UNIVERSIDADE DE BRASÍLIA
Faculdade do Gama

Sistemas de Banco de Dados 2

Tecnologias de Banco de Dados (TI-BD)
Bancos de Dados Dedutivos

Nicolas Chagas Souza - 200042327

Brasília, DF
2023

Sumário

Bancos de Dados Dedutivos

Definição

Um Banco de Dados Dedutivo é um sistema de bancos de dados que fornece a possibilidade de definir **regras dedutivas**, utilizadas para inferir ou deduzir informações a partir dos **fatos** armazenados no Banco de Dados. Essa área consiste na interseção entre áreas de bancos de dados, programação lógica e inteligência artificial, e por esse motivo, possui uma fundamentação teórica na lógica matemática, e utiliza a linguagem **Prolog** como ponto de partida.

Os Sistemas de Bancos de Dados Dedutivos (DDB, *deductive databases*) utilizam linguagens declarativas, linguagens que especificam o que conseguir, mas não determinam como. A linguagem declarativa **Datalog**, uma variação da linguagem Prolog**, é utilizada para definir regras com um conjunto de relações existentes, tratadas como literais na linguagem.

Os DDBs possuem duas categorias de especificações, **regras** e **fatos**. Os fatos possuem especificações semelhantes às relações, mas sem a necessidade de nomear os atributos, já que o significado de um valor em uma tupla está atrelado à sua posição nela. Já as regras, se assemelham às visões dos bancos relacionais, especificam relações virtuais, que não estão realmente armazenadas, formadas com base nos fatos, utilizando mecanismos de inferência. Podem envolver recursão. Podemos relacionar os fatos às tuplas em um banco de dados, e as regras às consultas (*queries*).

Objetivos

O principal objetivo dos Bancos de Dados Dedutivos é fornecer um framework, por meio da especificação de regras, para inferir, ou deduzir, novas informações a partir das regras. Tendo em vista o grande volume de dados armazenados atualmente, é possível perceber uma grande vantagem proposta por essa tecnologia. Os DDBs auxiliam na combinação entre Bancos de Dados Relacionais e programação lógica, e a criação de um banco de dados dedutivo utiliza uma linguagem de programação declarativa chamada Datalog.

Notação Prolog

A notação Prolog fornece **predicados** com nomes exclusivos. Um predicado possui um significado implícito, sugerido pelo nome, e um númnero fixo de argumentos. Se todos os argumentos forem constantes, o predicado indica que o fato é verdadeiro. Um predicado com variáveis é considerado uma **consulta**, parte de uma regra ou restrição. Já um predicado com argumento

constantes é chamado **predicado base** ou **predicado instanciado**.

Adotam-se algumas convenções, de modo a facilitar a leitura dos predicados, a saber:

- Todos os valores constantes em um predicado são strings numéricas ou de caractere, representados com identificadores inicando por letra minúscula.
- Nomes de variáveis iniciam com letra maiúscula.

O modelo utilizado para bancos de dados dedutivos assemelha-se ao modelo relacional, e para exemplificar tal relação considere a necessidade de armazenar a hierarquia de supervisores de uma empresa (Figura 1).

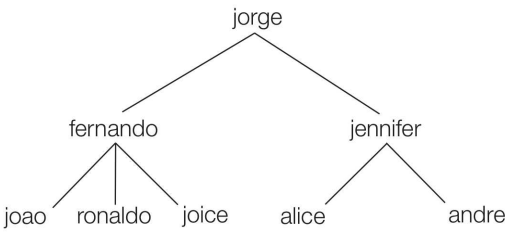

Figura 1: Hierarquia de supervisores. Fonte: [1]

A partir dessa situação, pode-se determinar **fatos**, como os representados no Trecho de Código 1.

```
SUPERVISIONA(jorge, fernando).
SUPERVISIONA(jorge, jennifer).
SUPERVISIONA(fernando, joao).
SUPERVISIONA(jennifer, alice).
SUPERVISIONA(jennifer, andre).
```

Trecho de Código 1: Exemplo dos fatos em Prolog.

Os fatos assemelham-se às relações tuplas no modelo relacional. A hierarquia (Figura 1) pode ser presentada pelas tuplas das relações criadas no Trecho de Código 2.

```
CREATE TABLE FUNCIONARIO (
    id    INT           NOT NULL AUTO_INCREMENT PRIMARY KEY ,
    nome VARCHAR(100) NOT NULL
) ENGINE = InnoDB AUTO_INCREMENT = 1;
CREATE TABLE supervisiona (
    idSuperior      INT NOT NULL,
    idSubordinado INT NOT NULL,
```

```
    CONSTRAINT supervisiona_superior_fk FOREIGN KEY (idSuperior)
        REFERENCES FUNCIONARIO (id),
    CONSTRAINT supervisiona_subordinado_fk FOREIGN KEY (idSubordinado)
        REFERENCES FUNCIONARIO (id));
INSERT INTO FUNCIONARIO (nome) VALUES ('jorge'), ('fernando'), ('joao'),
                ('ronaldo'), ('joice'), ('jennifer'), ('alice'), ('andre');
INSERT INTO supervisiona (idSuperior, idSubordinado)
VALUES (1, 2), (1, 6), (2, 3), (2, 4), (2, 5), (6, 7), (6, 8);
```

Trecho de Código 2: Criação das relações e inserção dos fatos no modelo relacional.

As **regras** são utilizadas pelo **mecanismo de inferência** para derivar novas informações. Conforme o diagrama (Figura 1), pode-se afirmar que Jorge é superior de João, pois ele supervisiona Fernando, que supervisiona João. Além disso, pode-se afirmar que Alice é subordinada de Jennifer. Essas informações podem ser traduzidas em regras**, na notação prolog, conforme ilustrado no Trecho de Código 3.

```
SUPERIOR(X,Y) :- SUPERVISIONA(X,Y).
SUPERIOR(X,Y) :- SUPERVISIONA(X,Z), SUPERVISIONA(Z,Y).
SUBORDINADO(X,Y) :- SUPERIOR(Y,X).
```

Trecho de Código 3: Criação das regras em Prolog.

Uma regra possui a sintaxe `cabeça :- corpo`, o símbolo `:-` é interpretado como "se e somente se". A cabeça também é chamada conclusão ou LHS (Left-hand side) e é geralmente formada apenas por um predicado. O corpo também pode ser chamado premissa ou RHS (right-hand side), e pode ser formado por um ou mais predicados separados por vírgulas, que indicam a operação lógica `AND`.

Portanto, para uma regra cabeca :- predicado$_1(X_1)$, ..., predicado$_n(X_n)$, tem-se predicado$_i(X_i)$ = VERDADEIRO $\forall\ 0 \leq i \leq n \Rightarrow$ cabeca = VERDADEIRO. Além disso, se houver mais do que uma regra com a mesma cabeça, o predicado será verdadeiro se ao menos um dos corpos for verdadeiro, i. e., `X:-Y` e `X:-Z` é equivalente a `X:-Y OR Z`.

Portanto, o segundo predicado, `SUPERIOR(X,Y) :- SUPERVISIONA(X,Z), SUPERVISIONA(Z,Y).`, indica que `X` supervisiona `Y` se, e somente se, `X` supervisionar um funcionário `Z` **e** `Z` supervisionar `Y`.

No modelo relacional, é possível consultar as supervisões diretas utilizando `VIEW`s, como a criada

no Trecho de Código 4, mas não é possível definir a regra de recursividade entre superiores.

```
CREATE VIEW v_Superior AS
    SELECT sup.nome Superior, sub.nome Subordinado
        FROM supervisiona s
            JOIN FUNCIONARIO sub ON s.idSubordinado = sub.id
            JOIN FUNCIONARIO sup ON s.idSuperior = sup.id;
```

Trecho de Código 4: Criação das views em SQL.

Por fim, é possível fazer consultas, sobre os dados armazenados e inferidos, representadas em prolog no Trecho de Código 5.

```
SUPERIOR(jorge,Y)?
SUPERIOR(jorge,joice)?
```

Trecho de Código 5: Consultas em Prolog.

Como o primeiro predicado possui uma variável, `Y`, trata-se de uma consulta, e a posição dos argumentos é relevante, a consulta representada determina todos os `Y` tais que `SUPERIOR(jorge, Y)` seja uma afirmação verdadeira, portanto o resultado dessa consulta será o conjunto $\{fernando, joao, ronaldo, joice\}$[1], conforme a hieraquia definida anteriormente (Figura 1). A segunda consulta possui dois argumentos constantes, então sua resposta será `VERDADEIRO` ou `FALSO`, como `SUPERIOR(jorge,fernando),SUPERIOR(fernando,joice)`, temos que `SUPERIOR(jorge,joice)` é `VERDADEIRO`.

Utilizando a `VIEW` criada no Trecho de Código 4, é possível efetuar a primeira consulta, representada no Trecho de Código 6, mas a segunda em SQL retornaria um conjunto vazio de resultados, embora Jorge seja superior de Joice, conforme demonstrado anteriormente.

```
# Consulta 1
SELECT Subordinado FROM v_Superior
                    WHERE Superior = 'jorge';
# Consulta 2
SELECT * FROM v_Superior
            WHERE Superior = 'jorge' AND Subordinado = 'joice';
```

Trecho de Código 6: Consultas sobre as views em SQL.

[1]Os mecanismos de inferência em Prolog geralmente retornam um valor, por vez, para a consulta, ao contrário do Datalog, que retorna um conjunto de resultados por vez.

A primeira consulta retorna o conjunto de subordinados {fernando, jennifer}, e a segunda, o conjunto vazio.

Notação Datalog

Um programa em Datalog é composto pela combinação de fórmulas atômicas, que são literais na forma $p(a_1, a_2, ..., a_n)$, onde p o nome do predicado, e n o número de argumentos, também chamado aridez ou grau de p. Estão disponíveis, em Datalog, os predicados embutidos de:

- **comparação binária**, em domínios ordenados: less ($<$), less_or_equal ($\leq$), greater ($>$) e greater_or_equal ($\geq$);
- **comparação**, em domínios ordenados ou não ordenados: equal ($=$) e not_equal ($\neq$).

Um literal pode ser positivo ou negativo, a segunda classificação se dá às fórmulas atômicas precedidas por **not**. Programas em Datalog são subconjuntos das fórmulas de cálculo de predicado, semelhantes às fórmulas do cálculo relacional de domínio, e toda expressão de álgebra relacional básica pode ser escrita em Datalog, mas para serem expressas em Datalog, essas fórmulas são convertidas para a forma clausular de Horn. Além disso, o Datalog pode expressar recursões, não representáveis na álgebra relacional.

Forma clausular

A forma clausular de uma fórmula ocorre quando:

- Todas as variáveis na fórmula são quantificadas universalmente[2]
- A fórmula é composta por uma série de cláusulas, e cada cláusula é uma disjunção de literais, i. e., os literais são conectados apenas pelo operador lógico `OR`.
- A fórmula é uma conjunção de cláusulas, i. e., as cláusulas são conectadas apenas pelo operador `AND`.

Ademais, para que uma fórmula esteja na forma clausular de Horn, ela deve estar na forma clausular e possuir no máximo um literal positivo.

Dados $P_i, Q, 1 \leq i \leq n$ predicados quantificados universalmente, uma cláusula de Horn pode ocorrer apenas em uma das seguintes formas:

$$\neg P_1 \vee \neg P_2 \vee ... \vee \neg P_n \vee Q \tag{1}$$

$$\neg P_1 \vee \neg P_2 \vee ... \vee \neg P_n \tag{2}$$

[2]O quantificador universal indica ”para todos ($\forall$)”, e o existencial indica ”existência ($\exists$)”.

Sabendo que a implicação $P \Rightarrow Q$ é logicamente equivalente à $\neg P \vee Q$, pode-se reescrever a primeira forma (1) como:

$$(\neg P_1 \vee \neg P_2 ... \vee \neg P_n) \vee Q \tag{3}$$

$$\neg(P_1 \wedge P_2 \wedge ... \wedge P_n) \vee Q \tag{4}$$

$$P_1 \wedge P_2 \wedge ... \wedge P_n \Rightarrow Q \tag{5}$$

E pela mesma justificativa, a segunda forma (2) torna-se:

$$P_1 \wedge P_2 \wedge ... \wedge P_n \Rightarrow \tag{6}$$

As cláusulas 5 e 6 podem ser representadas pelas seguintes expressões Datalog, respectivamente $Q :- P_1, P_2, ..., P_n$ e $P_1, P_2, ..., P_n$. A primeira expressão é uma regra e a segunda pode ser considerada uma restrição de integridade, que verifica se todos os predicados são verdadeiros.

Um programa em Prolog ou Datalog possui um mecanismo de inferência, que pode ser usado para processar e calcular os resultados das consultas.

Interpretação de Regras

O significado teórico das regras pode ser interpretado por duas abordagens: teórica de prova e teórica de modelo, mas, na prática, o mecanismo de inferência define a interpretação exata, que pode ou não coincidir com as interpretações teóricas.

A interpretação teórica de prova visualiza fatos e regras como afirmações verdadeiras, ou axiomas. Esses axiomas podem ser axiomas de base, quando não possuem variáveis, ou axiomas dedutivos. Os axiomas dedutivos são utilizados para derivar novos fatos (teoremas).

A interpretação teórica pode ser utilizada para provar que Alice é subordinada de Jorge, conforme representação na Figura 1, a partir das regras definidas no Trecho de Código 3 e dos fatos definidos no Trecho de Código 1.

1. SUBORDINADO(X,Y) :- SUPERIOR(Y,X). (regra 1)
2. SUPERIOR(X,Y) :- SUPERVISONA(X,Z),SUPERVISONA(Z,Y). (regra 2)
3. SUPERVISIONA(jorge, jennifer). (fato 1)
4. SUPERVISIONA(jennifer, alice). (fato 2)

5. SUPERIOR(jorge, alice). (Aplicando a regra 2 sobre os fatos 1 e 2)

6. SUBORDINADO(alice, jorge (Aplicando a regra 1 sobre 5)

A interpretação teórica de modelo parte da especificação de um domínio, geralmente finito, de valores constantes, testa um predicado com todas as combinações possíveis de argumentos e determina se esse conjunto de valores o torna positivo ou negativo. Na prática, uma vez determinado um conjunto de argumentos que torna o predicado verdadeiro, após testar todas as combinações disponiveis, assume-se que os demais conjuntos o tornam falso.

Os valores verdade de um predicado podem ser definidos de duas maneiras, por meio dos fatos (ou relações) ou por regras (ou visões), os primeiros decorrem dos valores armazenados nos bancos de dados, e os segundos são definidos por serem a cabeça (LHS) de uma ou mais regras Datalog. Um programa é definido como **seguro** quando gera uma quantidade finita de fatos, essa segurança pode ser violada, por exemplo, quando uma variável de uma regra assume valores em um domínio infinito de valores. Portanto, programas seguros possuem apenas variáveis limitadas em suas regras.

Ferramentas e Aplicações

O Datalog pode ser utilizado para análise de dados, como proposto pela LogicBlox, por meio do LogiQL, uma combinação entre linguagens de consulta em Banco de Dados e linguagens de programação funcional. Além disso, a implementação de Bancos de Dados dedutivos pode ser vista, por exemplo, em Datalog, LDL (*Logic Data Language*), NAIL! (*Not AnotherImplementation of Logic!*) e CORAL. A seguir, algumas das ferramentas utilizadas no passado e atualmente.

Datalog: o pacote Datalog contem um Sistema de Banco de Dados Dedutivo, onde as operações de queries e atualização do banco de dados são realizadas com a sintaxe Datalog. Também é possível utilizar o módulo de Datalog da linguagem de programação Racket.

CORAL: desenvolvido pela universidade de *Wisconsin*, CORAL é um sistema dedutivo com suporte à linguagem declarativa e provém uma 'interface' com C++. A linguagem de queries dá suporte às cláusulas de Horn, além de fornecer ferramentas para agrupamento, agregação, negação e relações com tuplas com variáveis universalmente quantificadas.

Datomic: o Banco de Dados Datomic utiliza um sistema dedutivo de queries, em uma forma

estendida do Datalog

Vantagens e Desvantangens

Os bancos de dados dedutivos apresentam vantagens em relação aos bancos relacionais, como a inferência automatizada, disponibilidade de ferramentas para manutenção da integridade e facilidade na escrita de consultas complexas.

Inferência automatizada: por meio dos bancos de dados dedutivos é possível deduzir novos dados a partir do conjunto de dados já existente, o que nem sempre é possível utilizando sistemas relacionais, como exemplificado na definição das regras de hierarquia entre os supervisores, representada na Figura 1, no Trecho de Código 3 e a impossibilidade de representar a recursividade em SQL. Além disso, nesse contexto as consultas Datalog trazem resultados mais completos, como exemplificado nos Trechos de Código 5 e 6.

Ferramentas para manutenção da integridade: as consultas Datalog podem ser utilizadas como restrições de integridade, conforme explicado na seção Forma Clausular, mais especificamente por meio da forma clausular positiva (eq. 5). Essa categoria de consulta pode ser utilizada para manter a integridade dos dados durante atualizações, por exemplo, o que poderia ser feito em bancos relacionais com auxílio de funções, aumentando, portanto, a complexidade de implementação.

Facilidade na escrita de consultas complexas: a escrita de consultas Datalog possui uma sintaxe com menor complexidade de compreensão, em relação às suas equivalentes em bancos relacionais, um exemplo dessa redução de complexidade pôde ser observado durante a representação de uma consulta Datalog (Trecho de Código 3) na forma de view, no Trecho de Código 4.

Também observam-se vantagens decorrentes das supracitadas, como a maior flexibilidade na modelagem de dados, a integração entre conhecimentos interdisciplinares — nas áreas de lógica, matemática, programação e bancos de dados — e a atualização dinâmica dos dados.

Apesar das vantagens apresentadas pela tecnologia, alguns fatores dificultam a sua implementação e popularização. O principal fator é a **quantidade reduzida de ferramentas e recursos** destinados aos bancos de dados dedutivos, tal escassez torna-se nítida ao comparar a quantidade de SGBRs[3] disponíveis gratuitamente, como, por exemplo, as ferramentas MySQL, PostgreSQL e MariaDB. A escrita de regras relevantes exige um bom domínio de lógica, devido ao alto ápice da **curva de aprendizagem** da tecnologia, **aumentando a complexidade**

[3]Sistemas Gerenciadores de Bancos de Dados Relacionais

de implementação em relação aos Bancos Relacionais, **elevando o custo de manuteção de profissionais** especializados nessa área. Por outro lado, a modelagem e manipulação de dados em bancos relacionais possui uma curva de aprendizagem mais favorável, diminuindo a complexidade de implementação e o custo de manutenção de mão obra.

Portanto, deve-se avaliar a situação por diferentes pontos de vista antes da implementação de bancos de dados dedutivos, se não houver necessidade de realizar inferências e derivar novos fatos por regras, não há necessidade de utilizar essa tecnologia.

Referência Bibliográfica

[1] ELMASRI, Ramez; NAVATHE, Shamkant. Sistemas de banco de dados. 6.ª edição. São Paulo: Pearson, 2011.

[2] NARDON, Fabiane. Estudo e Construção de um Sistema Gerenciador de Banco de Dados Dedutivo. 1996, 142 p. Dissertação (mestrado) — Curso de Pós-Graduação em Ciência da Computação — Universidade Federal do Rio Grande do Sul, Porto Alegre, 1996.

[3] BRISCHKE, Marcelo; MARCONDES, Eduardo. Banco de Dados Dedutivos. 2005, 31 p. Monografia (graduação) — Curso de Bacharelado em Informática — Universidade Estadual do Oeste do Paraná, Cascavel, 2005.

[4] KOTSIREAS, Ilias. Deductive Databases (DDBs). 2009. Apresentação em PDF. Disponível em https://web.wlu.ca/science/physcomp/ikotsireas/CP465/W10-DDBs/DeductiveDatabases.pdf. Acesso em: 17 de abril de 2023.

[5] BALAZINSKA, Magda. Lecture 10: Datalog. 2012. Apresentação em PDF. Disponível em https://courses.cs.washington.edu/courses/cse344/12au/lectures/lecture10-datalog.pdf. Acesso em: 17 de abril de 2023.

[6] Applications of Commercial Deductive Database Systems, **Geeksforgeeks**, 2023. Disponível em https://www.geeksforgeeks.org/applications-of-commercial-deductive-database-systems. Acesso em: 17 de abril de 2023.

[7] LogicBlox Platform Technology, **LogicBlox**, 2023. Disponível em https://developer.logicblox.com/technology/. Acesso em: 17 de abril de 2023.

[8] Datalog: Deductive Database Programming, **Racket**. Disponível em https://docs.racket-lang.org/datalog/index.html. Acesso em: 17 de abril de 2023.

[9] **CORAL DATABASE PROJECT**, **Coral Database Project**, c2001. Página inicial. Disponível em https://research.cs.wisc.edu/coral/. Acesso em 17 de abril 2023.

[10] **DATALOG USER MANUAL**, Datalog User Manual. c2004. Página inicial. Disponível em https://datalog.sourceforge.net/datalog.html. Acesso em: 17 de abril de 2023.

[11] **DATOMIC CLOUD DOCUMENTATION**, Datomic Cloud Documentation. c2023. Página inicial. Disponível em https://docs.datomic.com/cloud/. Acesso em: 17 de abril de 2023.