são baseadas em pelo menos quatro princípios fundamentais, a saber: consentimento, direito de ser esquecido, portabilidade e privacidade. Neste trabalho, estamos interessados no consentimento, que garante que os dados de um usuário serão processados somente para os fins específicos que o próprio usuário deu o consentimento ou a permissão. Assim os usuários devem dar permissão às organizações para que estas possam utilizar os dados em um dado tipo de processamento. Portanto, deve existir um controle de acesso sobre quais dados podem ser disponibilizados para um certo objetivo de processamento.

Os Sistemas Gerenciadores de Banco de Dados (SGBDs) são utilizados para armazenar os dados e, consequentemente, devem apresentar algum mecanismo que permita o controle do consentimento dos usuários sobre os seus dados. A Linguagem de Consulta Estruturada (*Structure Query Language - SQL*) já permite especificar algum tipo de controle de acesso sobre os dados, mas de maneira implícita, sendo que não há uma especificação de linguagem para lidar com o consentimento requisitado pela LGPD de maneira explícita. O presente trabalho tem como objetivo principal estudar e avaliar uma estratégia de especificação de consentimento por meio da extensão da gramática SQL, permitindo então o uso do consentimento de maneira explícita no ambiente dos SGBDs.

As contribuições do trabalho são (1) uma extensão da gramática SQL para considerar o conceito de consentimento de propósitos e (2) um exemplo de uso dessa extensão inserido no âmbito de um SGBD. A extensão da SQL suporta a especificação de propósitos associados aos dados para posterior verificação em tempo de processamento de consulta. O exemplo facilita a compreensão dos propósitos e mostra um caso simples de uso.

2. Processamento de Consultas Orientado a Propósito

No processamento de consultas tradicional de um SGBD, um usuário submete uma consulta SQL ao sistema e este, por sua vez, realiza as etapas de reconhecimento, otimização e execução para gerar a resposta correta [Elmasri and Navathe 2000]. Neste tipo de arquitetura, a resposta é formada por todos as tuplas que satisfazem às condições impostas pela consulta, como predicados de seleção, condições de junção, e eventuais outras restrições. Ao analisarmos cuidadosamente as normas de proteção e privacidade de dados, como a LGPD e a GDPR, identificamos uma nova restrição de acesso aos dados. Resultados de consultas devem ser consistentes com propósitos decorrentes de consentimentos explicitamente definidos por usuários de aplicações. Para possibilitar que SGBDs lidem com o conceito de consentimento, desenhamos uma arquitetura de processamento de consultas orientado a propósito mostrada na Figura 1.

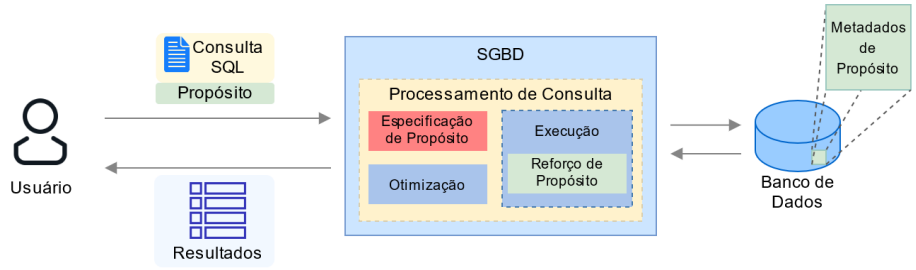

Figura 1. Fluxo de consultas com consentimentos/propósitos dos usuários.

No fluxo apresentado na Figura 1, o usuário encaminha além da consulta SQL, os propósitos dos quais aquela consulta faz parte. A sintaxe da consulta SQL, na interface do SGBD, não é alterada. O propósito da consulta é automaticamente formado pelo perfil do usuário e a aplicação com a conexão ativa. O ConSQL adiciona dois novos módulos ao processamento de consulta: Especificação de Propósito e Reforço de Propósito. Enquanto que o primeiro é um submódulo do módulo de Execução e é responsável por garantir que os dados utilizados durante a execução da consulta serão somente os que possuem os propósitos apresentados na consulta, o segundo é encarregado de realizar todo

o gerenciamento de propósitos. O módulo de especificação de propósito tem as funções de armazenar os propósitos existentes em um banco de dados, verificar se os propósitos presentes na consulta são sintaticamente e semanticamente corretos, e analisar a consulta e averiguar quais partes da consulta cada propósito está relacionado. Para tanto, este módulo irá armazenar e, posteriormente, utilizar as informações guardadas nos Metadados de Propósito.

Os módulos de extensão e metadados da Figura 1 estão parcialmente implementados no PostgreSQL, notadamente o módulo de especificação de propósito, foco deste trabalho. Nele apresentamos uma extensão da gramática SQL para dar suporte ao conceito de propósito e mostramos, por meio de um exemplo, a sua utilidade. O detalhamento completo da arquitetura e os resultados parciais de eficácia e desempenho do protótipo estão fora do escopo deste artigo.

3. Definição de Propósitos

O ConSQL representa propósitos como conjuntos de dados visto que esse modelo nos permite lidar com os propósitos de maneira simplificada e, como mostram [Kraska et al. 2019], é uma maneira de reduzir o espaço de armazenamento ocupado pelos metadados de propósito. Dessa forma, os propósitos definidos para um banco de dados são armazenados em estruturas de dados associadas ao esquema, de modo a permitir a recuperação quando necessário.

O ConSQL adiciona as construções abaixo à linguagem SQL. Criar e manter propósitos no escopo de um banco de dados é o principal objetivo. Além disso, pode-se associar propósitos a objetos do esquema, especificando a granularidade, como uma tabela, um atributo de uma tabela e até mesmo um conjunto de tuplas que satisfaz um predicado. Com isso, ConSQL garante que os propósitos criados estejam ligados aos dados de modo que esses dados sejam utilizados somente para os seus respectivos propósitos atrelados. Os novos comandos seguem a gramática definida mais adiante e, quando utilizados, adicionam metadados ao esquema de um banco de dados por meio de um conjunto P de propósitos.

CREATE PURPOSE p { ON SCHEMA s }	(1)
UPDATE PURPOSE p TO new_name	(2)
DROP PURPOSE p { ON SCHEMA s }	(3)
SET PURPOSE p TO TABLE t	(4)
SET PURPOSE p TO ROWS ON TABLE t { WHERE $P(x)$ }	(5)
SET PURPOSE p TO COLUMN c ON TABLE t	(6)
DELETE PURPOSE p FROM TABLE t	(7)
DELETE PURPOSE p FROM ROWS ON table t { WHERE $P(x)$ }	(8)
DELETE PURPOSE p FROM COLUMN c ON table t	(9)

Na construção (1), o usuário do SGBD insere uma cadeia de caracteres p que nomeia o propósito. As chaves representam uma entrada opcional. Dentro do esquema do banco de dados, o propósito recebe um identificador numérico único que será usado pelo SGBD para procedimentos internos. O nome p é vinculado a esse identificador numérico. Se um nome de esquema s for inserido, o propósito é adicionado ao conjunto P desse esquema s. Caso contrário, o propósito é adicionado ao conjunto do esquema da conexão atual do SGBD. Já na construção (2), o propósito p pertencente ao conjunto P será renomeado para *new_name* enquanto o identificador numérico do propósito permanece inalterado, uma vez que somente o nome do propósito será sobrescrito. Por fim, a

construção (3) permite que um propósito p e todas as suas referências sejam excluídos em cascata do esquema s, se este for inserido, ou do esquema atual, caso contrário. Em resumo, esses são as extensões necessárias na gramática SQL para permitir a definição de propósitos na linguagem.

A construção (4) é utilizada para fazer a ligação entre um propósito p e uma relação t. Ao receber esse comando, o SGBD cria, caso não exista, um mapa H que liga tabelas a um conjunto de propósito, utilizando o identificador único da tabela como chave e o identificador do propósito como valor. Caso t já esteja em H, e p' seja o propósito anterior, insere $p' \cup p$.

O comando (5) associa um propósito p a todas as tuplas x da relação t onde o predicado definido na cláusula opcional WHERE é atendido. Para isso, é criado, caso não exista, um mapa $h(t)$ para a relação t, que é armazenada nos metadados dessa relação. Caso x não esteja em $h(t)$, o par <x, p> é inserido, caso contrário, seja <x, p'> o par presente em $h(t)$, com p' o conjunto de propósitos atrelados a x, excluímos <x, p'> e inserimos <x, $p' \cup p$> a $h(t)$. É importante deixar claro que há somente um mapa H no banco de dados, já os mapas $h(t)$ são criadas para cada relação do banco de dados.

Para uma granularidade mais fina, o comando (6) permite associar um propósito p para um atributo c. Para isso, o SGBD cria, caso não exista, um vetor V, no qual cada posição representa um atributo da tabela, e guarda-o junto aos metadados da relação t. Os valores nas posições do vetor são instanciados com $\emptyset$ quando ele é criado. Em seguida, seja p' o valor anterior do propósito na posição c, p' recebe $p' \cup p$.

Os comandos (7), (8) e (9) são inversos aos comandos (4), (5) e (6) respectivamente. Ou seja, em vez de inserirmos a união, inserimos a diferença $p' \backslash p$, efetivamente removendo o propósito p do conjunto de propósitos da relação, tupla ou coluna. No entanto, caso $p' \backslash p$ resulte no conjunto vazio, removemos a relação t de H em (7) e removemos a tupla x de $h(t)$ em (8). Já em (9), p' recebe o resultado da operação $p' \backslash p$, isto é, uma representação do conjunto vazio.

3.1. Gramática de Propósitos

Para produzir a linguagem apresentada, estendemos a gramática SQL com as produções abaixo. O símbolo não-terminal <stmt> é o responsável por produzir todos os comandos SQL, de criação de tabelas a seleções e junções. Representamos isso com reticências (...). Primeiro, adicionamos o símbolo terminal PURPOSE. Os demais símbolos terminais presente no ConSQL (CREATE, UPDATE, DROP, SET, DELETE, ON, TO, TABLE, SCHEMA, ROWS, FROM e COLUMN) já fazem parte da SQL. Nossa gramática acrescenta cinco novas produções a este símbolo: <CreatePurposeStmt>, <UpdatePurposeStmt>, <DropPurposeStmt>, SetPurposeStmt> e <DeletePurposeStmt>. Ademais, a SQL já possui um símbolo que reconhece nomes de tabelas e de colunas, representados por <table_ref> e <columnref>, portando, basta acrescentar um símbolo que reconhece nome de propósitos. <SCONST> representa um símbolo padrão da SQL que reconhece cadeia de caracteres. Já o simbolo <where_clause_p> é uma simplificação da cláusula WHERE presente na gramática do SELECT cuja diferença entre elas é a de que <where_clause_p> só aceita as cláusulas lógicas, ou seja, não aceita operações mais complexas como seleções alinhadas e USING. A implementação atual do ConSQL aceita apenas cláusulas do tipo $table\_alias.column\_name = number$.

⟨*stmt*⟩ ::= ...
| ⟨*CreatePurposeStmt*⟩
| ⟨*UpdatePurposeStmt*⟩
| ⟨*DropPurposeStmt*⟩
| ⟨*SetPurposeStmt*⟩
| ⟨*DeletePurposeStmt*⟩

⟨*CreatePurposeStmt*⟩ ::= CREATE PURPOSE ⟨*purpose_ref*⟩
| CREATE PURPOSE ⟨*purpose_ref*⟩ ON SCHEMA ⟨*name*⟩

⟨*UpdatePurposeStmt*⟩ ::= UPDATE PURPOSE ⟨*purpose_ref*⟩ TO ⟨*Sconst*⟩

⟨*DropPurposeStmt*⟩ ::= DROP PURPOSE ⟨*purpose_ref*⟩
| CREATE PURPOSE ⟨*purpose_ref*⟩ ON SCHEMA ⟨*name*⟩

⟨*SetPurposeStmt*⟩ ::= SET PURPOSE ⟨*purpose_ref*⟩ TO TABLE ⟨*table_ref*⟩
| SET PURPOSE ⟨*purpose_ref*⟩ TO ROWS ON TABLE ⟨*table_ref*⟩
| SET PURPOSE ⟨*purpose_ref*⟩ TO ROWS ON TABLE ⟨*table_ref*⟩ ⟨*where_clause_p*⟩
| SET PURPOSE ⟨*purpose_ref*⟩ TO COLUMN ⟨*columnref*⟩ ON TABLE ⟨*table_ref*⟩

⟨*DeletePurposeStmt*⟩ ::= DELETE PURPOSE ⟨*purpose_ref*⟩ FROM TABLE t ⟨*statement*⟩
| DELETE PURPOSE ⟨*purpose_ref*⟩ FROM ROWS ON TABLE ⟨*table_ref*⟩
| DELETE PURPOSE ⟨*purpose_ref*⟩ FROM ROWS ON TABLE ⟨*table_ref*⟩ ⟨*where_clause_p*⟩
| DELETE PURPOSE ⟨*purpose_ref*⟩ FROM COLUMN ⟨*columnref*⟩ ON TABLE ⟨*table_ref*⟩

⟨*purpose_ref*⟩ ::= ⟨*SCONST*⟩

Exemplo. Para entendermos como essa linguagem pode ser usada, suponha que um administrador de banco de dados (DBA) de uma instituição sabe que certo setor de pesquisa deseja acessar dados dos membros da instituição, presentes na tabela Membros, com o intuito de realizar uma pesquisa estatística. Suponha ainda que a instituição tem um setor de recursos humanos que precisa ter acesso ao número de dependentes dos membros da instituição para cálculo de remuneração. Para garantir o direito de consentimento dos membros da instituição, o DBA faz uso das construções do ConSQL para inserir os seguintes comandos no SGBD:

```
CREATE PURPOSE "Pesquisas Estatísticas e Aprendizado de Máquina"
CREATE PURPOSE "Calculo de Remuneração"
SET PURPOSE "Calculo de Remuneração" TO COLUMN dependentes ON TABLE Membros
SET PURPOSE "Calculo de Remuneração" TO COLUMN salario ON TABLE Membros
```

Ao fazer isso, serão alocados identificadores para os propósitos “Pesquisas Estatísticas e Aprendizado de Máquina” e “Calculo de Remuneração”, por exemplo, “2” e “5”, respectivamente. Além disso, o valor x do vetor $V(\text{Membros})$ na posição da coluna “dependentes” receberá $x \cup \{5\}$. O mesmo acontece para a coluna “salario”. Cada membro pode então optar por compartilhar seus dados para a pesquisa através do comando:

```
SET PURPOSE "Pesquisas Estatísticas e Aprendizado de Máquina" TO ROWS ON TABLE
Membros AS m WHERE m.cpf = <cpf>
```

Ao inserir esse comando, a tupla em que o valor na coluna “cpf” for igual a <cpf> será incluída no mapa h(Membros) com o valor $\{2\}$.

Somente tuplas com o propósito “Pesquisas Estatísticas e Aprendizado de Máquina” devem estar disponíveis para consultas do setor de pesquisas que visam realizar operações estatísticas sobre os dados de membros da instituição. O processador

de consulta, por meio do processo de geração do plano de execução, vai automaticamente garantir essa restrição na geração do resultado. O detalhamento desse processo está fora do escopo deste trabalho, mas tem como base outras abordagens descritas em [Kraska et al. 2019] e [Pappachan et al. 2020], onde a omissão de tuplas é uma estratégia comum, mas incluem informações adicionais no resultado como a quantidade de tuplas omitidas por não satisfazer as propósitos de acesso.

4. Trabalhos Relacionados

A verificação de propósitos correspondentes a tuplas é uma instância do problema clássico de pertinência de conjuntos, já explorado em outros trabalhos, como [Rizvi et al. 2004] que descreve um modelo de autorização onde consultas são reescritas, ou [Byun and Li 2008] que trata propósitos como entidades hierárquicas, estabelecendo operações para a transição entre estes. Recentemente, o tema voltou ao foco por conta do surgimento de diversas legislações orientadas à preservação da privacidade de indivíduos e a proibição do uso irrestrito de seus dados.

Mais recentemente, [Pappachan et al. 2020] define um middleware para processamento de consultas orientadas a propósito, enquanto que [Wang et al. 2019] descreve um mecanismo para acesso a dados com propósito, com garantias sobre os dados manipulados. [Shastri et al. 2020] especifica um conjunto de propriedades para atender a legislação, executando um experimento que justifica uma nova evolução nos SGBDs, assim como [Machado and Amora 2020] elabora sobre os impactos causados pela legislação, além de fazer uma revisão da literatura e apontar oportunidades de pesquisa. Estes trabalhos tem em comum o argumento da necessidade de extensão dos SGBD como parte ativa da gestão dos acessos e permissões, assim como a discussão sobre impactos trazidos por essas extensões, sejam elas acopladas ou integradas ao SGBD.

[Agrawal et al. 2005] e [Pun 2010] atacam diretamente o problema da extensibilidade do SQL para a definição de propósitos de formas distintas. O trabalho [Agrawal et al. 2005] define um modelo de restrições semelhante a permissões, associando o acesso a um conjunto de permissões e reescrevendo a consulta de forma a manter o estado do banco em relação às restrições. [Pun 2010] altera as cláusulas padrão SQL para adicionar funções de generalização e validade aos dados, enquanto outras cláusulas passam a exigir de forma explícita o propósito de acesso.

5. Conclusão

O presente trabalho apresentou o ConSQL, uma extensão da gramática da SQL que viabiliza o suporte ao conceito de consentimento por meio de propósitos trazido pelas novas normas de proteção e privacidade de dados. Descrevemos a gramática, os novos comandos adicionados à linguagem e um exemplo de uso. Já em andamento, mas ainda com resultados parciais, estamos investigando (1) a definição formal da semântica das novas construções gramaticais introduzidas pelo ConSQL e (2) novos métodos eficientes de Reforço de Propósito em tempo de execução de consulta. Consolidados os resultados, tais investigações serão objetos de relatos futuros de investigação.

Agradecimentos

Esta pesquisa foi parcialmente apoiada pela CAPES (processo #88887.609129/2021) e LSBD/UFC.

Referências

Agrawal, R., Bird, P., Grandison, T., Kiernan, J., Logan, S., and Rjaibi, W. (2005). Extending relational database systems to automatically enforce privacy policies. In *ICDE*, pages 1013–1022. IEEE Computer Society.

Brasil (2018). Lei nº 13.709 - lei geral de proteção de dados pessoais (lgpd). `http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm`. Acessado em: 01-07-2021.

Byun, J. and Li, N. (2008). Purpose based access control for privacy protection in relational database systems. *VLDB J.*, 17(4):603–619.

California (2018). California consumer privacy act (ccpa). `https://www.caprivacy.org/`. Acessado em: 01-07-2021.

Elmasri, R. and Navathe, S. B. (2000). *Fundamentals of Database Systems, 3rd Edition*. Addison-Wesley-Longman.

General Data Protection Regulation (2016). Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46. *Official Journal of the European Union*, 59:1–88.

Kraska, T., Stonebraker, M., Brodie, M. L., Servan-Schreiber, S., and Weitzner, D. J. (2019). SchengenDB: A data protection database proposal. In *Heterogeneous Data Management, Polystores, and Analytics for Healthcare - VLDB 2019 Workshops, Poly and DMAH, Los Angeles, CA, USA, August 30, 2019*, volume 11721 of *Lecture Notes in Computer Science*, pages 24–38. Springer.

Machado, J. C. and Amora, P. R. P. (2020). How can db systems be ready for privacy regulations. In *SBBD*. SBC.

Pappachan, P., Yus, R., Mehrotra, S., and Freytag, J. (2020). Sieve: A middleware approach to scalable access control for database management systems. *Proc. VLDB Endow.*, 13(11):2424–2437.

Pun, S. (2010). Prisql: a privacy preserving sql language.

Rizvi, S., Mendelzon, A. O., Sudarshan, S., and Roy, P. (2004). Extending query rewriting techniques for fine-grained access control. In *SIGMOD Conference*, pages 551–562. ACM.

Shastri, S., Banakar, V., Wasserman, M., Kumar, A., and Chidambaram, V. (2020). Understanding and benchmarking the impact of GDPR on database systems. *Proc. VLDB Endow.*, 13(7):1064–1077.

Solove, D. J. and Citron, D. K. (2017). Risk and anxiety: A theory of data-breach harms. *Tex. L. Rev.*, 96:737.

Wang, L., Near, J. P., Somani, N., Gao, P., Low, A., Dao, D., and Song, D. (2019). Data capsule: A new paradigm for automatic compliance with data privacy regulations. *CoRR*, abs/1909.00077.