

Oracle

Saiba como utilizar o Data Guard com Fast-Start Failover

Modelagem

Modelagem dados de
maneira prática e intuitiva

SQL Server

Trabalhando com o Service Broker



ADMINISTRAÇÃO DO POSTGRESQL

Políticas de
backup e restore

Replicação na
nuvem da Amazon

Mineração de Dados
Comparação de algoritmos
de mineração de texto

SQL Server
Implemente uma solução
de Cluster Shared Volume



MVP

R\$ 1.000.000,00
INVESTIDOS EM CONTEÚDO
NOS ÚLTIMOS 12 MESES.

APlique esse investimento
na sua carreira...

E mostre ao mercado
quanto você vale!

CONFIRA TODO O MATERIAL
QUE VOCÊ TERÁ ACESSO:

- + de **9.000** video-aulas
- + de **290** cursos online
- + de **13.000** artigos
- DEVMEDIA API's consumido + de **500.000** vezes



POR APENAS
R\$ 69,90* mensais

*Tempo mínimo de assinatura: 12 meses.



PRA QUEM QUER EXIGIR
MAIS DO MERCADO!

 **DEV**MEDIA

Sumário

Conteúdo sobre Boas Práticas

06 – Passo a passo para realizar a modelagem de dados

[Rodrigo Ramos Nogueira]

Conteúdo sobre Boas Práticas

13 – Cluster Shared Volume no SQL Server 2014

[Leandro Romualdo da Silva]

Conteúdo sobre Boas Práticas

18 – Service Broker no SQL Server 2014

[Dhiego Piroto]

Conteúdo sobre Boas Práticas

24 – Estratégias de backup e restore no PostgreSQL

[Thiago Lima de Castro]

Conteúdo sobre Boas Práticas

29 – Replicando DB relacional na nuvem da Amazon

[Antonio Marcos Ferreira]

Conteúdo sobre Boas Práticas

40 – Mineração de texto: Análise comparativa de algoritmos

[Breno Santana Santos e Methanias Colaço Júnior]

Conteúdo sobre Boas Práticas

55 – Oracle Data Guard com Fast-Start Failover

[Ivan Ricardo Schuster]



Dê seu feedback sobre esta edição!

A SQL Magazine tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre esta edição, artigo por artigo, através do link:

www.devmedia.com.br/sqlmagazine/feedback

EXPEDIENTE

Editor

Rodrigo Oliveira Spínola (rodrigo.devmedia@gmail.com)

Subeditor

Eduardo Oliveira Spínola

Consultor Técnico

Joel Neto (joelrlneto@gmail.com)

Jornalista Responsável

Kaline Dolabella - JP24185

Capa e Diagramação

Romulo Araújo

Distribuição

FC Comercial e Distribuidora S.A
Rua Teodoro da Silva, 907
Grajaú - RJ - 206563-900

Atendimento ao leitor

A DevMedia possui uma Central de Atendimento on-line, onde você pode tirar suas dúvidas sobre serviços, enviar críticas e sugestões e falar com um de nossos atendentes. Através da nossa central também é possível alterar dados cadastrais, consultar o status de assinaturas e conferir a data de envio de suas revistas. Acesse www.devmedia.com.br/central, ou se preferir entre em contato conosco através do telefone 21 3382-5038.

Publicidade

publicidade@devmedia.com.br – 21 3382-5038

Anúncios – Anunciando nas publicações e nos sites do Grupo DevMedia, você divulga sua marca ou produto para mais de 100 mil desenvolvedores de todo o Brasil, em mais de 200 cidades. Solicite nossos Media Kits, com detalhes sobre preços e formatos de anúncios.

Fale com o Editor!

É muito importante para a equipe saber o que você está achando da revista: que tipo de artigo você gostaria de ler, que artigo você mais gostou e qual artigo você menos gostou. Fique à vontade para entrar em contato com os editores e dar a sua sugestão!

Se você estiver interessado em publicar um artigo na revista ou no site SQL Magazine, entre em contato com os editores, informando o título e mini-resumo do tema que você gostaria de publicar:

Rodrigo Oliveira Spínola - Editor da Revista
rodrigo.devmedia@gmail.com



RODRIGO OLIVEIRA SPÍNOLA

Editor Chefe da SQL Magazine, Mobile e Engenharia de Software Magazine. Professor da Faculdade Ruy Barbosa, uma instituição parte do Grupo DeVry. Doutor e Mestre em Engenharia de Software pela COPPE/UFRJ.

FÓRUM DEVMEDIA

O lugar perfeito para você ficar por dentro de tudo o que acontece nas tecnologias do mercado atual



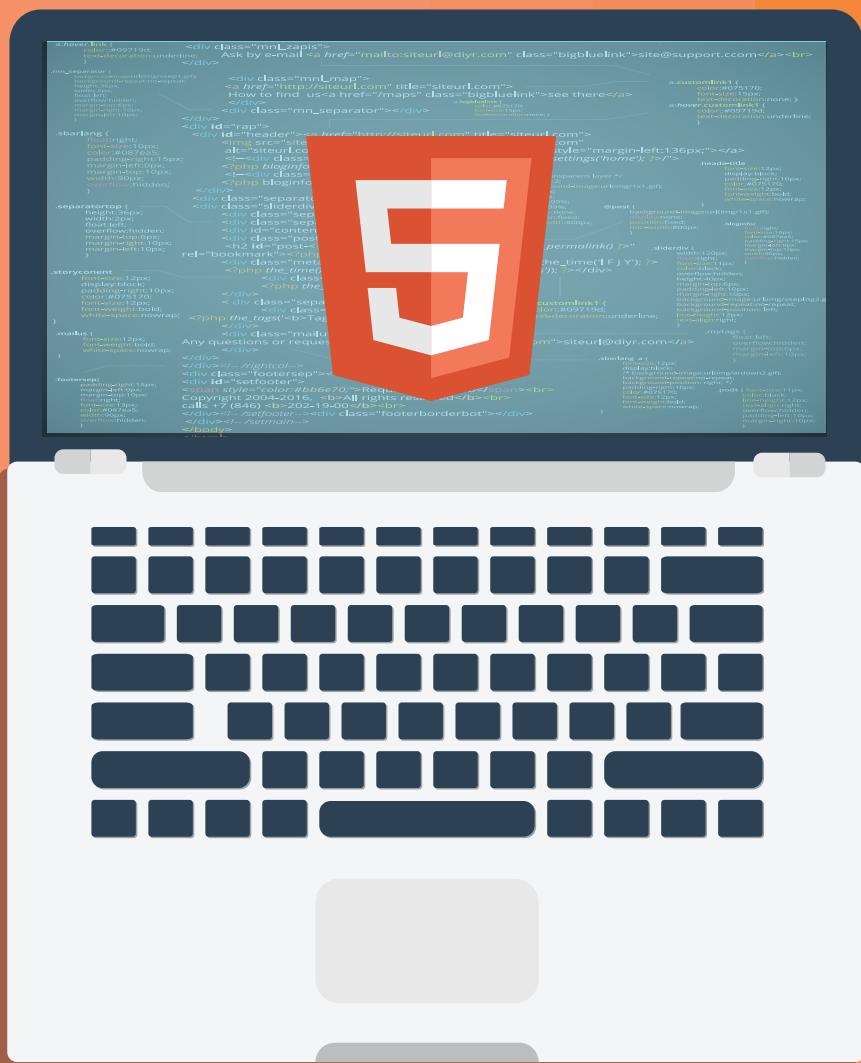
No fórum da DevMedia você irá encontrar uma equipe disponível e altamente qualificada com consultores e colaboradores prontos para te ajudar a qualquer hora e sobre qualquer assunto.

Temos as salas de Java, .NET, Delphi, Banco de Dados, Engenharia de Software, PHP, Java Script, Web Design, Automação comercial, Ruby on Rails e muito mais!

ACESSE AGORA
www.devmedia.com.br/forum

Guia HTML 5

Um verdadeiro manual de referência com tudo que você precisa sobre HTML!



DEVMEDIA

<http://www.devmedia.com.br/guias/guia-html/3>

Passo a passo para realizar a modelagem de dados

Aprenda a criar um modelo de dados com qualidade

Com o crescente aumento da quantidade de dados gerados em todo o mundo, que apenas no fim do ano de 2014 atingiram o volume de 2,7 *zettabytes*, pode-se dizer que esta é a era do Big Data, havendo o surgimento de novas técnicas e tecnologias de armazenamento, em uma época onde se fala muito em HPC (*High Performance Computing* – computação de alto desempenho), Hadoop e NoSQL.

Mesmo com novas tecnologias ascendentes, os Sistemas Gerenciadores de Bancos de Dados Relacionais ainda são maioria no mercado, sendo utilizados desde o desenvolvimento de pequenos e-commerces a grandes ERPs. Os bancos de dados relacionais podem ter sido criados há décadas, porém, sempre estiveram em constante atualização para se adaptar ao cenário do momento, e na era do Big Data incluem adaptações para armazenar e processar dados em larga escala.

Atualmente são diversas as ferramentas para modelagem de dados disponíveis no mercado, trazendo uma abordagem que envolve as principais características esperadas em um software desse tipo. Este artigo visa explorar as principais ferramentas, permitindo ao leitor decidir qual a melhor opção de uso, bem como demonstrar na prática como realizar a modelagem de dados passo a passo.

O modelo relacional tem como principal objetivo a integridade dos dados, característica fundamental e que tem relação direta com a continuidade de sua adoção até os dias atuais. Garantir a integridade dos dados significa que após o seu armazenamento, eles poderão ser recuperados no mesmo estado, sem sofrer alterações não autorizadas. E um dos responsáveis por esta melhoria é a normalização, assunto que será abordado na sequência.

Normalização dados

A normalização dos dados é o primeiro passo para se obter sucesso com um modelo de dados íntegro, uma

Fique por dentro

Ainda hoje, passados mais de 45 anos da criação dos bancos de dados relacionais e também da disseminação das soluções NoSQL, os bancos de dados relacionais ainda são muito utilizados por conta de seus diversos recursos para manipulação de dados, bem como sua eficiência e eficácia quando se trata da integridade dos dados. Este artigo será útil para todo desenvolvedor ou DBA que esteja no início de um projeto e deseja construir um modelo de dados que permita o armazenamento de informações com qualidade. Para isso, serão abordadas aqui as principais ferramentas de modelagem de dados, juntamente com conceitos fundamentais necessários para a modelagem relacional e sua aplicação na prática.

vez que sendo estas normas respeitadas, as redundâncias e inconsistências poderão ser evitadas.

Uma característica fundamental nos bancos de dados relacionais são as transações, operações que manipulam os dados no banco e que podem ser comportadas por inserções, buscas, atualizações ou exclusões. Espera-se que ao realizar transações sejam atendidas as propriedades ACID, descritas a seguir:

- **Atomicidade:** esta propriedade garante que uma transação seja atômica, ou seja, indivisível. De modo geral, ou tudo acontece ou nada acontece. Isto significa que em uma transação composta por várias operações, se por algum motivo alguma operação não for realizada, todas as demais também não serão e caso alguma tenha sido realizada anteriormente, deverá ser anulada;
- **Consistência:** esta propriedade garante que um banco de dados consistente deverá continuar consistente após uma transação. Um exemplo para melhor entendimento é o caso de uma transação que deve excluir 1.000 registros na base de dados e falha durante a exclusão do registro de número 600. Haverá essa mesma quantidade de registros a serem excluídos e 400 não excluídos temporariamente. Para manter a consistência em casos como este, a transação deverá deixar o banco de dados no estado em que ele estava antes da transação iniciar, ou seja, sem excluir os 600 registros;

- **Isolamento:** esta propriedade garante que cada transação dentro de um banco de dados ocorra de maneira isolada, ou seja, mesmo que haja várias transações acontecendo ao mesmo tempo, as operações de uma transação não deverão interferir nas demais;
- **Durabilidade:** esta propriedade está relacionada aos dados e sinaliza que um dado deve continuar imutável até que uma nova operação seja realizada. Para isso, caso uma operação seja realizada com sucesso, os dados modificados por ela devem ser persistidos no banco.

As propriedades ACID não estão ligadas diretamente à modelagem, porém ter conhecimento de sua existência e de suas definições é de extrema importância para quem constrói um modelo de banco de dados.

Para obter um modelo de dados dito normalizado, uma sequência de etapas de validação deve ser seguida, denominadas formas normais. As formas normais são regras para construção de tabelas e relações e será respeitando-as que o modelo normalizado será obtido. As formas normais são definidas a seguir:

- **Primeira Forma Normal:** A primeira forma normal trata da atomicidade dos atributos, proibindo atributos compostos, multivalorados e relações aninhadas. Para que uma tabela esteja na primeira forma normal é preciso obedecer aos seguintes passos:

- Identificar a chave primária da tabela;
- Procurar o grupo que está se repetindo e retirá-lo da tabela;
- Criar uma nova tabela com os atributos repetidos, onde a chave primária desta tabela será chave estrangeira na tabela anterior;
- Repetir este processo até que todas as tabelas geradas estejam na primeira forma normal.

- **Segunda Forma Normal:** A segunda forma normal está relacionada à dependência funcional da chave primária. Para estar na segunda forma normal a tabela deve estar na primeira forma normal e nenhum dos campos que não são chaves não podem depender de apenas parte da chave primária. Para remover a dependência funcional os passos a seguir devem ser respeitados:

- Identificar campos que são dependentes de apenas parte da chave primária e não da chave primária como um todo;
- Remover da tabela todos os campos que não são dependentes funcionais.

- **Terceira Forma Normal:** a terceira forma normal está relacionada à chamada dependência transitiva, ou seja, um campo não deve depender de um outro campo “não-chave”. Para remover a dependência transitiva se deve identificar os campos que são dependentes transitivos de outros campos e removê-los.

Visto a importância das formas normais para o desenvolvimento de um banco de dados relacional, bem como a dificuldade no seu entendimento, na parte prática deste artigo as formas normais serão aplicadas junto a um exemplo real.

Entidades e seus atributos

Uma entidade em um modelo de dados é a representação de um objeto do mundo real. Para distinguir uma entidade das demais, além do nome, elas são compostas de diferentes características, às quais damos o nome de atributos. Por exemplo, a entidade *carro* pode ter os atributos *ano*, *modelo* e *cor*. Ademais, é válido ressaltar que os atributos podem ser classificados de diferentes maneiras, a saber:

- **Compostos:** os atributos compostos são aqueles que podem ser divididos em partes menores como um atributo *endereço*, que pode ser dividido em *rua*, *bairro* e *número*;
- **Simples:** os atributos simples são aqueles que não podem mais ser divididos. Estes também são chamados de atômicos;
- **Monovalorados:** são atributos que armazenam apenas um valor para determinada entidade como a tabela *pessoa* só pode ter um valor para *idade*;
- **Multivalorado:** são atributos que podem armazenar vários valores para a entidade como um atributo *telefone*, que permite armazenar todos os telefones da entidade *pessoa*;
- **Derivado:** são atributos que podem derivar uns de outros como idade e data de nascimento.

Um atributo também pode ser definido como identificador, ou chave-primária, sendo então responsável por representar de maneira única todos os demais atributos em uma determinada entidade. Um conceito relacionado que é encontrado na literatura é o de chave-candidata. Esta sinaliza um atributo único que pode identificar a entidade, e por isso é dito que ele é uma chave-candidata a se tornar chave primária.

Relacionamentos e cardinalidades

No processo de modelagem, a cardinalidade é um passo intuitivo para quem tem experiência com banco de dados. No entanto, para os iniciantes na prática, pode representar um momento de dificuldades no desenvolvimento de diagramas. Na sequência, serão vistos os tipos de diagramas e a cardinalidade que se aplica a todos eles.

A cardinalidade representa a maneira sob a qual as tabelas se relacionam e podem ser de três tipos:

- **Relacionamento um-para-um ou 1:1:** este é utilizado quando uma entidade A se relaciona com uma entidade B e vice-versa, e cada uma deverá ter apenas um registro do outro. O exemplo mais clássico desta cardinalidade seria o relacionamento entre cônjuges, em uma tabela *pessoa*. Logicamente pode-se dizer que “Um marido está relacionado com apenas uma esposa e uma esposa está relacionada apenas com um marido”;
- **Relacionamento um-para-muitos 1:N:** este é utilizado nos casos em que uma entidade A pode se relacionar com uma ou mais entidades B. Um exemplo dessa cardinalidade é o relacionamento entre cidade e estado. Pode ser dito que “Uma cidade está relacionada a apenas um estado, mas um estado tem diversas cidades”;
- **Relacionamento muitos-para-muitos N:N:** este é utilizado quando uma entidade A pode se relacionar com várias entidades B e vice-versa. É nesse caso que será gerada uma entidade associativa

Passo a passo para realizar a modelagem de dados

entre A e B. Um exemplo deste relacionamento será a relação entre *aluno* e *disciplina*. Assim, “Um aluno pode se matricular em várias disciplinas e uma disciplina pode ter vários alunos matriculados”.

Conhecer o que são as cardinalidades e suas respectivas definições facilitará a compreensão dos relacionamentos abordados na parte prática deste artigo.

Modelagem de Dados

A modelagem é o ato de transcrever um problema de um cenário real para uma estrutura de armazenamento de dados. A modelagem de dados pode estar em três níveis de abstração:

- **Modelo Conceitual:** o modelo conceitual é uma visão mais alto nível de um modelo de banco de dados. Independentemente da sua implementação (SGBD), deve abranger as entidades e os relacionamentos de maneira simples, permitindo até mesmo que uma pessoa que não compreenda o conceito de banco de dados possa compreender a regra de negócios representada pelo modelo;
- **Modelo Lógico:** o modelo lógico é uma versão mais detalhada do modelo conceitual, descrevendo além das entidades e relações, também os campos que compõem as tabelas e os respectivos tipos de dados de cada campo;

- **Modelo Físico:** o modelo físico é o banco de dados após ter sua modelagem finalizada. Pode-se dizer que o modelo físico é o banco de dados implementado.

Uma questão que gera muita discussão na criação de um modelo de dados é a ferramenta a ser utilizada para a sua definição. Partindo desta problemática, neste artigo serão abordadas algumas das principais ferramentas do mercado, destacando as características de cada uma delas.

Ferramentas para Modelagem de Dados

Em sua grande maioria são ferramentas que permitem a construção do modelo lógico e a partir deste, gerar o script de criação do banco de dados. No entanto, uma ferramenta que gera o modelo conceitual, lógico e físico deve ser citada, o BrModelo. Esta opção é uma ferramenta de origem acadêmica e dado suas características didáticas é amplamente adotada no ensino de banco de dados. A interface do BrModelo é apresentada na Figura 1.

Descrever as ferramentas para modelagem de dados é dissertar sobre uma diversidade de fatores, que vão desde a usabilidade, completude dos recursos oferecidos, até mesmo a integração de suas funcionalidades ao sistema operacional e ao sistema gerenciador de banco de dados utilizado.

Como pode ser notado, o mercado é vasto de ferramentas. Para confirmar a frase anterior, em sites de downloads uma simples busca pode retornar mais de 100 programas para este fim. Neste artigo serão apresentadas as ferramentas de maior popularidade, visando atender os principais SGBDs do mercado.

Descrever todas as ferramentas certamente levaria a um número alto e nos impossibilitaria de manter um bom grau de profundidade do assunto, além de prolongar um processo que pode ser simplificado. Para auxiliar na restrição para seleção das ferramentas, as seguintes métricas de avaliação foram consideradas:

- **Software Livre:** esta métrica trata sobre a licença do software (GNU, GPL ou BSD). Sim para software livre e Não para software proprietário;
- **Sistema Operacional:** esta métrica indica para qual sistema operacional a ferramenta está disponível (Windows, Linux ou Mac OS);
- **Sistema Gerenciador de Banco de Dados:** após criar o modelo lógico, é esperado de uma ferramenta a capacidade de gerar o modelo físico, bem como a capacidade de integrar os dois modelos nos casos de alteração. Para esta métrica foram considerados dois dos principais bancos de dados proprietários e dois dos principais bancos de dados gratuitos, respectivamente Oracle e SQL Server, e MySQL e PostgreSQL;
- **Ano de Atualização:** esta métrica representa o ano de criação e o ano da última atualização da ferramenta de modelagem, o que possibilita avaliar o tempo de mercado da ferramenta, bem como a última vez que seus recursos foram expandidos.

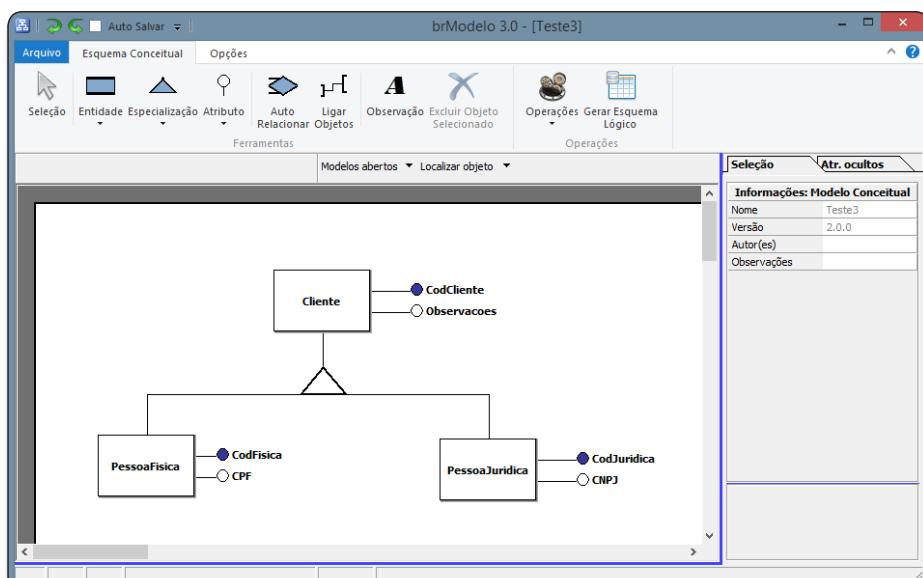


Figura 1. Tela da ferramenta BrModelo

O estudo comparativo entre as ferramentas estudadas de acordo com as métricas definidas está disponível na Tabela 1.

Essa tabela ilustra algumas das principais ferramentas do mercado. No entanto, para realizar a escolha também é necessário conhecer o sistema operacional e o sistema gerenciador de banco de dados. Por exemplo, um DBA que atua em uma empresa que utiliza o mesmo sistema operacional e um mesmo banco de dados terá menos preocupação do que um DBA que pretende

Ferramenta	Software Livre	Windows	Linux	Mac OS	SQL Server	Oracle	PostgreSQL	MySQL	Atualização
DBDesigner	Não	Sim	Sim	Sim	Sim	Sim	Sim	Sim	2015
DBEdit	Não	Sim	Sim	Sim	Sim	Sim	Sim	Sim	2015
Erwin	Não	Sim	Sim	Sim	Sim	Sim	Sim	Sim	2015
MySQL Workbench	Não	Sim	Sim	Sim	Não	Sim	Não	Não	2015
Oracle SQL Data Modeler	Sim	Sim	Sim	Sim	Não	Não	Não	Sim	2015
SQL Power Architect	Sim	Sim	Sim	Sim	Não	Sim	Sim	Sim	2012
Toad	Sim	Sim	Sim	Não	Não	Sim	Sim	Sim	2013

Tabela 1. Comparativo entre ferramentas de modelagem

realizar a implementação do mesmo modelo em bases de dados diferentes, utilizando sistemas operacionais diferentes.

Modelagem de dados na prática

Modelar não é um processo simples, muito menos rápido. É um processo que envolve conhecimento sobre o sistema que será desenvolvido, sobre modelagem, experiência e tempo para colocá-los em prática. Com o intuito de cobrir esse procedimento, até o final deste artigo passaremos por todas as etapas do desenvolvimento de um modelo de dados.

Como esperado, a primeira etapa da criação de um modelo de dados é conhecer o problema. Para este fim será utilizado um caso fictício, apresentado a seguir.

Senhor Alberto Santos é proprietário de uma escola e contratou a empresa X para desenvolver o seu sistema de gestão escolar. Após uma conversa com os profissionais dessa empresa foi elaborado o seguinte texto com os requisitos:

“O sistema deve armazenar dados de coordenadores, professores e alunos. Apenas coordenadores e professores acessarão o sistema como usuários. Cada professor será alocado a uma disciplina e cada aluno poderá ser matriculado em uma ou mais disciplinas. A alocação de alunos em uma disciplina lecionada por um professor caracteriza uma turma e o sistema deverá armazenar todos os dias que uma turma terá aula. Além disso, o sistema deve registrar a presença dos alunos na turma que frequenta, bem como suas notas nas disciplinas.”

Nota

Durante o ciclo de vida de um software há diversos documentos para o levantamento de requisitos do sistema. Neste artigo resumimos esta etapa a um parágrafo com a finalidade de manter o nosso foco na modelagem.

Sabendo que no universo de tecnologia da informação a utilização de ferramentas gratuitas é bem vista, pois sempre oferece um bom custo benefício ao desenvolvedor, como ferramenta de modelagem de dados para o nosso estudo foi escolhido o MySQL Workbench, que está disponível para os sistemas operacionais Windows, Linux e Mac OS.

A partir de agora será explanado um conteúdo misto, relacionando o desenvolvimento do modelo de dados à teoria. Antes disso,

lembre-se que a abstração em nosso contexto é a capacidade de compreender os componentes do mundo real e modelá-los no banco de dados e que essa capacidade de abstrair pode variar de pessoa para pessoa, o que fará com que um mesmo problema possa ser modelado de maneiras diferentes.

Dito isso, começaremos modelando as entidades que devem armazenar as informações de Aluno, Professor e Coordenador. Na abstração do mundo real, todos os três representam pessoas, e por isso há um conjunto de características em comum (Nome, RG, CPF e Endereço, por exemplo). A **Figura 2** demonstra a primeira tabela criada.

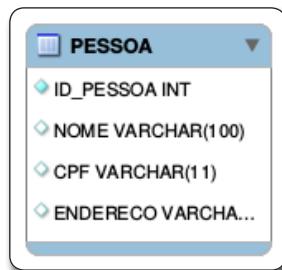


Figura 2. Passo 1 – Estrutura da entidade PESSOA

Note que a tabela pessoa é uma generalização, onde serão concentrados os dados em comum de alunos, coordenadores e professores. A partir disso, deve-se refletir para verificar se há ou não a necessidade de criar outras três tabelas especializadas. Neste caso o mais usual é verificar o que há de diferente entre as especializações. Conhecendo a descrição, sabe-se que o coordenador e o professor serão usuários que acessam o sistema e inicialmente não possuem atributos diferentes entre si, enquanto o aluno não acessará o sistema. Desta forma, a tabela Pessoa será especializada em Aluno e Funcionário (coordenadores e professores), conforme mostra a **Figura 3**.

Dando continuidade ao nosso modelo, sabendo que nesta escola cada disciplina é lecionada por apenas um professor, a **Figura 4** propõe a modelagem para a relação entre disciplina e professor.

Se neste momento for indagado se o banco está na primeira forma normal, a resposta será não. Isto porque a tabela Disciplina está infringindo a primeira forma normal, afinal, os dados do professor estão duplicados e como já há uma tabela para armazená-los, estes podem ser removidos da nova tabela, mantendo-se apenas ID_PROFESSOR.

Passo a passo para realizar a modelagem de dados

Após a normalização da tabela *Disciplina*, nota-se que foi adicionado um campo TELEFONES na tabela *Pessoa*, o que dá a entender que um único campo poderá armazenar diversos telefones. No entanto, isso caracteriza um caso de campo multivalorado, outra infração relacionada à primeira forma normal. Para manter *Pessoa* na primeira forma normal este atributo deve ser removido desta tabela e criada uma nova para armazenar os telefones. Em seguida basta criar uma relação entre *Pessoa* e os telefones pertencentes a ela.

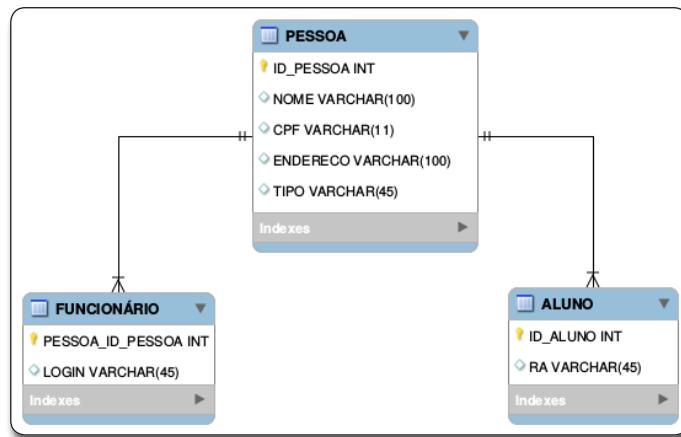


Figura 3. Passo 2 – Modelagem da especialização da tabela Pessoa

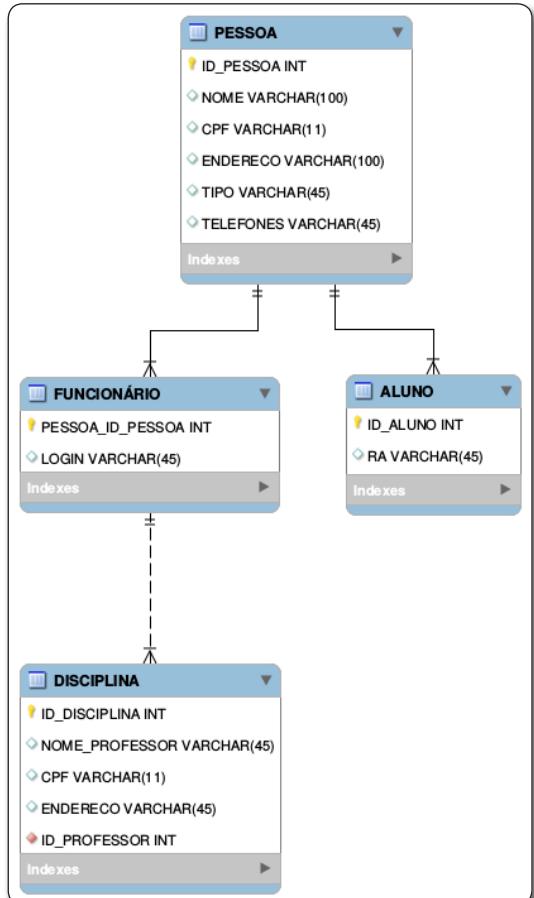


Figura 4. Passo 3 – Modelagem da entidade Disciplina

A modelagem com a tabela *Pessoa* e *Disciplina* normalizadas é demonstrada na **Figura 5**.

Como agora há o entendimento do que se trata a primeira forma normal, sabemos que criar um campo *alunos* em *Disciplina* para

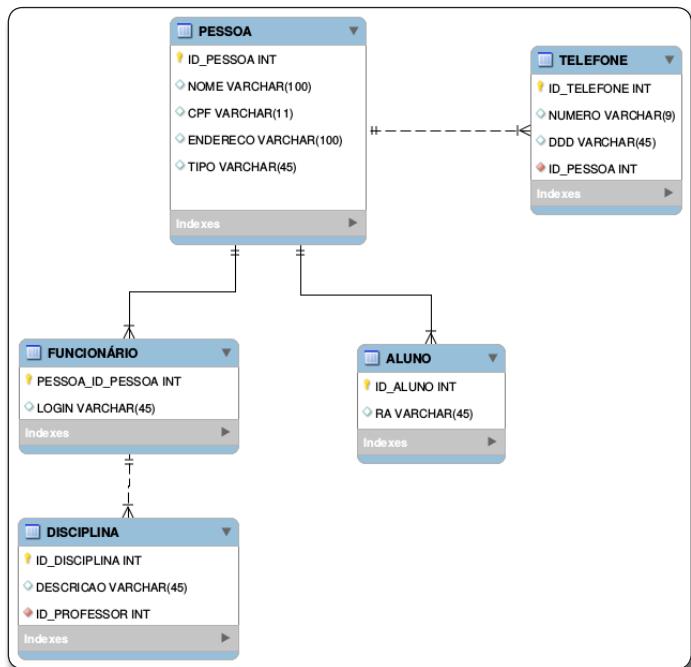


Figura 5. Passo 4 – Normalização das entidades Disciplina e Pessoa

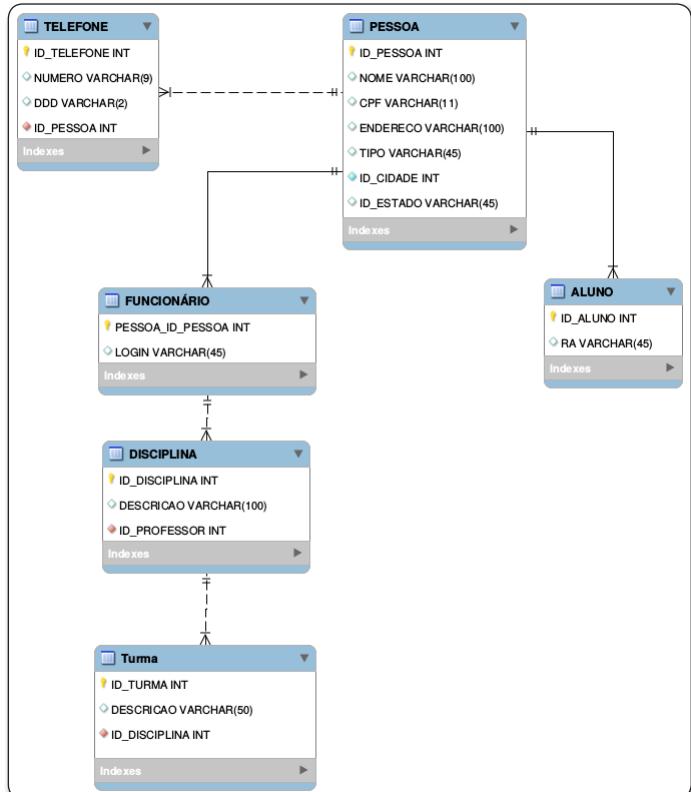


Figura 6. Passo 5 – Normalização da entidade Disciplina e modelagem da entidade Turma

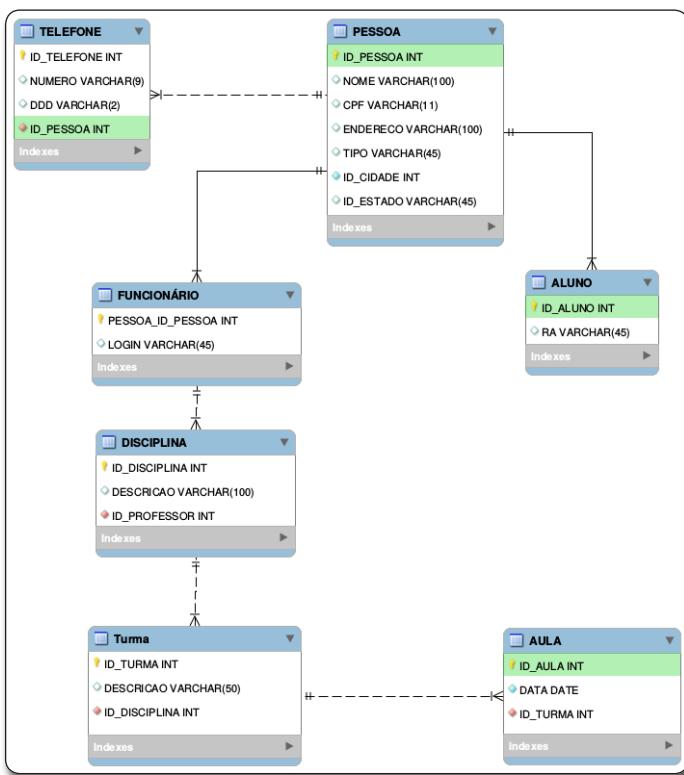


Figura 7. Passo 6 - Modelagem da entidade Aula

adicionar os alunos à disciplina seria errôneo. Note que com essa escolha novamente seria criado um campo multivalorado que infringiria a primeira forma normal. Como solução, observe a **Figura 6**. Esta mostra a tabela *Disciplina* normalizada, bem como a criação da tabela *Turma*, entidade que armazenará quando uma nova turma de determinada disciplina for ofertada.

Como dito anteriormente, o modelo deve armazenar as datas em que uma turma tiver aula. Contudo, caso seja adicionado em *Turma* um campo *Dias_de_Aula* e neste campo armazenar todas as datas, a primeira forma normal seria quebrada com relação à adição de atributos multivalorados. Para respeitar a normalização devemos criar uma entidade *Aula*, a qual será responsável por armazenar todas as datas que uma *Turma* tiver aula. A **Figura 7** ilustra a modelagem dessa entidade.

Com as entidades *Aluno* e *Turma*, para especificar um relacionamento entre elas sem quebrar a 1FN devemos criar a tabela *Aluno_Turma*, composta de duas chaves estrangeiras, sendo uma para *Aluno* e outra para *Turma*. Assim, *Aluno_Turma* conterá todas as informações necessárias. Ademais, adicionamos a ela o atributo *nota*, que representará a nota obtida pelo aluno em determinada disciplina. A esse tipo de entidade, que relaciona outras entidades em um relacionamento do tipo muitos para muitos, damos o nome de associativa (veja a **Figura 8**).

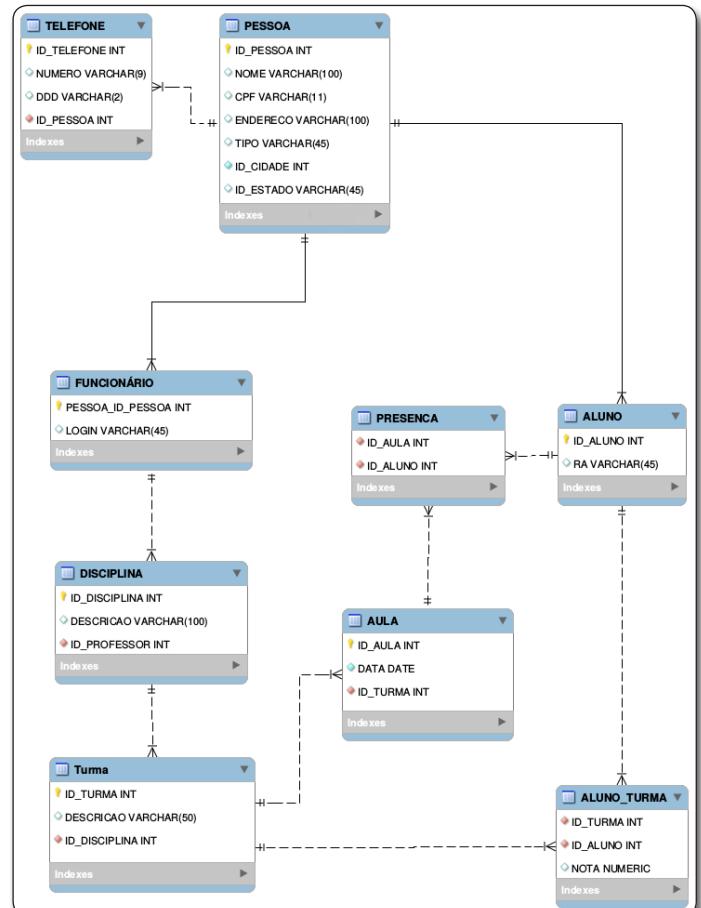


Figura 8. Passo 7 – Modelagem da entidade Aluno_Turma

Figura 9. Passo 8 – Modelagem da entidade Presenca

Passo a passo para realizar a modelagem de dados

Dando continuidade à modelagem, vamos incluir no modelo a presença do aluno. Para isso, aplicaremos a mesma abordagem utilizada no passo anterior, mas relacionando as entidades *Aluno* e *Aula*. A partir disso será criada uma entidade associativa, de nome *Presença*. Esta entidade receberá como chaves estrangeiras *id_aluno* e *id_aula*, e indicará quando o aluno esteve presente em determinada aula. A criação desta tabela é demonstrada pela Figura 9.

Neste momento pode-se dizer que o modelo de dados está quase pronto. Contudo, com o intuito de demonstrar mais uma forma normal, serão adicionadas as tabelas *Cidade* e *Estado*, criando uma relação com *Pessoa* (vide Figura 10).

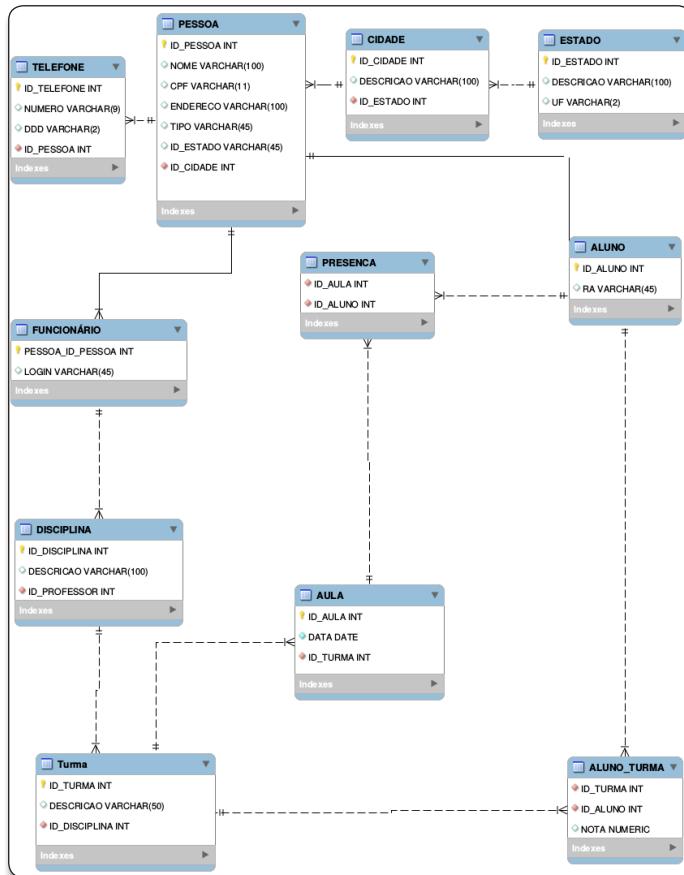


Figura 10. Passo 9 – Inclusão das entidades Estado e Cidade

Assim, a tabela *Pessoa* passará a contar com dois campos relacionados à *Cidade* e ao *Estado*, respectivamente, *id_cidade* e *id_estado*. No entanto, como *id_estado* depende transitivamente de *id_cidade*, afinal, a partir da relação entre as entidades *Pessoa* e *Cidade* pode-se descobrir o *id_estado*, a presença de *id_estado* na entidade *pessoa* está infringindo a terceira forma normal, que especifica que não deve haver dependência transitiva entre os campos.

Para solucionar este caso basta remover o atributo *id_estado* da tabela *Pessoa*. Com isso o modelo para o sistema escolar proposto estará pronto e normalizado de acordo com as três formas normais (vide Figura 11).

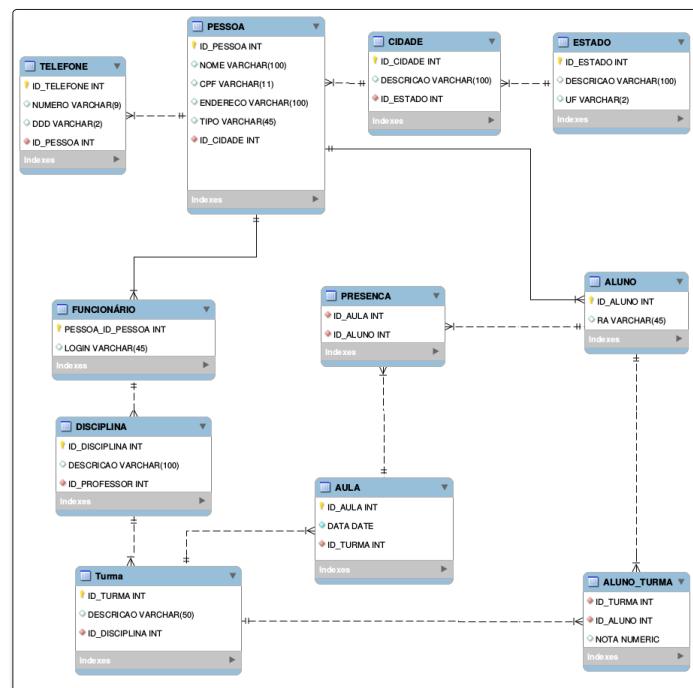


Figura 11. Passo 10 – Modelo Final

A bibliografia tradicional propõe geralmente as três formas normais abordadas neste artigo. Atualmente, no entanto, a literatura já destaca outras duas, além da forma normal de Boyce Codd. Ainda assim, a partir das técnicas abordadas neste artigo já será possível construir modelos de dados de alta qualidade e prontos para serem implantados no banco de dados.

Construir um modelo de dados é algo que pode se tornar simples se utilizarmos devidamente as regras e dominarmos o problema a ser solucionado. As formas normais, as propriedades ACID, as cardinalidades, enfim, todos os conceitos relacionados à criação de modelos de dados se tornarão habituais na rotina de um Administrador de Banco de Dados na medida em que forem colocados em prática, diariamente, em um processo contínuo de aprendizado.

Autor



Rodrigo Ramos Nogueira

wrrodrigo@gmail.com – nogueira.trow.net.br

Bacharel em Informática pela Universidade Estadual de Ponta Grossa, especialista em Desenvolvimento web/mobile, analista de sistemas e DBA. Atualmente mestrandando da Universidade Federal de São Carlos nas linhas de pesquisa de Banco de Dados e Big Data.



Você gostou deste artigo?

Dê seu voto em www.devmedia.com.br/sqlmagazine/feedback

Ajude-nos a manter a qualidade da revista!



Cluster Shared Volume no SQL Server 2014

Failover Cluster Instance com múltiplas sub-redes e CSV

Uma FCI (*Failover Cluster Instance*) faz parte do SQL Server Always On e é executada em um grupo de recursos do WSFC (*Windows Server Failover Clustering*) para prover alta disponibilidade em nível de instância e failover automático. Temos uma única instância do SQL Server instalada em nós do cluster, onde em caso de falha de um nó, o outro se torna proprietário do grupo de recursos e os serviços do SQL Server são iniciados. Isso permite manter o ambiente disponível, evitando a interrupção do serviço. No momento em que uma FCI é iniciada, um dos nós torna-se proprietário dos recursos e a instância de banco de dados se torna online.

A instalação dos binários do SQL Server é realizada em todos os nós envolvidos, porém, neste caso, os serviços serão gerenciados pelo cluster. Cada nó terá apenas os binários do SQL Server, uma vez que os recursos da instância estarão em propriedade do nó ativo. Com o armazenamento de Shared Storage, podemos instalar até 25 instâncias para cada WSFC. Esta limitação ocorre devido ao número de letras disponíveis no alfabeto para utilizarmos nas unidades. Com o CSV (*Cluster Shared Volume*) não temos esta limitação.

Para configuração de uma FCI, é necessário que os servidores envolvidos sejam executados em um cluster. Além disso, é necessário utilizar edições do SQL Server como Enterprise Edition, Business Intelligence ou Standard Edition. A edição Enterprise suporta até dezesseis nós de FCI, enquanto as versões Business Intelligence e Standard suportam até dois nós.

É necessário que o ambiente tenha um servidor DNS (*Domain Name System*) configurado, pois a instalação do SQL Server em cluster registra um nome virtual que referencia a interface de IP. A configuração de DNS deve permitir que os nós do cluster registrem um mapa de endereços de IP para o nome de rede. Caso não seja possível efetuar o registro, a instalação falhará.

Fique por dentro

Neste artigo abordaremos uma funcionalidade recentemente suportada pelo SQL Server a partir da versão 2014, o CSV (Cluster Shared Volume). Esta traz vários benefícios se comparada com um FCI (Failover Cluster Instance) com o armazenamento mais utilizado - Shared Storage. Um FCI faz parte das soluções de alta disponibilidade do SQL Server desde suas versões mais antigas. Esta solução pode tornar a utilização do armazenamento mais eficiente, por utilizar várias instâncias em uma mesma lun (logical unit number), eliminando a limitação de letras atribuídas às unidades lógicas.

Disco de Quórum

Determinamos se um cluster necessita de um disco de quórum dependendo da quantidade de elementos votantes de um cluster. Cada elemento do cluster tem um voto válido, um cluster se mantém ativo ou é iniciado se a maioria dos votos válidos estiver ativa.

Uma configuração de quórum adequada tem efeitos diretos na disponibilidade do cluster, assegurando que em caso de falhas de um nó (problemas de comunicação entre os nós, parada programada para atualizações e manutenções) o cluster se manterá ativo.

No cenário que descreveremos neste artigo utilizaremos dois nós, fazendo necessária a configuração de um disco de quórum.

Múltiplas Sub-Redes

Em uma configuração de múltiplas sub-redes, podemos ter cada nó do cluster respondendo em uma sub-rede diferente ou até mesmo um conjunto delas. Essa configuração pode ser no mesmo local ou em um local geograficamente diferente, o que além de proporcionar disponibilidade, pode fornecer um recurso de recuperação de desastres. Porém, nesta configuração geralmente não haverá compartilhamento do armazenamento, levando à necessidade de replicar os dados através das sub-redes utilizadas na configuração.

Como um exemplo de utilização de FCI com múltiplas sub-redes imagine um cluster com três nós: *node1* e *node2* estão na *subrede1*, enquanto *node3* e *node4* estão na *subrede2*. Esta é uma configuração em cluster de várias sub-redes, que permite fornecer uma alta disponibilidade local adicional devido ao fato do *node3* estar em outra sub-rede. Sua dependência de recurso IP será definida como OR.

Dependências de IP definidas como AND, OR ou mistas não são suportadas, como:

IP1 AND IP2 OR IP3

Não há suporte para mais de um endereço de IP para cada sub-rede. Em caso de configuração de mais de um IP para uma sub-rede, poderão ocorrer falhas de conexão.

A **Figura 1** demonstra a utilização de múltiplas sub-redes em locais geograficamente distintos.

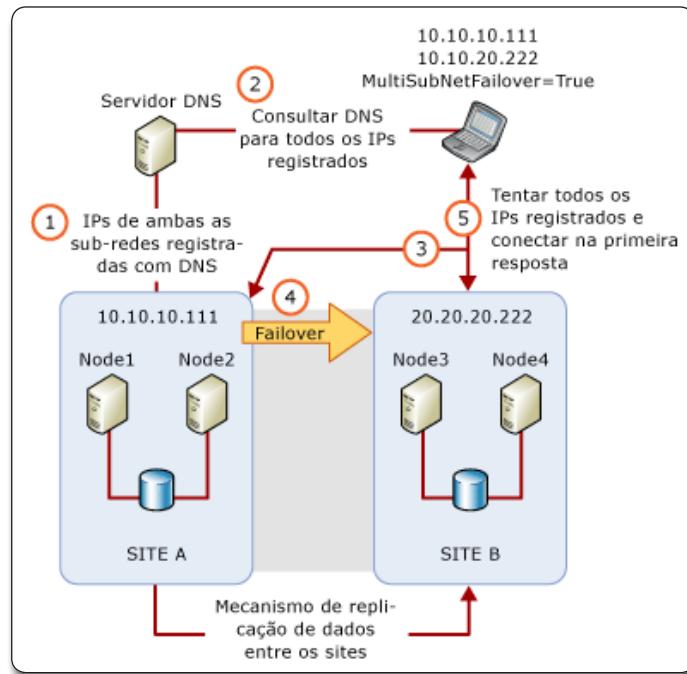


Figura 1. Múltiplas sub-redes em locais geograficamente diferentes

Cluster Shared Volume

Diferente de uma configuração de FCI com Shared Storage, onde cada instância SQL Server requer uma *lun* separada (pelo fato da *lun* ter de fazer o failover junto com a instância SQL), o uso do CSV pode fazer uso de múltiplas instâncias utilizando a mesma *lun*. Esse mecanismo torna a utilização do armazenamento mais eficiente e com maior escalabilidade, uma vez que não há um número de limite de mount points a serem criados para as instâncias SQL Server.

Em uma possível falha de comunicação de um nó com o armazenamento, o CSV fará o roteamento do tráfego através da rede utilizando SMB (*Server Message Block*), mantendo assim a instância

SQL Server operacional. Em caso de um failover, não será necessário a desmontagem dos volumes de um nó e montagem em outro. Outra vantagem é que o CSV faz uso do chkdsk, que permite a detecção de danos aos volumes sem tempo de indisponibilidade e uma melhor gestão da instância SQL Server, pois é possível gerenciar o armazenamento de qualquer nó quando existe acesso aos discos por ambos os nós.

Causa de um failover

A causa de um failover ou reinicialização é definida pela propriedade Failure Condition Level, que determina se o status de integridade pode ou não causar um failover ou reinicialização.

As condições de falhas são definidas de forma crescente considerando níveis entre 1 e 5. Cada um deles inclui as condições dos níveis anteriores além de suas próprias condições. Essas configurações podem ser alteradas para o nível desejado.

A seguir temos os níveis e condições para ocorrências de failover ou reinicialização:

- Nível 0: nenhum failover ou reinicialização automática em nenhum caso de falha. Destinado apenas à manutenção programada;
- Nível 1: failover ou reinicialização quando o servidor estiver inativo. Será gerado um failover ou reinicialização sempre que o serviço do SQL Server estiver inativo;
- Nível 2: failover ou reinicialização em caso de servidor sem resposta. Será gerado um failover ou reinicialização sempre que ocorrer a condição de nível 1 ou a instância não estiver respondendo;
- Nível 3: failover ou reinicialização em caso de erros críticos de servidor. Será gerado um failover ou reinicialização sempre que ocorrerem as condições de níveis 1 e 2 ou o procedimento armazenado SP_SERVER_DIAGNOSTICS retornar ‘erro de sistema’;
- Nível 4: failover ou reinicialização em caso de erros moderados de servidor. Será gerado um failover ou reinicialização sempre que ocorrer as condições de níveis 1, 2, 3 ou o procedimento armazenado SP_SERVER_DIAGNOSTICS retornar ‘erro de recurso’;
- Nível 5: failover ou reinicialização em qualquer condição de falha qualificada. Será gerado um failover ou reinicialização sempre que ocorrer as condições de níveis 1, 2, 3, 4 ou o procedimento armazenado SP_SERVER_DIAGNOSTICS retornar ‘erro de processamento de consulta’.

Momento do failover

No momento de um failover, todos os recursos disponíveis naquele nó serão transferidos para o outro disponível, isto é: o nome de rede registrado na criação do cluster, o endereço IP, o serviço do mecanismo de banco de dados, SQL Server Agent e Analysis Services (caso esteja instalado).

Neste momento, todas as páginas sujas do buffer serão gravadas em disco, o novo proprietário dos recursos inicia o serviço do SQL Server e as conexões serão destinadas ao novo proprietário do serviço. Vale a pena lembrar que não haverá alterações nas aplicações, o nome de rede utilizado será mantido.

O failover pode levar algum tempo até o serviço estar disponível novamente, isso porque o servidor detentor dos serviços poderá

estar com uma quantidade de páginas sujas muito grande no cache do buffer. O processo de failover irá acontecer assim que as páginas sujas forem gravadas em disco. A partir do SQL Server 2012 é possível utilizar pontos de verificações indiretos, o que pode ser uma vantagem na hora de restabelecer o banco de dados após um failover, pois desta forma o SQL Server limita a quantidade de páginas no cache do buffer.

Criação do WSFC

Neste artigo criaremos um cluster no Windows Server 2012 R2. Os passos de instalação não mudam muito em relação à criação no Windows Server 2008. Vale a pena lembrar que o usuário que criará o cluster necessita das permissões para criação de objetos de computador para a OU (Unidade Organizacional) no Active Directory. Caso não tenha as permissões, solicite ao administrador de domínio que pré-configure um objeto de domínio para o cluster. Para começar, é necessário instalar a role *failover clustering* no menu *add roles and features* do Server Manager em todos os nós envolvidos no cluster.

Feito isso em todos os nós, abra o Failover Cluster Manager. O próximo passo é primordial para a criação do cluster, pois garante que suas configurações de ambiente estejam de acordo com as recomendações da Microsoft. Execute a opção *validate configuration*, adicione os nós que participarão do cluster e execute todos os testes conforme recomendado. Após seu término, criaremos nosso cluster com a opção *create cluster*. Defina um nome de rede para seu cluster e abaixo serão exibidas as redes disponíveis. Selecione a(s) rede(s) e adicione um IP válido para cada uma. No próximo passo serão mostradas as configurações definidas para o cluster com nome, IP(s), e nós atribuídos ao cluster.

Com o cluster criado, você poderá ver em seu Domain Controller que foi criado um novo objeto como se fosse um computador de sua rede com o nome atribuído ao cluster em sua criação e a descrição “*failover cluster virtual network name account*”.

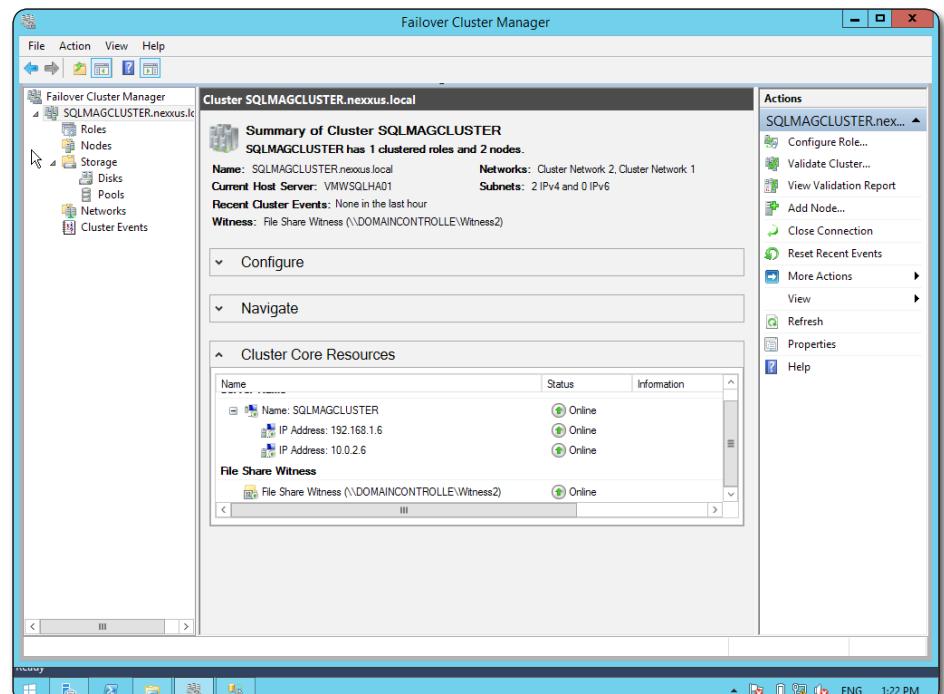


Figura 2. Cluster SQLMAGCLUSTER

O próximo passo é adicionar os discos ao cluster. Expanda a aba *storage/disks*, clique com o botão direito em *add disks*, serão listados os discos disponíveis aos nós. Selecione todos, clique com botão direito e clique na opção *add to cluster shared volumes*. Note que na raiz do C: do nó primário do cluster será criada uma pasta com nome de *ClusterStorage*. Este é nosso mount point, todos os discos estarão disponíveis dentro desta pasta e acessíveis a ambos os nós.

Para melhor administração, renomeie os mount points com a característica de cada um, por exemplo: “dados”, “log”, “instance”, “backup”, etc.

A execução da validação e execução de todos os testes do assistente é uma recomendação de extrema importância, uma vez que a Microsoft somente dá suporte a uma solução de cluster se suas configurações de servidor, rede e armazenamento passarem em todos os testes do assistente. Além de todo hardware estar marcado como certificado para sua edição do Windows Server.

Criação de Quórum

Como estamos utilizando um número par de nós em nossa configuração, precisamos configurar um disco de quórum.

Por se tratar de um ambiente de testes, foi criada uma pasta no servidor de Domain Controller que servirá como nossa File Share Witness.

Clicaremos sob o nome do cluster e em *actions*, na opção *more actions*, teremos a opção de *configure cluster quorum settings*. Podemos optar por três configurações distintas: “Configuração default”, “Selecionar um quórum Witness” ou “Configuração avançada”.

Escolheremos a opção quórum witness e *file share witness*, selecionando a pasta que foi criada no servidor de domínio e compartilhada. Depois, selecionaremos esta pasta e assim criaremos nossa *file share witness*. Foi adicionado o *file share* no servidor de domínio, mas poderia ser em qualquer outro servidor que não seja os nós que compõem o cluster, pois o quórum deve suportar falhas nos nós, se manter ativo e ter poder de voto para manter o cluster ativo. A Figura 2 apresenta o cluster já criado.

Instalação do SQL Server 2014 em cluster

Para instalar o SQL Server em cluster, podemos fazê-lo de duas formas:

- Instalação integrada ao cluster em um nó: desta forma, já teremos uma instância

Cluster Shared Volume no SQL Server 2014

de failover pronta para uso, porém sem alta disponibilidade pois há apenas um nó. Para adicionar outros nós, instalaremos o SQL Server com a opção de *Add Node to a SQL Server Failover Cluster*;

- Instalação avançada: que consiste na preparação dos nós com a opção *Advanced Cluster Preparation*. Ao término desta opção, ainda não há uma instância de fato instalada, por isso, é necessário executar a preparação em todos os nós que participaram do cluster. Depois, execute a instalação em um dos nós, mas o nó estará pronto para a instalação que será feita através da opção *Advanced Cluster Completion*. Após esta instalação, teremos uma instância instalada e operacional.

Como uma instância em cluster é muito semelhante a uma instalação em *single instance*, destacaremos apenas pontos que mereçam maior atenção durante a instalação.

Instalação avançada

Na instalação avançada teremos que preparar os nós que farão parte do nosso cluster. Para isso, executaremos em ambos os nós o setup do SQL Server. Em *advanced* encontraremos a opção *Advanced Cluster Preparation*.

Preparation. Esta opção é muito semelhante à instalação integrada, serão verificados os requisitos de instalação.

Caso já tenha uma instância instalada no nó, ela aparecerá no passo a seguir e será necessário informar um nome de rede para a instância. Este nome de rede será necessário para identificar a FCI na rede. Em seguida, adicionaremos um nome para o recurso de cluster do SQL Server e selecionaremos os discos que serão compartilhados no cluster de SQL Server. Estes discos serão os locais onde os arquivos desse SCBD serão instalados. Caso defina um disco para a instalação que não foi selecionado nesta etapa, sua instalação falhará por falta de permissão no disco.

Definiremos o IP(s) e rede(s) que vamos utilizar. Podemos especificar vários IPs diferentes para um cluster de várias sub-redes, mas apenas um IP é suportado para cada sub-rede. Cada nó deve ser um possível proprietário de, ao menos, um endereço de IP fornecido e, também, os discos que selecionamos nos passos anteriores onde instalaremos os arquivos do SQL Server. Com isso, a preparação da instalação do SQL Server 2014 em cluster é finalizada.

Com ambos os nós preparados, seguiremos com a instalação da instância do SQL Server em cluster. Ao término da instalação, todos os nós que foram preparados previamente já estarão configurados com a FCI.

Instalação integrada

Começamos pelo nó primário do cluster. Execute o setup do SQL Server 2014, vá até a aba *installation* e selecione a opção *New SQL Server Failover Cluster Installation*. Insira a chave de produto e aceite os termos de licença. Serão listadas as rules de failover cluster que serão instaladas.

É necessário que esta etapa seja concluída sem erros. Selecione as features que deseja instalar nesta instância, informe um nome de rede para ela, insira um nome para o grupo de recursos de cluster do SQL Server e selecione os discos que utilizaremos para ele. Serão listadas todas as redes disponíveis, no nosso caso, duas.

Vamos selecionar as redes que utilizaremos e definir IPs válidos para cada rede. Adicionaremos contas de domínio para executar o SQL Agent e o mecanismo de banco de dados. Na aba *data directories* poderemos personalizar nossa instalação definindo o local de cada tipo de arquivo em discos diferentes do nosso mount point, que aparecerá no caminho “C:/ClusterStorage/nomedisco” e com isso terminaremos nossa instalação no primeiro nó. Esta instância já está pronta para uso, porém, sem alta disponibilidade, pois temos apenas um nó instalado. Agora vamos adicionar uma nova instância.

Para adicionar nós à FCI, vamos ao nó que será adicionado e executamos o instalador do SQL Server. Na aba *installation* encontraremos a opção *Add node to a SQL Server Failover Cluster*. Após checar as rules que serão instaladas, será apresentado o nome do recurso de cluster definido na instalação anterior. Poderemos notar que estarão listados o hostname do servidor primário, as features e o nome de rede que definimos durante a instalação anterior. Vamos prosseguir definindo as redes que utilizaremos (observe que os IPs já estão definidos). Adicione as contas de domínio para executar os serviços e já temos nossa

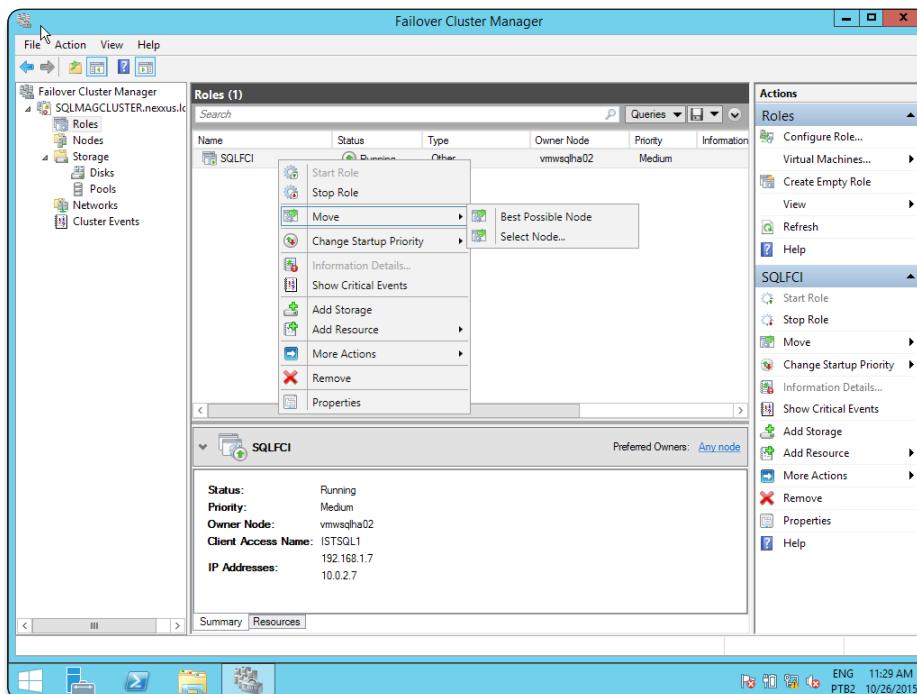


Figura 3. Failover

FCI com dois nós prontos e configurados. O detentor dos serviços no momento é o nó vmwsqlha02. Assim como mostra a **Figura 3**, vamos fazer o failover e mudar para o nó vmwsqlha01. O resultado da ação é mostrado na **Figura 4**.

Remoção de nós do cluster

Remover um nó de uma FCI é tão simples quanto adicionar. Executaremos o setup do SQL Server no assistente de instalação, em seguida, na aba *maintenance*, selecionaremos a opção *remove node from a SQL Server failover cluster*. Será necessário especificar a instância da qual desejamos retirar um nó e o nome do nó será listado no campo *name of this node*. O nó escolhido será removido do cluster.

Recuperação em caso de falha

Em caso de falha em um dos nós, é feito o failover para o outro nó disponível. Caso seja uma falha reparável, realize a manutenção no nó afetado e verifique se ambos estão online no cluster.

Caso seja uma falha irreparável, como um problema de hardware, software ou até mesmo uma substituição de hardware, vá até o gerenciador do cluster e remova o referido nó. Feito isso, realize a manutenção do nó e o adicione ao cluster novamente.

O SQL Server em FCI vem sendo aprimorado a cada versão. A versão 2014 com CSV pode nos trazer diversos benefícios como custos com armazenamento inferiores em relação a outras configurações de alta disponibilidade. Dependendo da forma de licenciamento, pode ser economicamente mais viável uma implantação deste tipo, pois nem sempre é necessário licenciar os nós inativos, podendo proporcionar uma melhor administração dos recursos. Durante o failover, a instância é migrada como um todo. Alertas, logins, planos de manutenção e jobs serão migrados.

A implantação desta solução é relativamente simples, mas devemos estar atentos às recomendações do fabricante e estudar nosso ambiente para ter a certeza de que essa é a solução mais apropriada para a necessidade do ambiente.

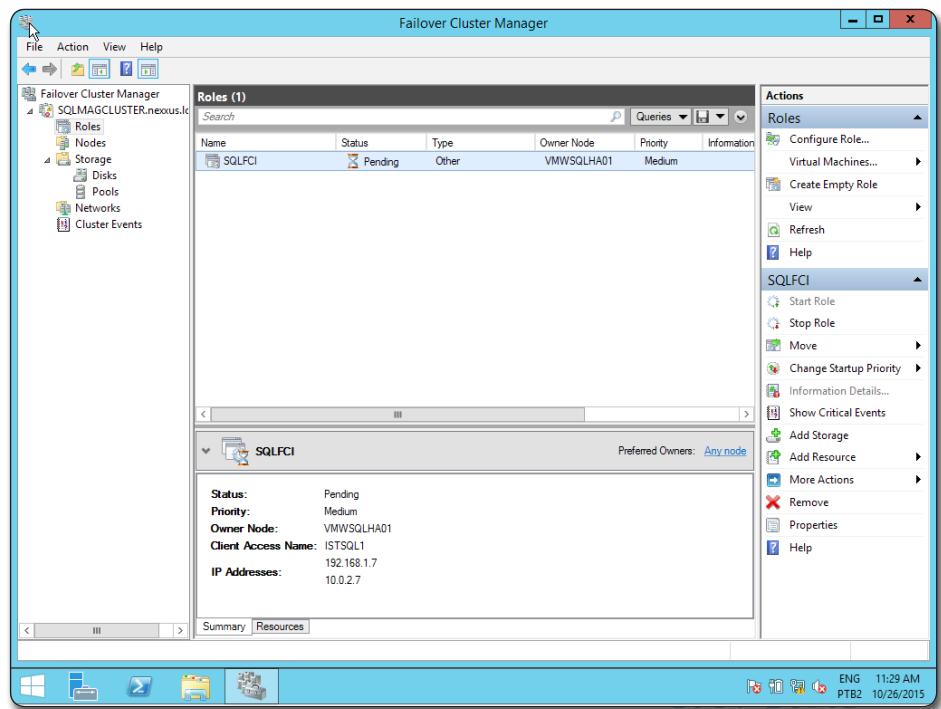


Figura 4. Resultado da mudança para o nó vmwsqlha01

Links:

Clustering de failover e Grupos de Disponibilidade AlwaysOn (SQL Server)
[https://msdn.microsoft.com/pt-br/library/ff929171\(v=sql.120\).aspx](https://msdn.microsoft.com/pt-br/library/ff929171(v=sql.120).aspx)

Atualizar um cluster de failover do SQL Server
[https://msdn.microsoft.com/pt-br/library/ms191009\(v=sql.120\).aspx](https://msdn.microsoft.com/pt-br/library/ms191009(v=sql.120).aspx)

Criar um novo cluster de failover do SQL Server (instalação)
[https://msdn.microsoft.com/pt-br/library/ms179530\(v=sql.120\).aspx](https://msdn.microsoft.com/pt-br/library/ms179530(v=sql.120).aspx)

Noções Básicas sobre Volumes Compartilhados de Cluster em um Cluster de Failover
<https://technet.microsoft.com/library/dd759255.aspx>

Instâncias de cluster de failover do AlwaysOn (SQL Server)
[https://msdn.microsoft.com/pt-br/library/ms189134\(v=sql.120\).aspx](https://msdn.microsoft.com/pt-br/library/ms189134(v=sql.120).aspx)

Usar volumes compartilhados do cluster em um cluster de failover
<https://technet.microsoft.com/library/jj612868.aspx>

Autor



Leandro Romualdo da Silva

leandroromualdosilva@gmail.com

Apixonado por tecnologia, em especial SGBD's, formando em banco de dados pela FIAP e DBA SQL Server e Oracle.



Você gostou deste artigo?

Dê seu voto em www.devmedia.com.br/sqlmagazine/feedback

Ajude-nos a manter a qualidade da revista!



Service Broker no SQL Server 2014

Principais conceitos e implementação

A ideia de um componente arquitetural que permita processar tarefas de forma assíncrona não é nova. Podemos notar este tipo de solução sendo aplicada no envio de e-mail ou na negociação de ações na bolsa de valores, por exemplo. A Microsoft já possuía soluções de envio de mensagem maduras e estabilizadas no mercado quando o SQL Server Service Broker foi adicionado à versão 2005 como uma alternativa a ser usada para o processamento assíncrono, tendo como principal diferencial sua implementação dentro da camada de banco de dados.

O service broker proporciona uma troca de mensagens de maneira transacional, permitindo que desenvolvedores criem aplicações distribuídas sem adicionar grande complexidade em seus códigos atuais. Isso é possível devido à arquitetura SODA (*Service-Oriented Database Architecture*), que permite que a carga de trabalho de um sistema possa ser distribuída em outros servidores e, com isso, aumenta-se a escalabilidade de uma aplicação.

É possível fazer uso de tecnologias de broker em programação assíncrona como processamentos massivos, limpeza de bases ou quando há a necessidade de executar uma tarefa que o tempo de execução não seja limitado ao período em que o cliente está mantendo uma conexão com o servidor. Isso melhora a percepção do usuário em relação à velocidade de processamento.

Um gerenciador de filas se faz necessário quando é preciso limitar o número de requisições atendidas para que não haja concorrência de recursos com solicitações que já estão em processamento. Neste caso, a mensagem é enviada para um serviço que irá colocá-la em sua respectiva fila, então uma thread disponível irá pegar essa mensagem, processá-la e devolverá o resultado.

Um equívoco comum é achar que vale a pena utilizar uma tabela comum como uma fila. Imagine um sistema com centenas de acessos simultâneos, se controlássemos as filas por uma tabela inevitavelmente esbarraríamos em locks. Já as filas do service broker, por fazerem par-

Fique por dentro

Este artigo tem por objetivo apresentar o service broker do SQL Server, um serviço para implementar o processamento assíncrono de mensagens. Esta funcionalidade existe desde a versão 2005 do produto e, se implementada da maneira correta, pode ajudar a solucionar graves problemas de concorrência. Este artigo é útil para lidarmos com situações nas quais o número de usuários cresce além do esperado e a concorrência por acesso aos dados torna-se um gargalo evidente. O service broker é uma alternativa para garantir a entrega dos dados de maneira ágil e organizada.

te da engine do SQL Server, conseguem resolver problemas de concorrência de maneira mais performática.

Novas terminologias

Configurar um service broker, para quem está fazendo pela primeira vez, pode ser um desafio. Um conjunto de novos termos será adicionado ao dia-a-dia do DBA. Alguns deles são:

- **Dialog Conversation:** ponto de início para qualquer troca de mensagens do Service Broker. Antes que uma troca de mensagens possa acontecer, uma conversation precisa ser estabelecida. As mensagens são entregues de maneira EOIO (*exactly-once-in-order*);
- **Conversation Groups:** geralmente usado para agrupar conversas relacionadas a uma mesma lógica de negócio. Normalmente é usado para que mensagens diferentes, que possam ser agrupadas por um mesmo grupo de negócio, sejam processadas em ordem;
- **Message Types:** utilizado para definir qual o tipo dado a mensagem irá conter;
- **Contratos:** os contratos definem quais *message types* podem ser trocados dentro de uma dada conversation;
- **Filas:** são objetos físicos que criam tabelas onde as mensagens serão armazenadas e servem de ponto de represamento para as mensagens que chegam até que elas sejam processadas. Estas tabelas são acessíveis através dos comandos SEND e RECEIVE;
- **Endpoint:** permite conexões com serviços via protocolos do SQL Server: TCP, Namedpipes e Sharedmemory. Uma definição

simples de um endpoint seria a de um objeto localizado na ponta da camada de transporte que permite o acesso através de um ponto único. Um exemplo seria o acesso ao SQL Server via porta 1433. Alguns métodos de autenticação aceitos são: Kerberos, NTLM (*NT LAN Manager*) ou ser baseado em certificados;

- **Serviços:** um serviço é a porta de entrada (*endpoint*) para que haja comunicação entre o Initiator e o Target. O serviço serve como uma abstração para toda a complexidade do broker, pois uma mensagem é enviada para um serviço que a redireciona para uma fila;

- **Rotas:** caminho entre a rede que uma mensagem irá percorrer até chegar ao SQL Server que irá efetivamente executar a tarefa. Um broker pode direcionar uma mensagem para outro até que o local da rota seja definido como “local”, ou seja, a mensagem será executada naquele servidor;

- **Prioridades:** recurso adicionado no SQL Server 2008 que permite quantificar uma prioridade (1 a 10) a um serviço local, remoto ou contrato.

- **Segurança:** segurança é um passo essencial na troca de mensagens. O Service Broker possui dois modelos de segurança: dialog security e transport security. O dialog security usa uma chave de sessão (*session key*) para garantir a troca de mensagens de maneira segura. Se o initiator e o target estão no mesmo servidor, a criptografia é automaticamente habilitada. Já o transport Security utiliza um endpoint para controlar o acesso a um servidor remoto.

Uso do Service Broker na prática

A partir de agora iremos criar os componentes básicos para troca de mensagens com o SQL Server Service Broker. Após a criação dos objetos, iremos exemplificar como realizar o envio, recebimento e tratamento de mensagens.

Criando um DB e habilitando o Broker

Vamos iniciar criando um novo banco. O service broker é habilitado em nível de base de dados de forma bem simples ([Listagem 1](#)). Lembre-se que se seu banco de dados não utiliza o broker e o serviço está habilitado, recursos computacionais estão sendo desperdiçados.

Listagem 1. Criando uma base de dados e habilitando o broker

```
01 CREATE DATABASE SQLMAG_BROKER_DEMO
02 GO
03 USE SQLMAG_BROKER_DEMO
04 GO
05 ALTER DATABASE SQLMAG_BROKER_DEMO SET ENABLE_BROKER
```

Iniciaremos criando um objeto do tipo “message type” que serve para definir qual tipo de mensagem deve ser enviada para que seja considerada válida. Os parâmetros a serem definidos são:

- **AUTHORIZATION:** role ou usuário do SQL Server;
- **VALIDATION:**
 - **None:** qualquer tipo de dado poderá ser colocado na mensagem;

- **Empty:** apenas mensagens vazias;
- **Well_Formed_xml:** apenas mensagem que contenham um XML bem formado (validado pelo XML Parser);
- **Valid_XML With Schema Collection:** somente documentos XML que passarem pela validação de um XSD específico.

Vale lembrar que validar documentos XML no SQL é uma tarefa custosa, já que a mensagem precisa ser carregada por um XML parser. Se for necessário e houver uma camada de aplicação antes ou depois do Service Broker, faça a validação nela. A [Listagem 2](#) demonstra a criação de dois message types.

Listagem 2. Criação de message type

```
01 --CRIA UM MESSAGE TYPE
02 CREATE MESSAGE TYPE SQLMAG_BROKER_DEMO_MESSAGE_TYPE_INITIATOR
03 AUTHORIZATION DBO
04 VALIDATION = WELL_FORMED_XML
05
06 CREATE MESSAGE TYPE SQLMAG_BROKER_DEMO_MESSAGE_TYPE_TARGET
07 AUTHORIZATION DBO
08 VALIDATION = WELL_FORMED_XML
```

O próximo passo será criar um Contrato. Este irá garantir que a troca de mensagens não processará formatos que não foram previamente definidos. Os parâmetros necessários para se criar um contrato são:

- **Authorization:** role ou usuário do SQL Server;
- **SENT BY:**
 - **Initiator:** o broker que iniciou a conversa com um serviço;
 - **Target:** o broker que seria destino de um Initiator;
 - **Any:** ambos podem usar os message types do contrato.

A [Listagem 3](#) demonstra a criação de um contrato que será usado por dois message types.

Listagem 3. Criação de contrato

```
01 --CRIA UM MESMO CONTRATO QUE SERÁ PARTILHADO POR DOIS MESSAGE TYPES
02 CREATE CONTRACT SQLMag_Broker_Demo_Contrato
03 AUTHORIZATION DBO
04 (SQLMAG_BROKER_DEMO_MESSAGE_TYPE_INITIATOR SENT BY INITIATOR,
05 SQLMAG_BROKER_DEMO_MESSAGE_TYPE_TARGET SENT BY TARGET)
```

O próximo passo é a criação de uma fila. Sua criação envolve uma série de parâmetros, descritos a seguir:

- **Status:** a fila está apta a receber mensagens;
- **Retention:** mantém uma cópia de todas as mensagens que chegarem para fins de auditoria;
- **Poison_Message_handling:** quando habilitado, permite que uma fila seja desabilitada caso sejam executados 5 rollbacks consecutivos.

Dentro de Activation temos os seguintes parâmetros:

- **Status:** é usado para habilitar ou desabilitar a execução da

procedure configurada no parâmetro “procedure_name” quando uma mensagem chega;

- **Procedure_name:** nome da procedure que será ativada;
- **Max_queue_readers:** valor que representa o número máximo de cópias da procedure que poderão ser executadas em paralelo. O valor pode ser alterado dinamicamente conforme sua carga de trabalho mude;
- **Execute as:** contexto de usuário sob qual a procedure será executada.

A ativação é a capacidade do broker de monitorar as filas e, após receber uma mensagem, iniciar seu processamento. Ela pode ser externa (haverá um serviço Windows que vai monitorar a fila para fazer o processamento) ou interna (conectada a uma procedure).

Antes de criarmos a fila, precisaremos criar duas procedures “vazias” apenas para que seja possível referenciá-las no script de criação da fila (ver **Listagem 4**). Posteriormente iremos adicionar alguma lógica nelas.

Listagem 4. Criação da procedure vazia

```
01 CREATE PROCEDURE DBO.SQLMAG_BROKER_DEMO_PROC_PROCESSA_
    chegada AS BEGIN
02 RETURN
03 END
04 CREATE PROCEDURE DBO.SQLMAG_BROKER_DEMO_PROC_PROCESSA AS
    BEGIN
05 RETURN
06 END
```

Na **Listagem 5** criaremos duas filas. Uma para envio e outra para recebimento de mensagens, respectivamente.

A seguir temos a criação do último componente da arquitetura de um broker, o serviço. Este é uma camada de abstração para acessar a fila. Nele se define um contrato para validar o conteúdo inserido na fila:

```
CREATE SERVICE SQLMAG_BROKER_DEMO_SERVICE
AUTHORIZATION DBO
ON QUEUE DBO.SQLMAG_BROKER_DEMO_FILA (SQLMAG_BROKER_DEMO_CONTRATO)
```

Quando trabalhamos com troca de mensagens entre diferentes servidores, devemos utilizar rotas que fazem o apontamento para os outros servidores relacionados. Porém, vamos utilizar a rota local (default) do broker que aponta para a instância local. Também não definiremos uma prioridade para uma fila. A **Figura 1** apresenta todos os componentes que criamos para nosso service broker.

Agora que temos a estrutura básica para a troca de mensagens, precisaremos adicionar alguma lógica à procedure *DBO.SQLMAG_BROKER_DEMO_PROC_PROCESSA*. Sua responsabilidade é realizar um tratamento adequado nas mensagens que chegarem na fila (*SQLMag_Broker_Demo_Fila_saida*) (ver **Listagem 6**).

Listagem 5. Criação de duas filas para envio e recebimento de mensagens

```
01 --Cria fila para envio e recebimento de mensagem
02 CREATE QUEUE SQLMag_Broker_Demo_Fila_saida
03 WITH
04   STATUS = ON,
05   RETENTION = OFF,
06   ACTIVATION ( STATUS = ON,
07     PROCEDURE_NAME=dbo.SQLMAG_BROKER_DEMO_PROC_PROCESSA,
08     MAX_QUEUE_READERS=10,
09     EXECUTE AS OWNER),
10   POISON_MESSAGE_HANDLING (STATUS = ON)
11
12 CREATE QUEUE SQLMag_Broker_Demo_Fila_chegada
13 WITH
14   STATUS = ON,
15   RETENTION = OFF,
16   ACTIVATION ( STATUS = ON,
17     PROCEDURE_NAME=dbo.SQLMAG_BROKER_DEMO_PROC_PROCESSA_chegada,
18     MAX_QUEUE_READERS=10,
19     EXECUTE AS OWNER),
20   POISON_MESSAGE_HANDLING (STATUS = ON)
```

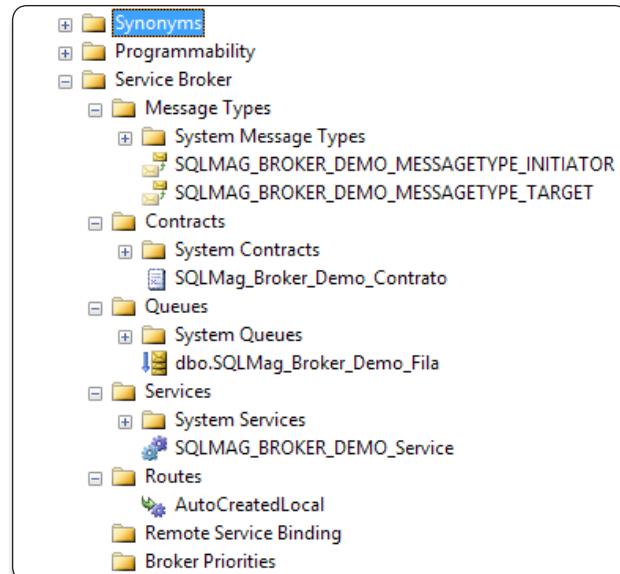


Figura 1. Componentes criados para o service broker

Iniciamos o script declarando as variáveis necessárias para realizar o *parse* do documento XML recém-chegado à fila (linhas 8 a 10). Em seguida, da linha 18 até a 24, usamos a cláusula *receive* que está sendo apontada para nossa queue (*SQLMAG_BROKER_DEMO_FILA_CHEGADA*). Este comando estará dentro de uma estrutura “*WAITFOR () TIMEOUT 100*” que significa que a cada 100ms haverá uma validação de checagem da fila afim de encontrar uma nova mensagem. Se nenhuma nova mensagem for encontrada, a procedure para de ser executada até a próxima validação (linha 27). Caso haja uma nova mensagem, o próximo passo será transformá-la de XML para o formato tabular e, então, inserir as informações nas tabelas de destino (linhas 40 a 66).

Para que a procedure funcione corretamente, será necessário criar algumas tabelas que receberão os dados contidos nas mensagens recebidas. A definição das tabelas pode ser vista na **Listagem 7**.

Listagem 6. Definição da lógica da procedure

```
01 --Recebendo Mensagens
02 ALTER PROCEDURE DBO.SQLMAG_BROKER_DEMO_PROC_PROCESSA_chegada AS
03
04 BEGIN
05
06 --Variáveis de manipulação
07 DECLARE @INICIO DATETIME = GETDATE()
08 DECLARE @handle INT --tratamento xml
09 DECLARE @PrepareXmlStatus INT --tratamento xml
10 DECLARE @MENSAGEM XML, @message_type_name varchar(100),
    @conversation_handle varchar(100)
11
12 --WHILE (1 = 1)
13 BEGIN
14     BEGIN TRANSACTION;
15
16     declare @tabelaStage as table
17     (
18         conversation_handle VARCHAR(100),
19         message_type_name VARCHAR(100),
20         MENSAGEM VARCHAR(MAX)
21     )
22     WAITFOR (
23         --captura primeira mensagem que estiver na fila
24         --Quando o comando receive é executado uma mensagem é removida da fila.
25         RECEIVE TOP (1)
26             @conversation_handle = conversation_handle,
27             @message_type_name = message_type_name,
28             @mensagem = CONVERT(NVARCHAR(MAX),message_body)
29         FROM SQLMAG_BROKER_DEMO_FILA_CHEGADA
30     ), TIMEOUT 10000; --tempo em ms que uma mensagem será aguardada
31
32     --Se nenhuma mensagem chegar, nada faz.
33     IF @@ROWCOUNT = 0
34     BEGIN
35         COMMIT TRANSACTION;
36         return;
37     END
38
39     else begin
40
41         --Inicia tratamento
42
43         /*Le o texto passado como parâmetro (@mensagem), realiza o parse do XML através
```

```
44         da mxmssql.dll e deixa o XML pronto para ser consumido*/
45         EXEC @PrepareXmlStatus= sp_xml_preparedocument @handle OUTPUT,
46             @mensagem
47
48         --CABEÇALHO DA VENDA
49         INSERT INTO VENDAS (venda, vendedor, data, total)
50             (SELECT VENDA, VENDEDOR, DATA, sum(TOTAL*QUANTIDADE)
51             FROM OPENXML(@handle, '/PRODUTOS/VENDA;2')
52                 WITH(
53                     VENDA VARCHAR(100) '@ID',
54                     VENDEDOR VARCHAR(100) '@VENDEDOR',
55                     DATA VARCHAR(10) '@DATA',
56                     TOTAL DECIMAL(10,2),
57                     QUANTIDADE INT
58                 )
59             group by VENDA, VENDEDOR, DATA)
60
61
62         --DETALHES DA VENDA
63         INSERT INTO VENDAS_DETALHES (venda, codigo, descricao, total, quantidade)
64             (SELECT *
65             FROM OPENXML(@handle, '/PRODUTOS/VENDA;2')
66                 WITH(
67                     VENDA VARCHAR(100) '@ID',
68                     CODIGO INT,
69                     DESCRIAO VARCHAR(100),
70                     TOTAL DECIMAL(10,2),
71                     QUANTIDADE INT
72                 )
73             )
74
75         /*Libera a área de memória alocada pelo documento no primeiro passo.*/
76         EXEC sp_xml_removedocument @handle
77
78         DECLARE @HASH VARCHAR(100) = (SELECT conversation_handle FROM
79             @tabelaStage)
80
81         INSERT INTO LOG_BROKER_MANUAL (HASH_MENSAGEM, INICIO, FINAL)
82             VALUES (@HASH, @INICIO, GETDATE())
83
84         COMMIT TRANSACTION;
85     end
86
87 END --PROC
```

Listagem 7. Criação das tabelas

```
01 --TABELA QUE IRÁ ARMAZENAR O CABEÇALHO DE UMA VENDA
02 CREATE TABLE VENDAS(
03     VENDA INT,
04     VENDEDOR VARCHAR(100),
05     DATA DATE,
06     TOTAL DECIMAL(10,2)
07 )
08
09 --TABELA QUE IRÁ ARMAZENAR OS DETALHES DA VENDA
10 CREATE TABLE VENDAS_DETALHES(
11     VENDA VARCHAR(100),
12     CODIGO INT,
13     DESCRIAO VARCHAR(100),
14     TOTAL DECIMAL(10,2),
15     QUANTIDADE INT
16 )
17
18 --TABELA QUE IRÁ ARMAZENAR LOG DE EXECUÇÃO DE CADA IMPORTAÇÃO.
19 CREATE TABLE LOG_BROKER_MANUAL(
20     ID INT IDENTITY(1,1),
21     HASH_MENSAGEM VARCHAR(100),
22     INICIO DATETIME,
23     FINAL DATETIME,
24     DATA_REGISTRO DATETIME DEFAULT(GETDATE())
25 )
```

Basicamente, criamos as tabelas de vendas, detalhes da venda e uma terceira responsável por armazenar o log de execução de cada importação.

Troca de Mensagens

Para que possamos simular uma troca de mensagens real, será necessário que um documento XML seja criado e fornecido como um parâmetro para o SQL Server. A **Listagem 8** apresenta um trecho de um documento XML que representa a venda de livros. Na linha 3 declaramos uma variável do tipo XML, que será usada para armazenar o documento XML relacionado à venda de livros e uma variável uniqueidentifier (@dialog_id), que será usada na **Listagem 9** como parte da sintaxe de criação de uma conversation. As demais linhas da listagem definem um conjunto de dados de venda.

Listagem 8. Documento XML

```
01 --Para nosso exemplo usaremos um XML que simula a troca de mensagens
02 --de venda de livros.
03 declare @mensagem xml, @dialog_id uniqueidentifier
04 set @mensagem =
05 '<?xml version="1.0"?>
06 <PRODUTOS TIPO="XML_SQLMAGAZINE">
07   <VENDA ID="2">VENDEDOR="DHIEGO PIROTO" DATA="2015/01/01">
08     <CODIGO>12</CODIGO>
09     <DESCRICAO>GUIA DE BOLSO MSSQL SERVER</DESCRICAO>
10     <TOTAL>331.99</TOTAL>
11   <QUANTIDADE>10</QUANTIDADE>
12 </VENDA>
13 <VENDA ID="2">VENDEDOR="DHIEGO PIROTO" DATA="2015/01/01">
14   <CODIGO>15</CODIGO>
15   <DESCRICAO>C# GUIA DO PROGRAMADOR</DESCRICAO>
16   <TOTAL>59.90</TOTAL>
17   <QUANTIDADE>3</QUANTIDADE>
18 </VENDA>
19 <VENDA ID="2">VENDEDOR="DHIEGO PIROTO" DATA="2015/01/01">
20   <CODIGO>14</CODIGO>
21   <DESCRICAO>GUIA DO XML</DESCRICAO>
22   <TOTAL>95.9</TOTAL>
23   <QUANTIDADE>10</QUANTIDADE>
24 </VENDA>
25 </PRODUTOS>'
```

Enviando uma mensagem

A atividade primária de um broker é a troca de mensagens. Os comandos básicos para as atividades de envio e recebimento de mensagens são, respectivamente, SEND e RECEIVE.

Porém, antes de estabelecer uma troca de mensagens, será necessário iniciar uma conversation. Isso poderá ser feito com o comando BEGIN DIALOG CONVERSATION (**Listagem 9**).

As primeiras quatro linhas fazem parte da estrutura de criação básica para que uma conversation seja estabelecida. A variável @DIALOG_ID gerará um GUID (16 bytes) que será usado na **Listagem 9** quando a mensagem for enviada.

Existe uma camada de abstração quando realizamos trocas de mensagens. Esta camada recebe o nome de serviços, sendo as mensagens trocadas entre eles. Nas linhas 2 e 3 são definidos os serviços de envio e recebimento da mensagem, respectivamente.

Esta mensagem será validada por um contrato, cujo o nome é definido na linha 4.

Para finalizar a criação da conversation, temos as opções avançadas que, em nosso caso, explicitamos que a mensagem não será enviada de forma encriptada (encryption = off) e que o tempo limite durante o qual a conversation permanecerá aberta em segundos é de lifetime = 10. Ambos os endpoints (entrada e saída) devem finalizar o processamento antes do lifetime expirar.

Listagem 9. Definição de uma conversation

```
01 BEGIN DIALOG CONVERSATION @DIALOG_ID --INICIA DIALOGO
02   FROM SERVICE SQLMAG_BROKER_DEMO_SERVICE_SAIDA --SERVIÇO PARA
          QUAL UMA MENSAGEM SERÁ ENVIADA APÓS O PROCESAMENTO
03   TO SERVICE 'SQLMAG_BROKER_DEMO_SERVICE_CHEGADA' --SERVIÇO DE
          DESTINO
04   ON CONTRACT SQLMAG_BROKER_DEMO CONTRATO --CONTRATO QUE IRÁ
          VALIDAR A MENSAGEM.
05   --OPÇÕES AVANÇADAS
06   WITH
07     ENCRYPTION = OFF,
08     LIFETIME = 2000;
```

Para que possamos enviar mensagens, faremos uso do comando SEND e ele possui dois parametros válidos:

- ID da conversation;
- Conteúdo da mensagem.

Estes parâmetros serão retornados através dos comandos contidos na **Listagem 9**. Os detalhes de como enviar uma mensagem estão definidos a seguir:

```
SEND ON CONVERSATION @DIALOG_ID MESSAGE TYPE SQLMAG_BROKER_DEMO_
MESSAGETYPE_INITIATOR (@MENSAGEM)
```

Depois que uma mensagem for postada na fila, nossa procedure estará apta a perceber que a fila não está mais vazia e iniciará o processamento dessa nova mensagem, em nosso caso, capturando o XML e aplicando o devido tratamento para que os dados possam ser inseridos nas tabelas de destino.

Consultando Filas

Se você quiser saber quantas mensagens ainda estão em uma dada fila, basta executar uma instrução select simples no objeto *queue*. A consulta a seguir retorna a lista de mensagens aguardando para serem processadas. A **Figura 2** apresenta o resultado da execução da consulta:

```
SELECT CONVERT(NVARCHAR(MAX),MESSAGE_BODY), --Mensagem
      *
FROM SQLMAG_BROKER_DEMO_FILA_SAIDA
```

Monitoramento de Filas

Assim como todo produto Microsoft, é possível monitorar o comportamento de uma determinada funcionalidade através do

MSG_XML		status	priority	queuing_order	conversation_group_id	
<PRODUTOS TIPO="XML_SQLMAGAZINE"><VENDA ID="2" V...		1	5	99	CD73EEDC-3382-E511-8297-005056C00008	
conversation_handle		message_sequence_number	service_name	service_id	service_contract_name	service_contract_id
CE73EEDC-3382-E511-8297-005056C00008		0	SQLMAG_BROKER_DEMO_SERVICE_chegada	65546	SQLMag_Broker_Demo_Contrato	65539

Figura 2. Mensagens aguardando serem processadas

Performance Monitor (*perfmon*) e DMVs (*Dynamic Management Views*). DMVs ou DMFs (*Dynamic Management Functions*) são objetos da engine do SQL Server utilizadas para consultar seus metadados ou informações mais detalhadas sobre o que um determinado serviço está executando.

Monitorar a extensão de uma fila é fundamental para entender se o número de threads que atendem uma fila está sub-dimensionado ou se o processamento da mensagem está lento.

O SQL Server possui duas DMVs muito úteis para consultar o que está acontecendo no broker:

- sys.conversation_endpoints: lista todas as conversations abertas e que estão sendo usadas. É preciso ter cuidado com a quantidade de conversations abertas, pois isso pode causar lock de páginas no tempdb, levando a um crescimento demaisado do arquivo de dados;
- sys.transmission_queue: lista todas as mensagens que tiveram algum problema durante sua transmissão.

É importante considerar que o Service Broker é um serviço de troca de mensagens confiável e, dependendo dos requisitos do seu sistema, uma ótima alternativa para seu ambiente, já que toda sua lógica está incorporada dentro do SQL Server, permitindo ganhos de performance significativos.

Autor



Dhiego Piroto

dhiegopiroto@gmail.com



Graduado em Sistemas de Informação. Atua há seis anos na área de TI, apoiado na tecnologia de banco de dados SQL Server 2005 a 2014. Possui o título de MCP e MCTS Microsoft SQL Server e é membro ativo da comunidade MSDN SQL Server.

Links:

Service Broker Performance

[https://technet.microsoft.com/en-us/library/ms166135\(v=sql.105\).aspx](https://technet.microsoft.com/en-us/library/ms166135(v=sql.105).aspx)

Uso do ssbdiagnose Utility

<https://msdn.microsoft.com/en-us/library/bb934450.aspx>

Perfmon

[https://technet.microsoft.com/en-us/library/ms166069\(v=sql.105\).aspx](https://technet.microsoft.com/en-us/library/ms166069(v=sql.105).aspx)

Você gostou deste artigo?

Dê seu voto em www.devmedia.com.br/sqlmagazine/feedback

Ajude-nos a manter a qualidade da revista!



Estratégias de backup e restore no PostgreSQL

Saiba como criar um ambiente robusto

Uma das atividades mais importantes de um administrador de banco de dados é, sem dúvida, planejar e implementar uma boa estratégia de backup para se prevenir de possíveis desastres. Nem sempre é possível prever tudo, mas para se prevenir de possíveis falhas é importante considerar desastres que podem ocorrer com seu banco de dados: terremoto, furacão, incêndios, falha de hardware, falha de software e até falha humana. Para se proteger de todas elas existe um custo e a implementação vai depender do nível de investimento que será feito e dos riscos que a empresa está disposta a correr. Nem sempre o administrador de banco de dados consegue se proteger da maioria dos desastres, pois não depende só dele. Nesses casos é interessante deixar os responsáveis pelos sistemas cientes da situação.

Para criação da melhor estratégia de backup para o ambiente, temos que considerar quais as necessidades dos clientes que usam o serviço de banco de dados. A seguir abordaremos quais pontos devem ser levados em consideração.

Uma sigla bastante importante para definição da nossa estratégia é o ANS (Acordo de Nível de Serviço) ou em inglês SLA (Service Level Agreement). Proveniente do ITIL, biblioteca com melhores práticas de infraestrutura de TI, o ANS é o acordo entre o provedor de serviços de TI e o cliente para um determinado tipo de serviço. No nosso caso, o serviço em questão é o de banco de dados. E o acordo será o tempo de indisponibilidade desse serviço durante uma parada não programada. Quanto tempo é aceitável para o cliente ficar sem esse serviço no caso de um desastre?

No caso de um desastre, dependendo da estrutura da empresa, é muito provável que será necessário um restore usando o backup realizado. E claro, de nada vale um backup se o restore não funciona. Por isso, um outro conceito que precisamos entender aqui é o RTO (Recovery Time Objective), que é o tempo gasto do desastre até a completa recuperação dos dados. De uma maneira direta, nossa estratégia de backup precisa ter um RTO que atenda o ANS relacionado a desastres.

Fique por dentro

Este artigo objetiva mostrar como implementar uma boa estratégia de backup e de restore para seu banco de dados. Para isso, será abordado o funcionamento do Continous Archiving, também conhecido como backup incremental. Vamos saber em que situações este tipo de backup é útil, quais os passos para habilitá-lo, como realizar o restore e o recover da base de dados. Se você tem um banco de dados PostgreSQL e considera que os dados contidos nele são importantes, então é recomendável que você entenda como criar uma boa estratégia de backup e que também tenha em um bom procedimento para realizar a recuperação desses dados. Caso contrário, poderá ter problemas no dia em que precisar realizar um restore no ambiente de produção.

Outro acerto que deve ser definido é o período aceitável de perda de informações em caso de falha. Existe um período aceitável em que, no caso de um restore, os dados podem ser perdidos ou não é aceitável de maneira alguma perder dados?

A criação dos backups históricos também terá um peso grande na definição da estratégia. Em alguns casos em que ocorre o expurgo das informações do banco de dados em determinados períodos, se faz necessária a criação de um backup que será armazenado por um longo período. Essa retenção visa atender algumas leis ou até mesmo auditoria da empresa.

Para atender todas essas situações, é necessário um investimento bem alto, pois é preciso ter uma boa infraestrutura. Além desse investimento, existe um outro essencial que é a alocação de pessoal. Assim como a etapa da implementação, existe toda a parte da criação da documentação, revisão e atualização dos processos. Mesmo ciente dos pontos necessários, nem sempre as empresas estão dispostas a pagar. Existem casos em que os dados armazenados não justificam um enorme valor de investido no backup.

WAL - Write-Ahead Logging

Para começar, é importante termos em mente que os logs de transação, conhecidos como logs de WAL no PostgreSQL, possuem um papel fundamental para garantir a integridade dos dados e também têm uma função muito importante no desempenho do banco de dados.

Os logs de WAL nada mais são do que um histórico das transações ocorridas no banco de dados. Todas as operações são gravadas primeiramente nos logs de transação para só depois serem escritas nos arquivos de dados do banco, também conhecidos com datafiles. Por isso a importância deles no desempenho, pois como temos a certeza de que os logs possuem a informação guardada, o banco não precisa ficar se preocupando em gravar todas as alterações em tempo real nos datafiles.

Os arquivos de WAL também são necessários para integridade. Com o histórico guardado nos arquivos de log, mesmo as informações que ainda não foram salvas nos datafiles depois de uma queda de energia ou outro desastre podem ser refeitas. Esse processo é chamado de *roll-forward recovery*, também conhecido como REDO. Em resumo o fluxo seria: usuário inseriu uma informação e logo em sequência realizou um *commit*; os dados são descarregados do buffer dos logs para os arquivos de redo, para só depois serem escritos nos arquivos do banco de dados.

No PostgreSQL, os arquivos de log são armazenados dentro do diretório pg_xlog que fica armazenado no diretório de dados da sua instância. Por padrão, o banco inicia com 3 segmentos WAL. Estes segmentos possuem um tamanho total de 16M e são compostos por blocos de 8k.

Tipos de backup e continuous archiving

Existem dois tipos de backups: o lógico e o físico. O backup lógico é geração dos comandos SQL necessários para recuperação da base de dados dentro de um arquivo texto. Este tipo de backup não é muito indicado para recuperação de desastres, pois o arquivo com os comandos para recuperação foi gravado em um momento específico do tempo e durante a recuperação dos dados teremos que voltar o banco para este mesmo momento em que foi gerado. Imagine um ambiente em que é realizado um backup lógico todo dia às 2h e que no mesmo dia houve um problema no seu hardware e será preciso recuperar os dados de seu backup. Se você estiver usando apenas o backup lógico, você só conseguirá recuperar seu banco de dados até as 2h. Então você terá perdido pelo menos 10h de informação da sua base de dados.

O lógico funciona mais como um incremento na sua estratégia e pode ser usado, por exemplo, para backups históricos que devem ser armazenados por um longo período.

Durante o backup físico é feita uma cópia binária dos arquivos do seu banco de dados e ele pode ser dividido em quente e frio. O backup frio nada mais é do que parar todo seu banco de dados e realizar a cópia de todos os arquivos, mas aí temos o problema da indisponibilidade da base. Dependendo do ambiente, esta indisponibilidade pode não ser aceitável. Já no quente, não precisamos parar o banco de dados. Mas esse tipo de backup só é possível quando temos habilitado o arquivamento de logs WAL.

Como todo arquivo importante, os arquivos de REDO também devem ter seus backups para evitar perda de informação em caso de sua corrupção. Então é importante que seja habilitado o arquivamento dos logs de WAL, que nada mais é que um backup dos seus arquivos de REDO. Após habilitado, é recomendado que

os logs arquivados sejam gravados em um servidor diferente de onde está seu banco de dados.

O Continuous Archiving é nome dado aos backups contínuos das sequências dos logs WAL. Esses backups serão usados em conjunto com o backup físico do seu banco de dados. No caso de um recover, precisaremos ter a sequência contínua dos logs de REDO desde o último backup realizado. Desta maneira conseguiremos realizar a recuperação completa do banco de dados ou até mesmo recuperar para um determinado momento no tempo, procedimento conhecido como PITR (*Point-in-Time Recovery*). Por isso que os backups físicos são recomendados no caso de um desastre; diferente dos lógicos, podemos recuperar o banco em qualquer ponto no tempo. Além das utilidades já citadas, este arquivamento será bastante importante para economizar tempo nos backups de grandes bases de dados onde é inviável realizar uma cópia completa de todo o banco de dados todos os dias da semana.

Imagine um banco de dados de 800Gb. Dependendo da infraestrutura disponível, seria muito custoso ou até impossível realizar um backup completo de toda estrutura da sua base de dados todos os dias da semana sem ter um impacto direto no ambiente de produção. Para otimização deste tempo, poderia ser usada a combinação do backup base de todo o banco durante um ou dois dias na semana e, durante os dias restantes, ser realizado apenas o backup dos archives dos logs. E no caso de uma recuperação do seu banco, você iria combinar o seu backup base com a sequência de logs arquivada.

Outra função importante dos archives é a construção do banco de dados standby. Os logs arquivados no banco de dados principal são enviados para o secundário e lá as transações são aplicadas para criar um espelho do banco de dados de produção. No caso de um desastre, a recuperação do seu ambiente com o banco de dados standby seria bem mais rápida. Porém, como é algo que requer um investimento maior, nem sempre ele é implementado.

A escrita nos logs de REDO é sequencial e cíclica. Como padrão temos três segmentos, a escrita é alternada entre eles. Essa alternância entre os segmentos é conhecida no PostgreSQL como xlog switchs. E sempre que um switch ocorre, é gerado o arquivamento do segmento antigo. Mas em que momento ocorre essa transição? A seguir temos alguns:

- Durante o início de um backup online;
- Após o uso da função pg_switch_log;
- Durante o shutdown do banco de dados, ocorre o switch para que os arquivos atuais sejam arquivados.

Restore dos backups realizados

Com certeza, mais importante do que o backup é o restore. Imagine você realizar um grande investimento em infraestrutura, levar vários dias planejando sua estratégia de backup e, no dia em que houver um desastre no seu servidor de banco de dados de produção, você não conseguir realizar a recuperação de toda a informação necessária porque não havia testado antes se seus backups realmente conseguiram recuperar as informações conforme a necessidade da empresa. Todo seu esforço foi jogado fora.

Estratégias de backup e restore no PostgreSQL

Por esta razão, além de ter um bom planejamento para os backups, você deve ter uma excelente estratégia para o restore. Tudo que citamos para o backup, é válido para o restore. Também teremos que ter um bom investimento para criação de toda a documentação, atualização, revisão dos procedimentos para realização da recuperação. Tudo isso com a intenção de que nossos backups possam ser constantemente testados e validados para que no dia que seja necessário, eles funcionem.

Além da documentação de restore usando os backups feitos para desastres, é recomendado que seja feita a documentação para recuperação dos backups históricos.

Nem sempre é possível, devido a custos, mas o ideal é que existisse um ambiente semelhante ao de produção para que a recuperação dos backups fosse testada com regularidade. Assim, todo o procedimento poderia ser validado com uma precisão maior.

Habilitando o arquivamento dos logs de WAL

Para criação destes exemplos foi usado o ambiente com o sistema operacional CentOS 6 e PostgreSQL 9.4. Para começar, vamos criar uma estrutura diferente do local onde está instalado nosso banco de dados. Esta nova estrutura será usada para armazenar os backups. No nosso exemplo, será criado o diretório /backup e dentro dele a pasta /archives. Como citado antes, é interessante que o /backup esteja em um disco diferente do seu servidor de banco de dados. É importante também certificar-se de que o usuário postgres tem acesso aos diretórios criados:

```
mkdir -p /backup/archives  
chown postgres.postgres /backup  
chown postgres.postgres /backup/archives
```

Após a criação dos diretórios que receberão os arquivos, vamos habilitar o arquivamento dos logs de WAL. Primeiro, iremos fazer login com o usuário postgres, logo em seguida conectar no banco de dados usando o utilitário psql para identificar onde se encontra o arquivo postgresql.conf. Para localizá-lo, usaremos o comando `show config_file`. É nele que faremos as configurações necessárias. Como mostrado na **Listagem 1**, nosso arquivo está no caminho /postgres/postgresql.conf.

Listagem 1. Localizando o arquivo postgresql.conf

```
01 psql (9.4.5)  
02 Postgres=# show config_file;  
03 config_file  
04 -----  
05 /postgres/postgresql.conf
```

Após localizar o arquivo postgresql.conf, devemos editá-lo com os valores descritos na **Listagem 2** para que o arquivamento dos logs de REDO seja habilitado:

- `wal_level`: determina a quantidade de informação que é armazenada no WAL. O valor padrão é *minimal*, que guarda somente a informação necessária para recuperar a base de um desastre.

No nosso caso, precisaremos alterar o parâmetro para `archive` ou `hot_standby`:

- `archive_mode`: quando alterado para *on*, faz com que os segmentos de WAL sejam enviados para a localização de armazenamento que será indicada no parâmetro `archive_command`. O `archive_mode` não pode ser habilitado quando o `wal_level` está como *minimal*;
- `archive_command`: aqui iremos passar o comando shell que irá realizar o arquivamento dos logs de WAL. O %p será substituído pelo caminho que será armazenado e %f é alterado pelo nome do arquivo;
- `max_wal_senders`: número de processos que fazem o envio dos logs arquivados de WAL para os servidores standby. O padrão é 0, indicando que a replicação está desabilitada. No nosso caso iremos habilitar pois será necessário para o uso do utilitário `pg_basebackup`.

Listagem 2. Parâmetros para habilitar arquivamento dos logs WAL

```
01 wal_level = archive  
02 archive_mode = on  
03 archive_command = 'cp %p /backup/archives/%f'  
04 max_wal_senders = 1
```

Criando o backup base

Agora que já temos os logs de REDO sendo copiados, podemos criar nosso backup online completo da base de dados. Iremos utilizar o utilitário `pg_basebackup`. Ele realiza uma cópia de todos os arquivos do cluster de banco de dados PostgreSQL. Caso você tenha mais de um banco de dados nas suas instâncias, não é possível realizar o backup de apenas um banco de dados.

Este tipo de backup é feito utilizando o protocolo de replicação para conexão com o banco de dados, por isso, a realização da conexão com o banco deve ser feita por um usuário que tenha permissões de replicação. Outro pré-requisito é que o parâmetro `max_wal_senders` esteja com a configuração mínima necessária para que haja pelo menos uma sessão disponível durante a realização do backup.

Vamos criar um usuário com as permissões necessárias para realizar nosso backup. Usando um super usuário do banco de dados, faça login na base e execute o código a seguir:

```
postgres=# create role usu_bkp replication login password 'backup';
```

Após a criação do usuário com o privilégio `Replication`, precisamos adicionar uma nova linha ao arquivo `pg_hba.conf`. Essa nova linha é necessária para que nosso novo usuário possa se conectar ao banco de dados. Como este arquivo pode variar em cada instalação, usaremos o comando `show hba_files` para descobrir onde ele se encontra, como mostrado na **Listagem 3**.

Feito isso, adicione a linha apresentada na **Listagem 4** ao final do arquivo `pg_hba.conf`. Logo em seguida, reinicie o banco de dados para que as mudanças tenham efeito.

Listagem 3. Localizando o arquivo pg_hba.conf

```
01 psql (9.4.5)
02 postgres=# show hba_file;
03   hba_file
04 -----
05 /postgres/pg_hba.conf
```

Listagem 4. Definição de permissão e reinício do servidor

```
01 -- Adição de permissões arquivo pg_hba.conf
02 host replication usu_bkp 127.0.0.1/32 trust
03
04 -- Restart serviço PostgreSQL
05 pg_ctl restart
```

Depois de configurar o arquivamento, iremos realizar o backup completo de toda estrutura da base. Usaremos o utilitário pg_basebackup passando os parâmetros necessários para nossas configurações:

```
pg_basebackup -h127.0.0.1 -U usu_bkp -D /backup/`date +%d%m%Y` --xlog-method="fetch" -P -v --format=t -z
```

- -h: especifica o nome do servidor do banco de dados;
- -U: nome do usuário que irá realizar a conexão;
- -D: diretório onde será gravado o backup. Foram usadas as variáveis %d%m%y para especificar junto ao diretório a data em que está sendo gerado;
- --xlog-method: inclui todos os logs gerados durante o backup;
- -P: mostra o progresso da operação;
- -v: habilita o modo verbose. A intenção é de identificar possíveis problemas durante a operação de backup;
- --format=t: formata a saída do arquivo para o padrão .tar;
- -z: habilita a compressão gzip no arquivo.

Realizando o Restore e o Recovery

Vamos entender como realizar o restore dos backups bases e, logo em seguida, a recuperação aplicando as sequências WAL arquivadas. Lembrando que nesse ambiente usaremos o mesmo servidor em que o backup foi realizado. Se quiser realizar o procedimento em outro servidor, será necessário que o software do PostgreSQL esteja instalado.

Aqui teremos dois cenários de recuperação:

1. No primeiro faremos o restore do backup base e depois a recuperação completa até o momento da falha;
2. No segundo faremos o restore do backup base e depois a recuperação incompleta para um determinado ponto no tempo.

O processo de restauração do backup base é igual para os dois cenários. Usaremos a **Listagem 5** como base para os comandos descritos neste processo. O primeiro passo para realizar o restore é parar todos os processos relacionados ao PostgreSQL (linha 01). Não é obrigatório, mas se for possível, é interessante realizar um backup da atual posição do banco antes que seja realizado o restore. No nosso caso, será criado um diretório

chamado de /bkp_anter_restore onde faremos a cópia de todo o diretório /postgres (linhas 02, 03 e 04).

Feito isso, iremos acessar a localização em que foram realizados os backups, no nosso caso é /backup. Nesse diretório teremos a pasta /archives e uma outra criada com a data da realização do backup. Iremos acessar a pasta 31102015, data em que foi realizado o backup que queremos usar para o restore (linha 05). Dentro dela teremos o arquivo chamado base.tar.gz. O arquivo está nesse formato porque foi compactado durante o backup. Este arquivo deve ser descompactado dentro do diretório base do nosso banco de dados (linha 06).

Listagem 5. Restore do backup base

```
01 pg_ctl stop
02 mkdir -p /bkp_anter_restore
03 chown postgres.postgres /bkp_anter_restore
04 cp -r /postgres/* /bkp_anter_restore/
05 cd /backup/31102015/
06 tar -xvf base.tar.gz -C /postgres/
```

Cenário 1

Já temos nosso backup base restaurado, agora precisamos realizar a recuperação completa do nosso banco de dados. Para isso, precisamos criar o arquivo *recovery.conf*. Este arquivo passa ao PostgreSQL os parâmetros necessários para o recovery. Nesse cenário iremos indicar apenas a localização das sequências arquivadas dos logs de WAL e durante o processo de início do banco de dados serão lidos e aplicados todos os logs de REDO para que o banco seja recuperado até a última sequência gerada. Observe como proceder no código a seguir:

```
-- Recuperação completa
restore_command ='cp /backup/archives/%f %p'
```

Após a leitura e recuperação usando os logs arquivados, o PostgreSQL irá renomear o arquivo *recovery.conf* para *recovery.done* indicando que ele realizou a recuperação conforme solicitado no arquivo.

Cenário 2

A diferença deste cenário é que aqui iremos realizar a recuperação para um determinado período no tempo e não necessariamente até a última sequência arquivada. Assim como no primeiro cenário, após a restauração do backup base, iremos criar o arquivo *recovery.conf*, porém aqui serão necessárias algumas linhas a mais no arquivo, conforme mostrado a seguir:

```
recovery_target_time='2015-10-31 13:30:42'
recovery_target_inclusive = true
restore_command ='cp /backup/archives/%f %p'
```

- *recovery_target_time*: período que queremos recuperar o banco de dados;

- `recovery_target_inclusive`: indica que a recuperação será realizada até o momento descrito no comando. Se a opção escolhida fosse false, então a recuperação iria ocorrer até um período menor do que o indicado no comando;
- `restore_command`: indica o local onde estão os arquivos.

Existem outras configurações durante a recuperação. Recomendamos a leitura da documentação para se aprofundar no assunto (veja seção **Links**).

Autor



Thiago Lima de Castro

thiago_L_C@hotmail.com



Possui 8 anos de experiência na área de banco de dados. Atuou em vários projetos de alta disponibilidade usando tecnologias Oracle em ambiente de missão crítica. Possui certificações Oracle OCP, OCE, ITIL Foundation e IBM DB2. Graduado em Telemática pelo IFCE - Instituto Federal de Educação, Ciência e Tecnologia do Ceará. Atualmente trabalha como DBA alocado no Tribunal de Justiça do Ceará.

Sabendo da importância de ser manter boas estratégias de backup e de recuperação dos dados, a implementação de uma boa estratégia deve ser encarada com seriedade e planejamento. Além da implementação, constantes revisões devem ser realizadas para garantir a segurança dos dados e verificar se o tempo para completar toda a rotina continua atendendo aos requisitos do negócio.

Links:

SQL Dump - Backup lógico

<http://www.postgresql.org/docs/8.3/static/backup-dump.html>

Standby Servers - Banco de dados standby (replicação)

<http://www.postgresql.org/docs/9.4/static/warm-standby.html>

Recovery Target Settings - Configurações usadas durante o recover

<http://www.postgresql.org/docs/9.1/static/recovery-target-settings.html>

Você gostou deste artigo?

Dê seu voto em www.devmedia.com.br/sqlmagazine/feedback

Ajude-nos a manter a qualidade da revista!



Replicando DB relacional na nuvem da Amazon

Uma forma simples de garantir disponibilidade e contingência de dados

A utilização de bases de dados sempre foi de suma importância na maioria dos projetos de software. Antes o analista ou desenvolvedor responsável pela base de dados (DBA) tinha simples escolhas para fazer. Agora ficou complexo, pois existe uma infinidade de banco de dados, e cada um com suas particularidades, diferenças e teorias. No final o objetivo é sempre o mesmo: gravar, ler e manter os dados.

Abordaremos nesse artigo o uso de réplicas para melhorar a performance e disponibilidade do banco de dados. Mas quando falamos de performance, se tratando de banco de dados, o assunto é muito abrangente. Não é simplesmente melhorar “JOINS”. Estamos entrando em um universo que envolve a escolha certa do banco de dados para a aplicação, o local físico onde este banco de dados será instalado e como devemos escrever e acessar estes dados.

A escolha do banco de dados é um assunto muito delicado, por se tratar de uma escolha metade técnica e metade pessoal. A maioria das aplicações de pequenas e médias empresas está bem servida rodando em bancos de dados relacionais como MySQL, SQL Server, Oracle e PostgreSQL. Quando existe uma demanda de *data science*, estas empresas o fazem em paralelo.

A escolha do local físico onde instalar o banco de dados é um tanto quanto óbvia. Provavelmente a primeira coisa que pensamos é colocar este banco de dados em uma máquina dedicada ou na nuvem. Ninguém deveria, nos dias atuais, arriscar deixar a base de dados dentro do ambiente vulnerável da empresa. O local físico onde o banco de dados deve estar tem que ser no mínimo acessível e tolerante a falhas, devendo se reestabelecer automaticamente caso ocorra alguma interrupção.

Fique por dentro

Este artigo mostra como melhorar o desempenho, disponibilidade e segurança em bases de dados relacionais utilizando os recursos do Amazon Web Services (AWS). Vamos tratar da importância de fazer réplicas de leitura mostrando as três maneiras comuns de implementar este recurso através da nuvem da Amazon. Este artigo contém informações valiosas para quem deseja garantir disponibilidade, acessibilidade e segurança para seus dados. Além de disponibilidade, você pode aplicar este conhecimento para obter contingência e performance. Estas informações também servem para quem deseja conhecer as ferramentas do AWS e seus recursos direcionados para bancos relacionais.

Melhorias em consultas SQL é um assunto velho e muito debatido em diversos fóruns. Mas é preciso que o analista ou desenvolvedor tenha em mente uma boa estrutura de dados, que permita consultas inteligentes e rápidas. Nos casos de grandes massas de dados e muitos acessos de leitura, é muito importante entender sobre normalização e desnormalização.

A acessibilidade, disponibilidade e contingência são essenciais em quase todas as aplicações, principalmente nas aplicações móveis e empresas de serviço online. Através do console do AWS, podemos melhorar nossa base de dados com uso de réplicas de leitura para ter mais desempenho, réplica Multi-AZ para maior segurança, ou unindo os dois recursos para obter maior desempenho e segurança. Tudo isto pode ser feito sem entender de código ou ter profundos conhecimentos de banco de dados.

Réplica de banco de dados

Atualmente, é de extrema importância entender e utilizar as réplicas de banco de dados, seja para melhorar a performance

ou para servir de contingência. Com o aumento da mobilidade, as cargas de acesso a bases de dados cresceu exponencialmente. Replicar a base de dados é quase uma obrigação do desenvolvedor, é praticamente parte necessária para garantir maior acessibilidade e contingência ao sistema como um todo, principalmente sistemas críticos.

Quando uma base de dados é replicada, temos a vantagem de direcionar escritas e leituras. Só com esta pequena modificação, provavelmente podemos ganhar 50% de performance, simplesmente porque o banco de dados de escrita está ocupado somente com as escritas, e o banco de dados de leitura está ocupado somente com as leituras.

Quando temos leituras e escritas na mesma base de dados, muitas vezes temos problemas relacionados à memória e I/O. Imagine um usuário fazendo uma consulta na tabela "transações" em busca de todas as transações de uma loja durante um ano, acessando o banco de dados por meio de um sistema. Se esta consulta demora 0,5 segundos, multiplique por 100 usuários e teremos um banco de dados ocioso por 50 segundos. Neste momento, uma escrita certamente estará esperando, pois boa parte da memória está sendo utilizada para a consulta. Este é um exemplo simples, mas na realidade podemos ter problemas bem mais sérios, chegando até a travar a aplicação.

Para contornar o gargalo da memória, muitos profissionais simplesmente aumentam a capacidade das máquinas. Isto resolve em curto prazo, mas certamente chegará o dia em que ele deverá aumentar novamente a memória, até chegar ao limite de memória e CPUs suportados pela máquina. Usando o recurso de réplicas de leitura resolvemos estes problemas muitas vezes com menor custo, pois podemos ter réplicas específicas para um determinado acesso.

Outro benefício da réplica é a disponibilidade, por exemplo, se a base de dados de escrita (Master) cair, o sistema ainda funcionará para as leituras. A equipe terá tempo de ajustar o problema sem o software ficar totalmente indisponível. Assim, também, se a base de dados utilizada para leitura (Slave) cair, o sistema ainda estará disponível para escrita.

Quando utilizamos apenas uma base de dados, na ocorrência de uma queda, o sistema estará totalmente fora do ar. Vale mencionar e lembrá-lo que banco de dados fora do ar significa prejuízo. A empresa perde clientes, contratos, e tem sua reputação manchada. O cliente perde por não conseguir utilizar o recurso que deveria estar disponível. O analista perde, pois sua confiança estará em jogo e provavelmente terá problemas a resolver.

Muitas vezes, por questões financeiras, a réplica fica inviável. Entretanto, é melhor estar preparado, pois geralmente os problemas de banco de dados não acontecem no começo, eles ocorrem justamente no momento em que a empresa mais precisa da acessibilidade, disponibilidade e da segurança.

A nuvem da Amazon e o PostgreSQL

Você provavelmente já deve ter ouvido falar ou já utiliza os recursos de computação em nuvem oferecidos pela Amazon (AWS).

A Amazon criou uma grande quantidade de ferramentas que auxiliam o desenvolvimento back-end e deploy de aplicações, transformando as tarefas de configuração e instalação em rotinas simples e até agradáveis de fazer.

Outro fator importante dentro da estrutura da Amazon é a possibilidade de partitionar os recursos e ter elasticidade, permitindo investir somente o necessário no momento exato da necessidade.

Existem outras empresas que oferecem IaaS (Infraestrutura como serviço) como Microsoft, Rackspace, CenturyLink, Google, IBM, Virtustream, Vwware e outras. Porém, neste artigo vamos tratar apenas do AWS.

Escolhemos o PostgreSQL para os exemplos deste artigo, por ser um banco de dados muito utilizado em pequenas, médias e até grandes corporações. Devido à sua grande quantidade de recursos, este banco de dados atende a muitos tipos de aplicação. O PostgreSQL é *open source*, o que ajuda a reduzir um pouco o custo da infraestrutura. Dentre suas principais características, vale destacar a capacidade de integração com linguagens de programação. Por exemplo, você pode criar dentro do banco rotinas em C, C++, Ruby, Java, dentre outras. É sem sombra de dúvida uma ótima escolha para a maioria dos projetos de software que precisam gravar e ler informações de forma remota com as melhores características de um verdadeiro SGBD.

Apesar de utilizarmos o PostgreSQL como exemplo neste artigo, você pode aplicar a técnica apresentada aos bancos Amazon Aurora, Maria DB, MySQL, Oracle e SQL Server, com pequenas alterações pertinentes a cada SGBD.

Na prática, iremos construir uma base de dados e depois criar uma réplica. Vamos apresentar também outra forma de replicar dentro da Amazon por meio do recurso Multi-AZ.

Preparando o ambiente

Antes de criar nosso banco de dados e replicá-lo, precisamos fazer nosso plano de estrutura. Devemos pensar sobre a real necessidade da base de dados e tentar prever os detalhes, tendo respostas para perguntas como:

- De que forma vamos utilizar nossa base de dados?
- Como ele será acessado?
- Onde ele precisa estar?
- Qual o tamanho necessário para armazenamento?
- Qual a banda de acesso necessária?

A Amazon oferece data centers nas principais regiões do mundo. Isto facilita muito a escolha do local onde será instalado o banco de dados. Você pode escolher colocar a base de dados bem próxima da sua região ou da região onde seus clientes estão. Ter a base de dados próxima ajuda a obter uma melhor latência.

No Brasil, a Amazon possui data centers no Rio de Janeiro e em São Paulo, mas os custos são mais elevados que os data centers nos EUA devido aos impostos brasileiros. Por isso, muitas vezes é preferível perder um pouco em latência e pagar menos utilizando serviços fora do país.

Vamos utilizar duas formas de replicar nossa base de dados. Em seguida, vamos unir as duas formas e ter uma superestrutura de dados acessível e segura. Para simplificar o entendimento, vamos criar apenas uma réplica de banco de dados, sendo que para escalonar basta criar outras réplicas se necessário.

Estas duas formas de fazer réplicas não são as únicas, mas são as mais simples e mais utilizadas pela maioria dos desenvolvedores dentro do ambiente da Amazon. As duas formas de fazer réplicas apresentadas neste artigo são:

- Réplica de leitura: para melhor desempenho;
- Réplica Multi-AZ: para maior segurança.

A primeira forma consiste em criar uma rede, uma sub-rede, o banco de dados **Master** e, em seguida, criar uma réplica de leitura. O ponto positivo desta abordagem é o ganho de performance, pois teremos uma base de dados que podemos utilizar somente para leituras, direcionando as grandes consultas SQL para esta base, deixando o banco de dados principal para receber as escritas. Estas réplicas permitem que você escala para além das limitações de capacidade de uma única instância DB para cargas de trabalho de banco de dados de leitura pesada.

Quando você cria uma réplica de leitura no Amazon RDS (*Relational Database Services*) para MySQL e PostgreSQL, o Amazon RDS configura um canal de comunicação seguro usando criptografia de chave pública entre a instância DB principal e a réplica de leitura.

O ponto negativo é uma pequena perda de desempenho na própria replicação devido à sincronização de dados. Você pode escolher criar a réplica na mesma Amazon Zone (AZ), com isto terá menor latência. Em alguns casos, pode ocorrer um pouco de delay na replicação dos dados, fazendo com que a informação recém salva possa não ser recuperada instantaneamente no banco de dados de leitura. A **Figura 1** mostra um modelo de arquitetura que utiliza a réplica de leitura.

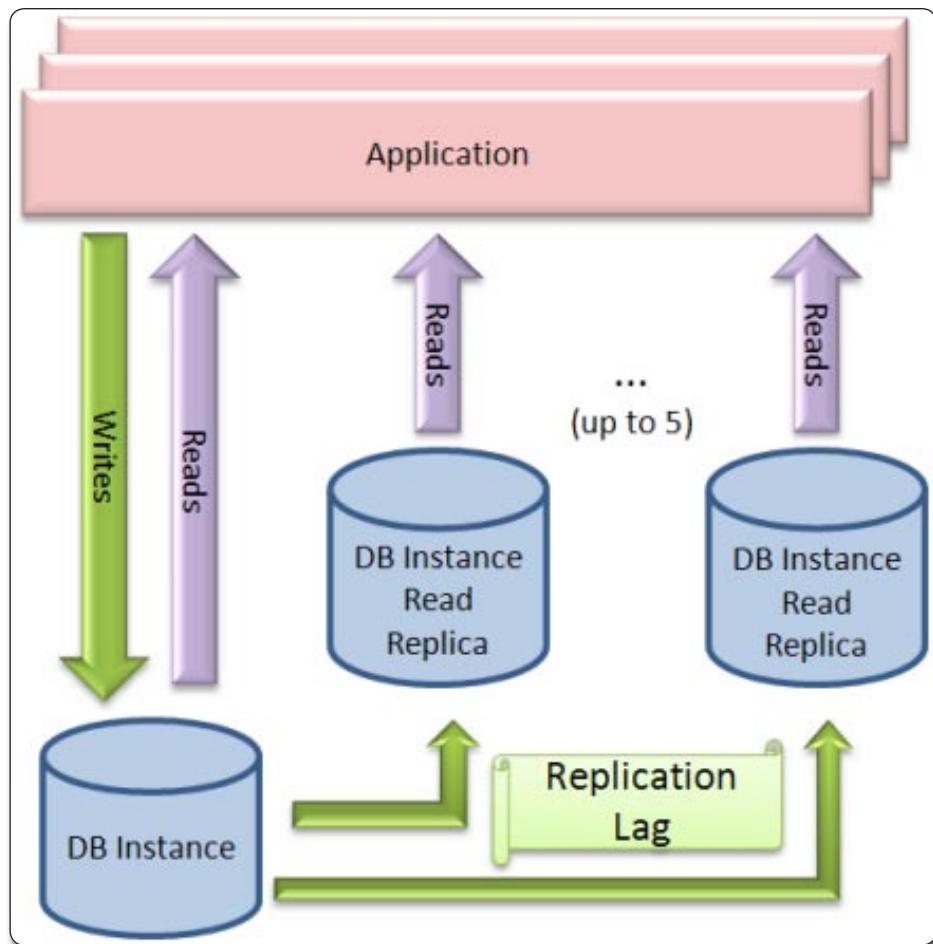


Figura 1. Estrutura de réplica de leitura

A segunda forma consiste em criar a base de dados utilizando o recurso Multi-AZ da Amazon. Com este recurso, a Amazon cria uma réplica (Standby) do banco de dados em uma região da Amazon (Amazon Zone - AZ) diferente e mantém os dados sincronizados oferecendo o recurso de failover automático. Isto significa que se um data center falhar, o outro assume automaticamente e assim, quando o banco de dados que falhou voltar a funcionar, os dados serão sincronizados.

A principal vantagem é que, em ambiente de produção, este método pode simplesmente salvar o negócio da empresa, pois dificilmente dois data centers teriam seus serviços interrompidos.

A desvantagem é que embora tenhamos duas bases de dados, as leituras e escritas são direcionadas sempre para o mesmo endpoint, ou seja, se precisar separar leituras de escritas, será necessário criar

mais uma réplica de leitura. Isto pode encarecer a solução, mas é uma ótima opção para situações críticas e de baixa tolerância a falhas. No fim do artigo iremos unir a réplica Multi-AZ com a réplica de leitura e ter uma excelente configuração. A arquitetura do recurso Multi-AZ pode ser observada na **Figura 2**.

A estrutura Multi-AZ é altamente recomendada pela Amazon para ambientes de produção devido às melhorias de segurança da aplicação oferecidas por este modelo. Ela oferece maior disponibilidade e durabilidade para as instâncias de banco de dados. Em caso de uma falha de infraestrutura (por exemplo, falha de hardware, armazenamento ou de rede), o Amazon RDS executa um failover automático para a instância standby. Desta forma, ela pode retomar as operações assim que o failover completar (geralmente de 1 a 6 minutos). Como o endpoint permanece o mesmo

Replicando DB relacional na nuvem da Amazon

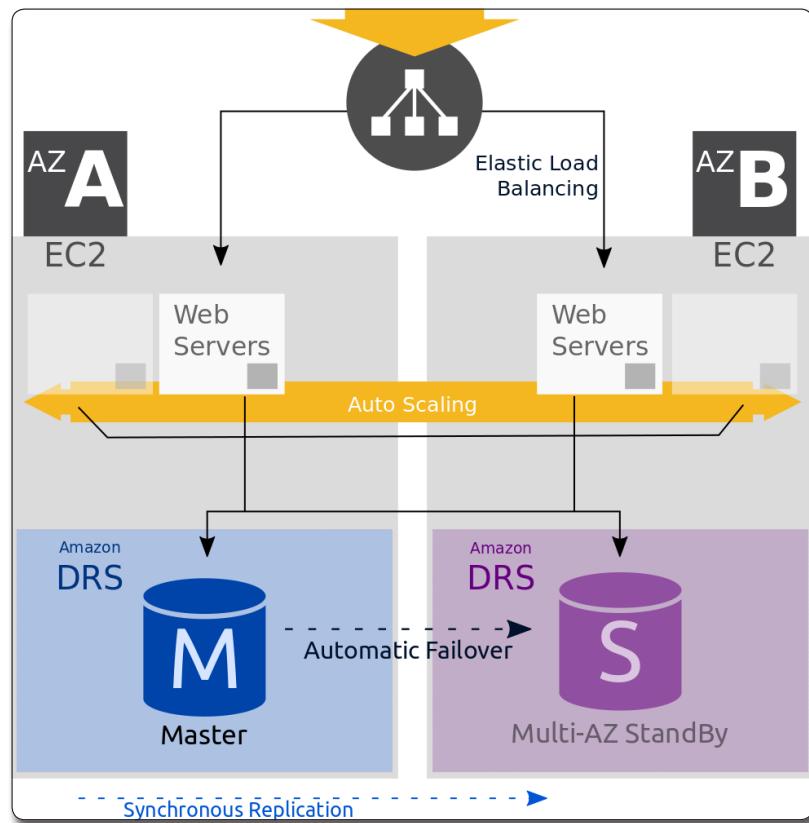


Figura 2. Arquitetura Multi-AZ

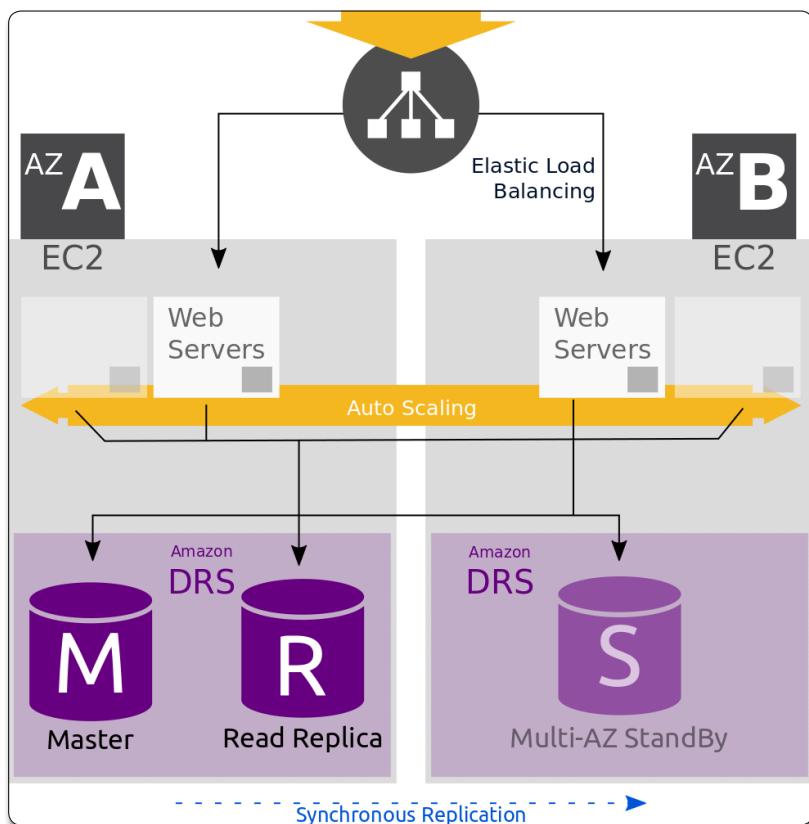


Figura 3. Arquitetura Multi-AZ com Réplica de Leitura

depois de um failover, as operações de banco de dados são retomadas automaticamente sem nenhuma necessidade de intervenção manual.

No serviço RDS, quando utilizado sem a opção Multi-AZ, havendo falha de banco de dados, uma restauração manual backup point-in-time será necessária. Dependendo do tamanho do seu banco de dados, esta operação poderá levar muitas horas para ser concluída, deixando sua aplicação indisponível até que o processo esteja concluído. Contudo, ainda se pode correr o risco de não conseguir restaurar todos os backups.

Por fim, vamos unir a réplica Multi-AZ e a réplica de leitura para obter o melhor desempenho e maior segurança. Sem dúvida, um dos recursos mais interessantes oferecidos pela Amazon RDS, pois de forma muito simples podemos juntar os dois modelos de replicação e garantir estabilidade em sistemas de grande concorrência. Veja na Figura 3 como fica a arquitetura ao unir réplica Multi-AZ e réplica de leitura.

Réplica de leitura para melhor desempenho

Iremos criar uma base de dados master e uma slave dentro da mesma estrutura, mas antes de qualquer coisa, você precisa ter uma conta no AWS. Se você não tem, você pode se cadastrar de forma muito rápida. A Amazon oferece um plano "free" para pessoas que desejam testar as ferramentas do AWS.

Depois de abrir uma conta na Amazon, você terá acesso ao Console Amazon AWS. Ele pode ser acessado através do navegador web e também como ferramenta que pode ser utilizada por meio de linha de comando, através de um terminal no computador. Você pode executar quase todas as funções do console através de linha de comando.

No Console Amazon AWS você tem acesso a todas as ferramentas disponíveis. Seguiremos escolhendo RDS (*Amazon Relational Database Service*). Ao fazer isto, teremos acesso ao painel do RDS com todas as funcionalidades que precisamos para lidar com nossos bancos de dados relacionais.

O RDS é uma ferramenta que permite a instalação, manutenção e escala de banco de dados relacionais. A proposta da empresa é facilitar estas ações e deixar o cliente com tempo maior para sua aplicação.

Escolhido o RDS, clique no menu *Instances* e escolha a opção *Launch DB Instance* dentro do RDS Dashboard. Em seguida, escolha o banco de dados PostgreSQL. Siga as instruções até chegar na tela onde você deve escolher se vai utilizar o recurso Multi-AZ ou não (ver Figura 4).

Escolhemos a opção “No, this instance...” nesta tela porque estamos fazendo a réplica de leitura apenas. Nossa objetivo agora é primeiro criar nossa instância do banco e depois criar a réplica de leitura.

Ao clicar no botão *Next Step* vamos ver outra tela onde iremos definir os detalhes do nosso banco de dados. Por este motivo, é preciso que você já tenha feito uma análise prévia da sua demanda de dados, acesso e tamanho da base. A ferramenta RDS é muito flexível, você pode alterar alguns parâmetros depois, como tamanho da base de dados, mas você não pode alterar a memória da máquina e a quantidade de processadores de forma muito simples. Ou seja, você não deve errar o tipo da máquina que irá utilizar, pois caso pegue uma máquina muito pequena em memória, e seu projeto precise de muito processamento, você certamente terá problemas de desempenho.

Mesmo após ter escolhido não utilizar Multi-AZ, o painel retorna a questionar se faremos uso de “Multi-AZ Deployment”. Escolha *No* para esta opção, pois já definimos que este banco de dados não utilizará Multi-AZ. Na opção *Storage Type* você pode escolher *Magnetic* ou *SSD*, a grande diferença entre os dois é o ganho de performance em I/O e o custo. O armazenamento em discos magnéticos é bem mais barato que em SSD. Ao escolher armazenamento SSD você precisará informar a quantidade de IOPS que deseja. Praticamente quanto mais IOPS, mais velocidade você terá. IOPS são operações de entrada/saída por segundo. São medidas utilizadas pela Amazon EBS. Cada operação de I/O por segundo (que é 256 KiB ou menor) equivale a um IOPS. Operações I/O que são maiores do que 256 KiB são contados em 256 unidades de capacidade KiB. Por exemplo, uma única operação de 1024 KiB I/O contaria como 4 IOPS. Operações de I/O 1024 em 1 KiB cada, contaria como 1.024 IOPS.

A Figura 5 mostra as definições do banco de dados que estamos utilizando neste exemplo. Escolhemos a menor instância oferecida pela Amazon para banco de dados.

Após definir estas configurações básicas e clicar em *Next Step* o painel irá abrir uma

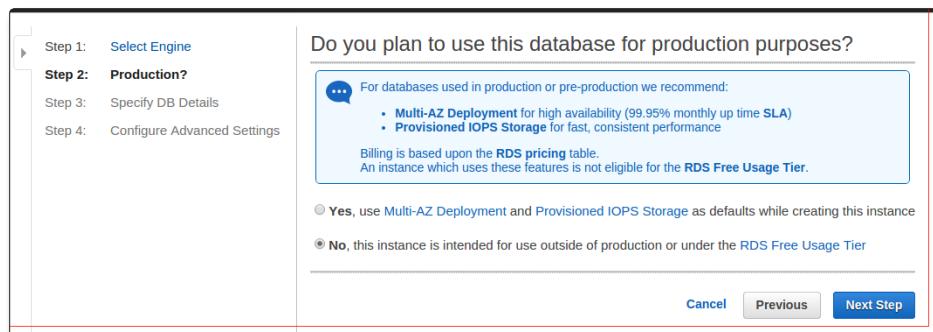


Figura 4. Escolha entre Multi-AZ ou Single-AZ

Figura 5. Configuração do banco de dados

nova página que solicitará as definições mais avançadas para a sua estrutura de dados. Você deve configurar as opções de rede, segurança, banco de dados, backup e manutenção. Praticamente toda a sua infraestrutura. Este ponto é sem dúvida o mais importante de todos, errar aqui pode ser muito prejudicial para a sua arquitetura.

Vamos explicar a seguir as decisões mais importantes que você precisa fazer nesta tela.

- Defina sua **VPC (Virtual Private Cloud)**, uma espécie de rede privada que você possui dentro da Amazon. É preciso criar ou escolher um ambiente de rede virtual para instalar sua instância DB. Caso você não tenha criado uma VPC na Amazon, esta é uma das primeiras coisas que você deve fazer. Se não estiver familiarizado com isto, escolha a opção *Create*

new VPC e deixe que a ferramenta crie uma para você;

- Defina o **Subnet Group**, trata-se da sua sub-rede. Assim como você deixou o sistema criar uma VPC, você também pode escolher deixar o sistema criar uma sub-rede. Ela estará contida em sua rede maior que é a VPC;
- Na opção **Publicly Accessible** você deve escolher *Yes* se quiser permitir o acesso externo à VPC onde estará hospedado o seu banco de dados. Se você selecionar *No*, o Amazon RDS não irá atribuir um endereço IP público para a instância DB, e nenhuma instância EC2 (*Elastic Compute Cloud*) ou dispositivos fora da VPC será capaz de conectar à sua base. Se você selecionar *Yes*, você também deve selecionar um ou mais grupos de segurança do VPC que especificam quais EC2, instâncias e dispositivos podem se conectar à sua instância

RePLICANDO DB RELACIONAL NA NUVEM DA AMAZON

de banco de dados. Amazon EC2 fornece capacidade de computação escalável na nuvem da Amazon. O EC2 elimina sua necessidade de investir em hardware no futuro para que você possa desenvolver e implantar aplicativos mais rápidos. Você pode usar Amazon EC2 para criar a quantidade de servidores virtuais que você precisar, configurar a segurança, rede e gerenciar o armazenamento. Amazon EC2 permite que você dimensione para cima ou para baixo suas instâncias para lidar com mudanças nos requisitos ou picos de popularidade, reduzindo a necessidade de prever o tráfego;

- **Availability Zone:** simplesmente escolha uma zona (região) para instalar sua infra;
- **VPC Security Group(s):** você deve selecionar o grupo de segurança ou grupos

que têm regras que autorizam conexões de todas as instâncias de EC2 e dispositivos que precisam acessar os dados armazenados na sua instância. Por padrão, os grupos de segurança não autorizam quaisquer ligações, você deve especificar regras para todas as instâncias e dispositivos que se conectam a ela. Ou seja, sua VPC está bloqueada por padrão para a maioria dos acessos, então você deve criar um grupo de segurança que permita o acesso à sua base de dados. Para isso, você deve deixar selecionada a opção *Create New Security Group*, ou escolher algum grupo de segurança já criado;

- **Database Name:** defina novamente o nome do banco de dados;
- **Database Port:** a porta padrão é 5432, mas você pode modificar e colocar uma porta de seu interesse;

- Deixe as opções **DB Parameter Group**, **Option Group**, **Copy Tags To Snapshots** e **Enable Encryption** definidas como estão, com seus valores padrões;
- Defina suas configurações de backups. Em **Backup Retention Period** selecione o período em que será feito seu backup. Na opção **Backup Window** escolha *No preferences* ou escolha *Select Window* e defina o período da janela;
- Em **Auto Minor Version Upgrade**, especifique *Yes* para ativar as atualizações automáticas para novas versões quando elas forem liberadas. Os upgrades automáticos ocorrem durante a janela de manutenção para a instância;
- A **Maintenance Window** funciona como a **Backup Window**. Escolha *No preferences* ou escolha "Select Window" e defina o período da janela de manutenção.

The screenshot shows the 'Launch DB Instance' configuration wizard. It includes sections for Subnet Group (Create new DB Subnet Group), Publicly Accessible (No), Availability Zone (No Preference), and VPC Security Group(s) (Create new Security Group). In the 'Database Options' section, the Database Name is set to 'appmaster', Database Port to '5432', DB Parameter Group to 'default.postgres9.4', Option Group to 'default:postgres-9-4', and Enable Encryption to 'No'. A note indicates that storage encryption is not supported. The 'Maintenance' section shows Auto Minor Version Upgrade set to 'Yes' and Maintenance Window set to 'No Preference'. At the bottom, there are buttons for 'Cancel', 'Previous', and 'Launch DB Instance'.

Figura 6. Configuração do banco de dados

Veja como ficou a configuração de exemplo na **Figura 6**.

Ao clicar em *Launch DB Instance* o Console irá criar todas as configurações necessárias para instalar sua base de dados e, por fim, criará sua base. Uma mensagem de sucesso aparecerá na próxima tela, e uma opção para você visualizar suas instâncias. Você pode observar sua instância com o banco de dados PostgreSQL clicando no item *Instâncias* no menu do RDS Dashboard.

Até aqui você conseguiu criar sua base de dados PostgreSQL na nuvem da Amazon. Mas ainda falta criar a sua réplica para leituras. Afinal, este é nosso foco desde o começo deste artigo.

Agora que você já tem a rede montada e sua instância com o PostgreSQL que será nossa base de dados Master, vamos selecioná-la e no botão *Instance Action* vamos escolher a opção *Create Read Replica*. Ao criar uma réplica de leitura, ela herdará todas as configurações da instância Master. A única configuração necessária nesta etapa é definir o nome da réplica de leitura. Se sua aplicação precisar, você pode definir o parâmetro *Publicly Accessible* para *Yes*, assim você pode acessar a base de dados estando fora da rede privada criada anteriormente. A **Figura 7** mostra como ficou a nossa configuração.

Concluindo esta etapa, levará apenas alguns minutos para que a Amazon crie a réplica de leitura e configure a base de dados master. Você terá uma base de dados PostgreSQL master pronta para receber escritas e leituras e outra base de dados PostgreSQL slave, pronta para ser utilizada somente como leitura.

Precisamos agora preparar a aplicação para trabalhar com as duas bases de dados distintas, uma para escrita e outra para leitura. Se você já tem uma aplicação em produção e ainda não está utilizando esta técnica de réplica, faça a experiência utilizando um backup de seus dados e criando uma estrutura conforme fizemos aqui, observando obviamente uma configuração de memória e CPU mais apropriada para sua aplicação. Os benefícios em ganho de performance são notáveis já no primeiro teste.

Para configurar o endereço do banco de dados em sua aplicação, na definição do host, entre com o endereço definido como endpoint na sua instância DB. Veja na **Figura 8** como ficou a instância DB master e a slave.

Para acessar as bases de dados, além de configurar o host com o endpoint correto, você também precisa criar grupos e contas de usuários dentro da sua VPC. Para isto, vá até o menu superior do Amazon AWS Console e clique sobre o seu nome, escolhendo o submenu *Security Credentials*. Com as informações do próprio painel você será capaz de criar e configurar contas de usuários para acessar suas bases de dados. Lembre-se que

Instance Specifications

DB Instance Class: db.t2.micro — 1 vCPU, 1 GiB RAM ▾

Storage Type: Magnetic ▾

Settings

Read Replica Source: appmaster

DB Instance Identifier*: appslave1

Network & Security

This instance will be created with the new Certificate Authority rds-ca-2015. If you are using SSL to connect to this instance, you should use the [new certificate bundle](#). Learn more [here](#)

Publicly Accessible: Yes ▾

Availability Zone: No Preference ▾

Database Options

Database Port: 5432

Copy Tags To Snapshots:

Maintenance

Auto Minor Version Upgrade: Yes ▾

Figura 7. Configuração da réplica de leitura



Replicando DB relacional na nuvem da Amazon

estes usuários serão superusuários em suas bases. Então, depois que criar um usuário, acesse sua base de dados PostgreSQL e crie os usuários e bancos de dados com permissões específicas. Nunca utilize o superusuário para acessar sua base através da aplicação.

Réplica Multi-AZ para maior segurança

O principal motivo de utilizar o recurso Multi-AZ da Amazon é garantir maior segurança para nossa instância. Este recurso é muito simples de ser implantado e os benefícios são enormes, conforme vamos mostrar na sequência.

O recurso Multi-AZ não é barato, ele simplesmente duplica o seu custo com instâncias da Amazon, e seu maior benefício somente será perceptível no caso de failover, ou seja, somente se a sua instância de banco de dados master sofrer uma interrupção. Neste caso, a instância com a réplica será ativada, assumindo o trabalho da instância master até que ela volte a funcionar. Esse recurso é muito recomendado para sistemas críticos e de alta disponibilidade.

O RDS Multi-AZ para o MySQL, Oracle e PostgreSQL utiliza a replicação física e síncrona para manter os dados da instância standby atualizados e idênticos aos dados da instância primária. Já para o Microsoft SQL Server é feita replicação lógica e síncrona com o recurso nativo de Mirroring desse SGBD para obter o mesmo resultado.

Para utilizar este tipo de réplica, também é importante saber em quais casos ocorrerá um failover automático, o que pode ser nas seguintes situações:

- Falha na zona de disponibilidade da instância primária;
- Conexão de rede perdida com a instância primária;
- Falha no CPU da instância primária;
- Falha de armazenamento na instância primária.

O processo para criar a infraestrutura utilizando o recurso Multi-AZ é semelhante ao descrito em “Réplica de leitura, para melhor desempenho”. Acesse o Console do Amazon AWS, escolha a ferramenta RDS, ao abrir a tela com o RDS Dashboard escolha

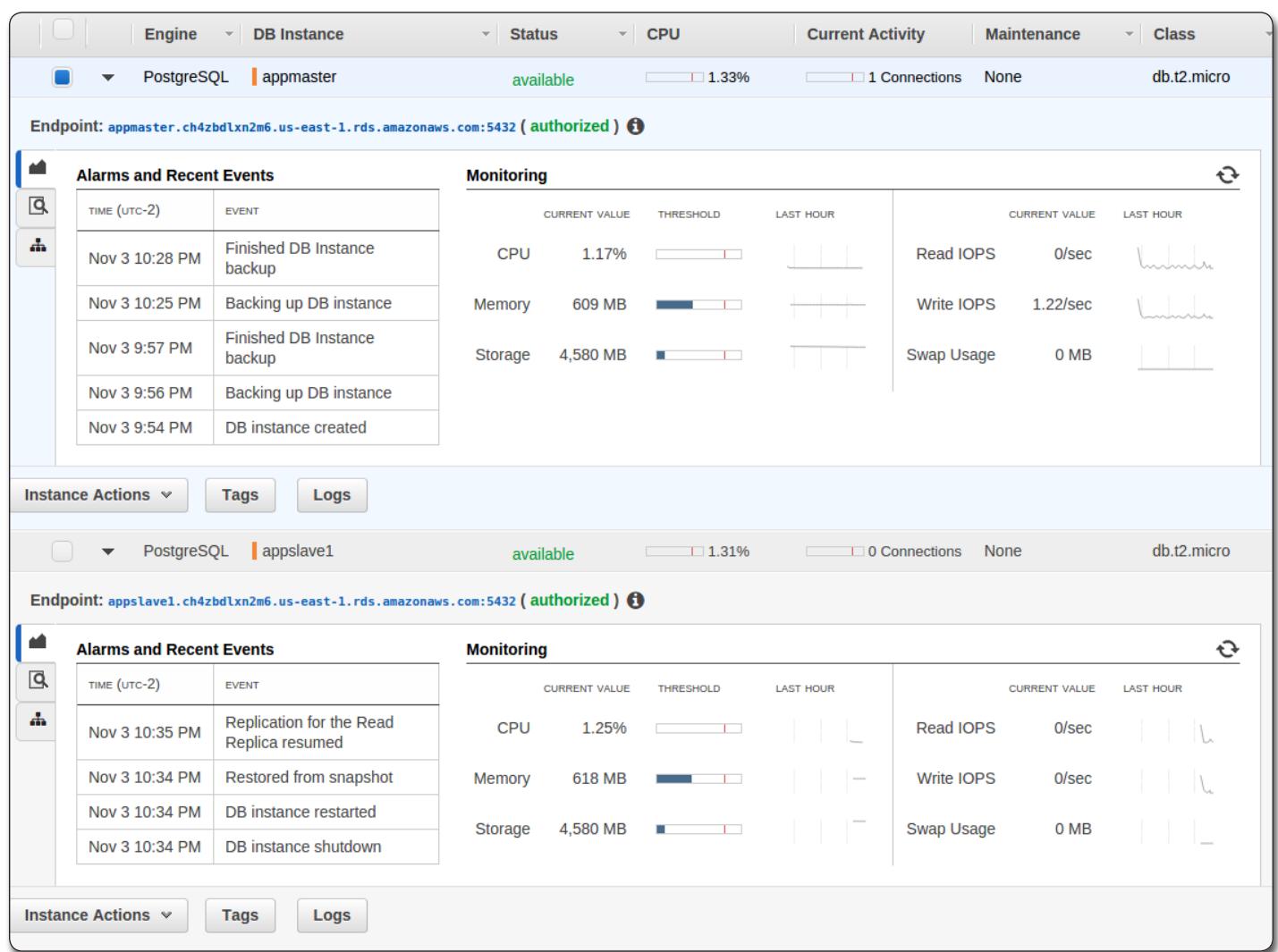


Figura 8. Configuração da réplica de leitura

Step 1: Select Engine

Step 2: Production?

Step 3: Specify DB Details

Step 4: Configure Advanced Settings

Info The following selections disqualify the instance from being eligible for the free tier:

- Multi-AZ Deployment

[Learn More .](#)

Specify DB Details

Instance Specifications

DB Engine	postgres
License Model	postgresql-license
DB Engine Version	9.4.4
DB Instance Class	db.t2.micro — 1 vCPU, 1 GiB RAM
Multi-AZ Deployment	Yes
Storage Type	Magnetic
Allocated Storage*	5 GB

Settings

DB Instance Identifier*	appmultiaz
Master Username*	appmultiaz
Master Password*
Confirm Password*

* Required

[Cancel](#) [Previous](#) [Next Step](#)

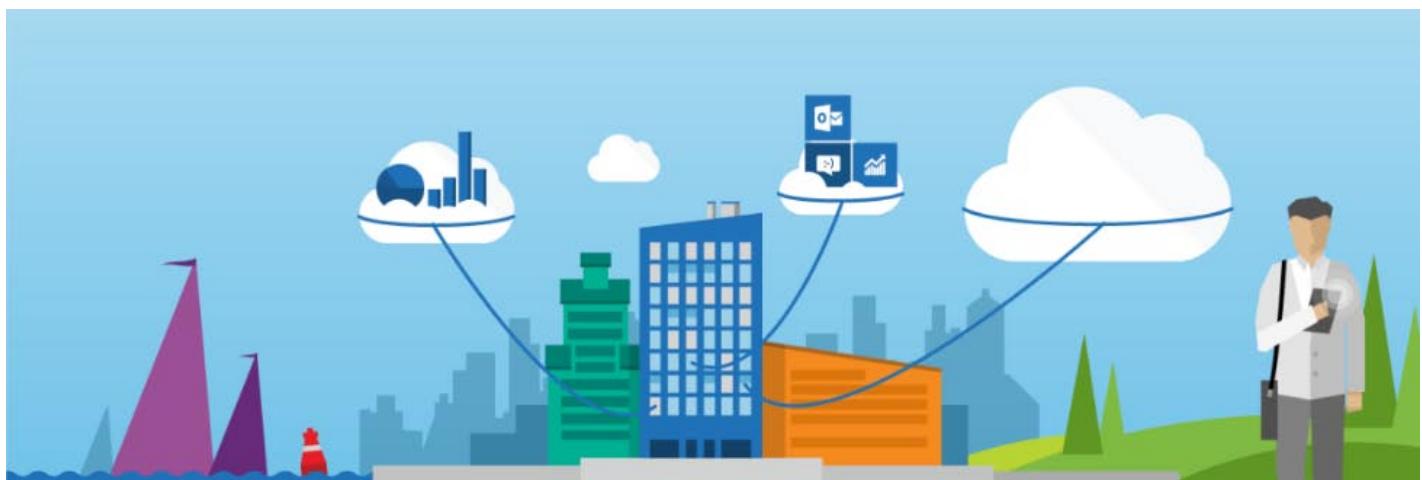
Figura 9. Configuração da instância utilizando Multi-AZ

no menu a opção *Instances*, depois clique sobre o botão *Launch DB Instance* e, em seguida, escolha o banco de dados PostgreSQL. Siga as instruções até chegar na tela onde você deve escolher se vai utilizar o recurso Multi-AZ ou não (**Figura 4**).

Desta vez você vai escolher a opção “*YES, use Multi-AZ Deployment...*”. Ao clicar em *Next Step*, a nova tela irá pedir para você informar as configurações básicas da sua nova instância. Escolha a versão do PostgreSQL, o tipo da instância em *DB Instance Class* e selecione *YES* para a opção *Multi-AZ Deployment*.

Nas configurações de *Storage Type*, proceda da mesma forma que fizemos na configuração anterior em “Réplica de leitura, para melhor desempenho”. A **Figura 9** apresenta como ficou a configuração da nossa instância DB.

Ao clicar em *Next Step*, as configurações avançadas da instância de banco de dados serão apresentadas. Note que esta tela é exatamente a mesma que apareceu quando estávamos criando uma instância de banco de dados utilizando *Single AZ* (sem o recurso Multi-AZ).



Replicando DB relacional na nuvem da Amazon

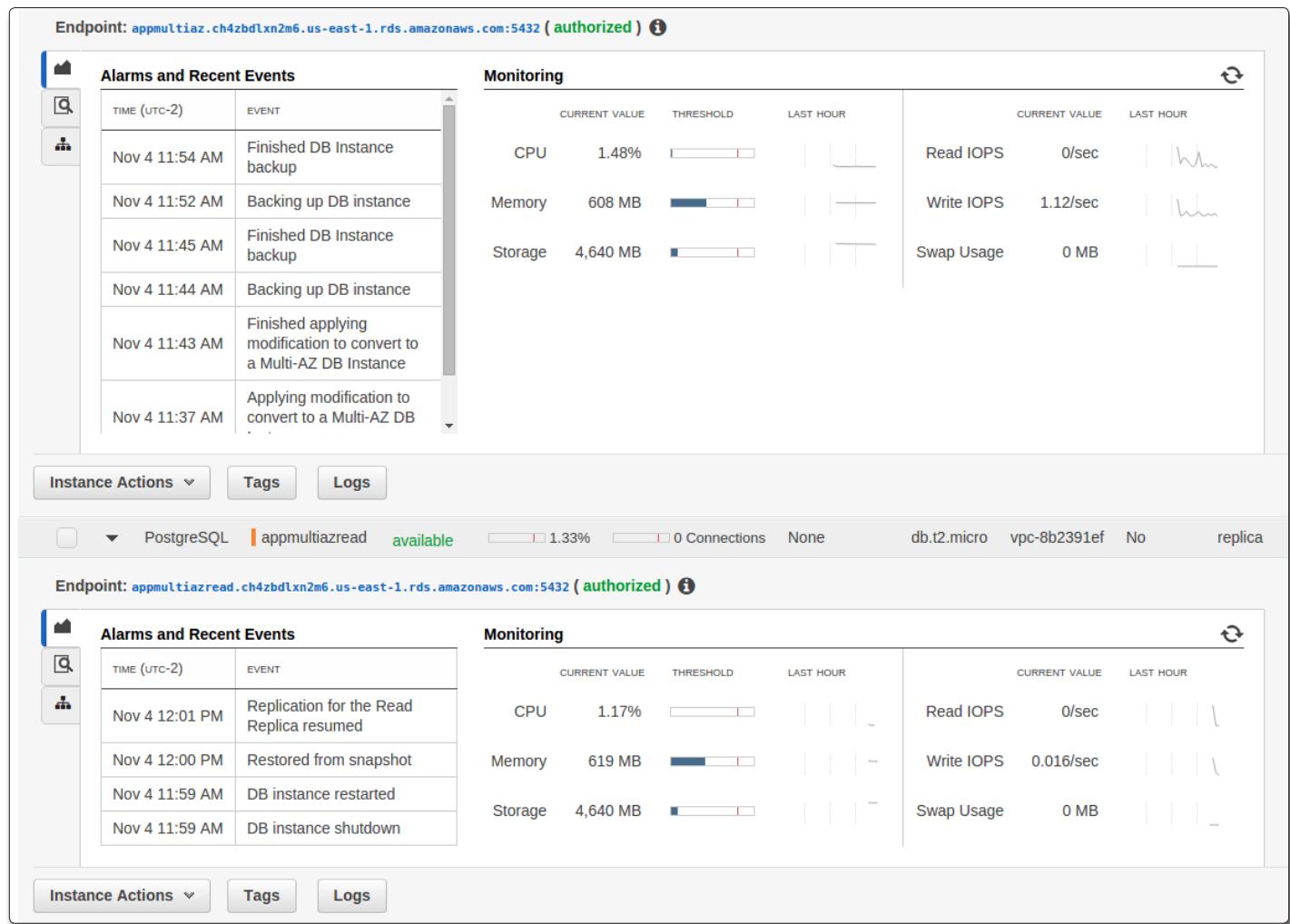


Figura 10. Instância Multi-AZ e sua Réplica de Leitura

Você pode ver a representação desta tela na [Figura 6](#), onde a única modificação que fizemos foi inserir um nome para nosso banco de dados.

Ao clicar em *Launch DB Instance* o console irá criar todas as configurações necessárias para instalar sua base de dados e, por fim, criará o banco de dados. Uma mensagem de sucesso aparecerá na próxima tela e uma opção para você visualizar suas instâncias. Você pode observar sua instância com o banco de dados PostgreSQL clicando no item *Instâncias* no menu do RDS Dashboard.

Provavelmente a sua instância DB ainda não estará pronta. É preciso aguardar alguns minutos até que o Console do Amazon AWS termine todas as configurações necessárias para disponibilizá-la.

Assim que concluir, você terá uma instância DB com Multi-AZ configurada. Para utilizá-la em sua aplicação, você deve definir o nome do host com o endpoint da sua instância. Agora temos uma base de dados para escrita e leitura, e uma réplica em standby que será ativada quando a instância primária tiver problemas, garantindo disponibilidade para a aplicação mesmo quando

houver uma falha. Vale ressaltar que um failover automático para a instância standby pode demorar (geralmente de 1 a 6 minutos) para retomar as operações de banco de dados.

Multi-AZ com réplica de leitura para maior desempenho e segurança

Até este ponto falamos sobre a “Réplica de Leitura” e a “Réplica utilizando Multi-AZ”. Agora que conhecemos as duas e sabemos como aplicá-las, podemos fazer uso de ambas em conjunto e oferecer maior desempenho e maior segurança para nossa aplicação.

Unir estes dois pontos é muito simples. Primeiro crie sua instância DB utilizando Multi-AZ, como fizemos na seção “Réplica Multi-AZ, para maior segurança”. Feito isto, vá até o painel RDS Dashboard e clique sobre o menu *Instâncias*. Aparecerão suas instâncias DB, então selecione a instância DB Multi-AZ que foi criada e, no botão *Instance Action* escolha a opção *Create Read Replica*. O console irá criar uma réplica de leitura do banco de dados principal.

Ao criar a réplica, ela herdará todas as configurações da instância DB principal. A única configuração necessária nesta etapa foi definir o nome da réplica de leitura. Veja na **Figura 7** um exemplo desta tela.

Neste modelo, a aplicação é configurada para escrever na base de dados principal e ler na base de dados de leitura. Caso ocorra uma falha de infraestrutura (falha de hardware de instância, falha de armazenamento ou interrupção da rede), o Amazon RDS executa um failover automático para a instância de espera, permitindo retomar as operações de banco de dados assim que o failover é concluído. Como o endpoint para sua instância permanece o mesmo após um failover, seu aplicativo pode retomar as operações sem a necessidade de intervenção administrativa manual.

A **Figura 10** apresenta as duas instâncias de banco de dados, uma acima com Multi-AZ configurado e outra a seguir com a réplica de leitura.

Para escalar horizontalmente seu banco, basta criar mais réplicas de leitura. Se o problema for escrita, escolha uma instância com maior capacidade de processamento e armazenamento.

Lidar com banco de dados é muito simples com os recursos do AWS. Aqui utilizamos o PostgreSQL, mas você pode utilizar estes mesmos procedimentos com outros bancos de dados relacionais.

Com os recursos computacionais que temos hoje, como os que foram mostrados neste artigo, ficou muito mais simples fazer a gestão e configuração de ambientes complexos ligados a bancos relacionais. Com tranquilidade e poucos minutos criamos réplica para leitura para uma melhor disponibilidade e réplica (standby) para funcionar em casos de falha, servindo como contingência. Não foi preciso utilizar nenhuma linha de código e ainda conseguimos unir desempenho e tolerância a falhas.

Autor



Antonio Marcos Ferreira

hipertrix@gmail.com

Graduado em Sistemas Para Internet pela Faculdade Metrocamp IBTA em Campinas. Co-fundador da empresa W3da, onde desempenha o papel de Gestão de Negócios, BI e P&D. Com mais de 15 anos de experiência em sistemas para internet, sendo destes 6 anos focados em sistemas de gestão de cartões de crédito e débito.



Links:

O que são IOPS?

<http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-io-characteristics.html>

Sobre a Amazon Aws

<https://aws.amazon.com/pt/enterprise/>

O que são EC2?

<http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html>

Implantações Multi-AZ do Amazon RDS

<https://aws.amazon.com/pt/rds/details/multi-az/>

Amazon RDS: Announcing Read Replicas

<https://aws.amazon.com/pt/blogs/aws/amazon-rds-announcing-read-replicas/>

Você gostou deste artigo?

Dê seu voto em www.devmedia.com.br/sqlmagazine/feedback

Ajude-nos a manter a qualidade da revista!



Mineração de texto: Análise comparativa de algoritmos

Avaliação de acurácia, cobertura, precisão e tempo de execução

Grandes massas de dados são geradas diariamente pelos sistemas que apoiam as atividades rotineiras das organizações, dificultando a tarefa analítica dos gestores. Diante dessa necessidade, surgiram os Sistemas de Apoio à Decisão (SADs) que permitem apoiar, contribuir e influenciar no processo de tomada de decisão. Os SADs permitem, a partir dos dados transacionais da organização, gerar informações gerenciais que facilitam o referido processo.

Como grande parte dos dados manipulados pelas organizações está em formato textual, torna-se fundamental o uso da técnica de mineração de texto (também conhecido por *Knowledge Discovery in Texts, KDT*) para identificar padrões e conhecimentos para auxiliar nas decisões.

O conhecimento gerado pode ser avaliado para determinar se o mesmo é relevante ou não para o usuário, ou seja, avaliar o desempenho do processo de mineração para a geração do conhecimento. Existem várias métricas, sendo as principais relacionadas ao desempenho, à acurácia, precisão e cobertura.

Neste artigo iremos apresentar um estudo de caso realizado em uma organização ABC. A ACB lida com um imenso volume de informações, sendo necessária a utilização de mecanismos que tornem efetivas as atividades de auditoria. Auditoria é a atividade que realiza a validação das informações, verificação da obediência às normas e recomendações e avaliações dos controles em busca dos resultados da gestão. Objetivando atender as necessidades da ABC, desenvolvemos uma aplicação que realiza a mineração de texto em qualquer campo descritivo de um sistema, a ferramenta TextMining.

A aplicação permite determinar se uma descrição é ou não evidência de irregularidade, tornando efetivo o

Fique por dentro

O uso de mineração de texto é importante para o processo de extração de conhecimento em bases textuais. Contudo, é importante avaliar se o conhecimento extraído ou gerado é relevante ou não para o usuário. Neste artigo avaliamos a performance de algoritmos de mineração de texto da ferramenta TextMining. A discussão apresentada neste artigo é útil pois é cada vez mais comum a necessidade de descobrirmos informação útil a partir de dados textuais. Através do estudo de caso apresentado será possível entender o impacto que diferentes algoritmos de mineração de texto trazem a esta atividade. Esta informação poderá apoiar a tomada de decisão sobre qual estratégia de mineração utilizar.

trabalho do auditor na identificação de irregularidades. Para classificar uma descrição, a ferramenta dispõe de um algoritmo, Naïve Bayes, de forma parametrizada, especificando um limiar mínimo para auxiliar no processo classificatório. É importante destacar que existem três métodos para o Naïve Bayes: "Híbrido" (utilização da frequência do termo da amostra com *tf, term frequency*, da sentença), "Frequência Inversa" (*tfidf, term frequency – inverse document frequency*, da amostra com *tf* da sentença) e "Frequência" (frequência da amostra com frequência da sentença).

Este trabalho introduziu um segundo algoritmo, Similaridade, na ferramenta citada e foram avaliadas as métricas de qualidade e desempenho para as duas abordagens. A avaliação se deu por meio da coleta de métricas de tempo médio, acurácia, cobertura, medida F e precisão de cada algoritmo.

Assim, este artigo objetiva comparar o desempenho e qualidade de dois algoritmos de mineração de texto aplicados a históricos de contas públicas custodiadas pela organização ABC. A análise comparativa determinará o melhor algoritmo da ferramenta

TextMining e, consequentemente, o conhecimento gerado por essa abordagem será efetivo e relevante para os auditores na descoberta de irregularidades como a identificação de uma descrição de motivo de viagem para a qual não é permitida o pagamento de diárias.

Descoberta de Conhecimento em Bases de Dados

KDD (*Knowledge Discovery in Databases*) é o processo não-trivial de identificar padrões válidos, novos, potencialmente úteis em dados, ou seja, é o processo de descoberta de conhecimento ou padrões úteis e desconhecidos em grandes massas de dados.

O processo de KDD consiste de várias etapas, as quais envolvem preparação dos dados, busca por padrões, avaliação do conhecimento e refinamento, todos repetidos em múltiplas iterações. Esse processo é composto por cinco passos bem definidos: seleção, pré-processamento, transformação, mineração de dados, análise / assimilação, conforme é mostrado na **Figura 1**.

Na etapa seleção serão definidas as fontes de dados relevantes, ou seja, as bases de dados importantes para o problema em questão, o qual se deseja resolver. No pré-processamento, os dados serão tratados, pois como esses dados podem ser oriundos de diversas fontes, os mesmos podem conter divergência de valores e outras inconsistências. Na transformação, os dados pré-processados serão convertidos para uma estrutura compatível com o algoritmo de mineração escolhido. Já na etapa mineração de dados, objetivo do processo de KDD, é escolhida e executada uma técnica e algoritmo de mineração de acordo com o problema em questão, por

exemplo, classificação, regressão, agrupamento e summarização. E, por fim, na etapa de análise/assimilação, o conhecimento gerado será avaliado se é útil ou não para a tomada de decisão.

Como é mostrado na **Figura 1**, o processo de KDD é um processo iterativo e interativo, em que o usuário participa e realiza decisões nas diversas etapas do processo, as quais podem também ser repetidas, dependendo do conhecimento gerado ou pela ausência do mesmo.

O processo de KDD pode ser aplicado em diversas áreas, incluindo marketing, finanças, detecção de fraudes, manufaturas e telecomunicações. Um exemplo clássico de utilização de KDD é o conhecimento descoberto nos dados da rede de supermercados Walmart. Foi descoberto que a maioria dos pais que iam comprar fraldas para seus filhos acabavam comprando cerveja. Em uma jogada de marketing, as fraldas foram colocadas próximas da cerveja, sendo que as batatas fritas estavam entre elas. Consequentemente, houve um aumento das vendas dos três produtos.

Outro exemplo de utilização do processo de KDD foi o uso do sistema ADVANCED SCOUT da IBM para ajudar os treinadores da NBA, no ano de 1996, a procurar e descobrir padrões interessantes nos dados dos jogos da NBA. Com esse conhecimento obtido, os treinadores podiam avaliar a eficácia das decisões de táticas e formular estratégias de jogo para jogos futuros. O sistema foi distribuído para dezesseis das vinte e nove equipes da NBA, sendo usado de forma efetiva por algumas equipes para a preparação de jogadas e processos analíticos, como foi o caso do time Seattle Supersonics, o qual atingiu as finais da NBA.

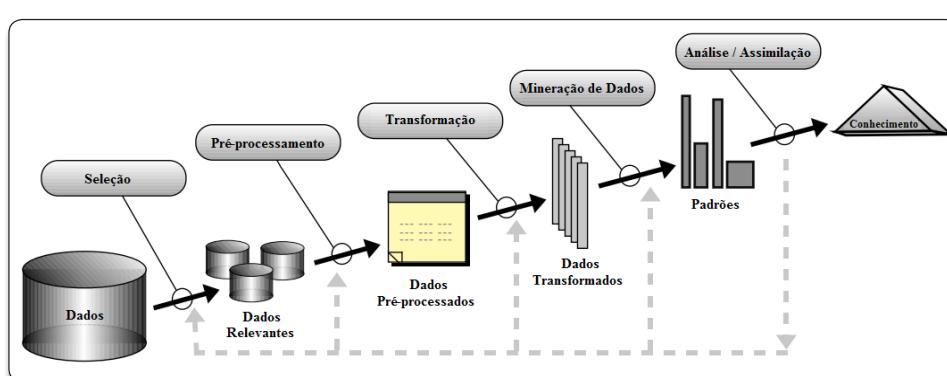


Figura 1. Passos que compõem o processo de KDD

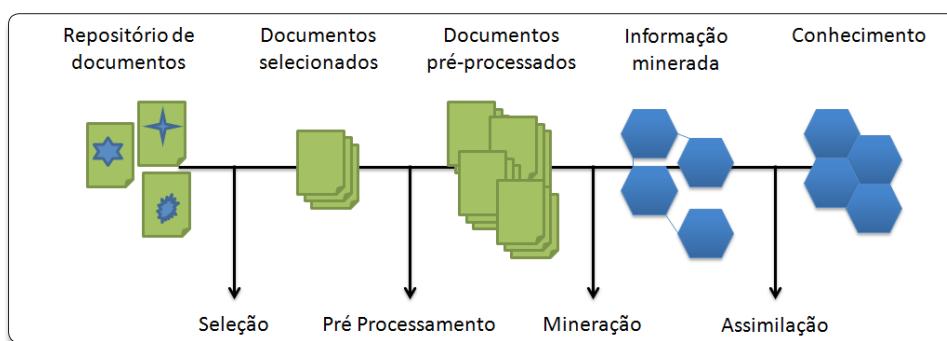


Figura 2. Processo de mineração de texto

Mineração de Texto

Mineração de texto é o processo de descoberta de conhecimento, potencialmente útil e previamente desconhecido, em bases de dados desestruturadas, ou seja, extração de conhecimento útil para o usuário em bases textuais.

O processo de mineração de texto é dividido em quatro etapas bem definidas: seleção, pré-processamento, mineração e assimilação, conforme é mostrado na **Figura 2**.

Na seleção, os documentos relevantes devem ser escolhidos, os quais serão processados. No pré-processamento ocorrerá a conversão dos documentos em uma estrutura compatível com o minerador, bem como ocorrerá um tratamento especial do texto. Na mineração, o minerador irá detectar os padrões com base no algoritmo escolhido. E por fim, na assimilação, os usuários irão utilizar o conhecimento gerado para apoiar as suas decisões.

É notório a semelhança entre os processos de KDD e KDT, sendo que o KDT

Mineração de texto: Análise comparativa de algoritmos

não possui a etapa de transformação. O fato da ausência da etapa transformação, etapa no processo de KDD que converte os dados pré-processados para uma estrutura utilizada na etapa de mineração de dados, é justificada pelo fato de que a etapa de pré-processamento no KDT, além de realizar um tratamento no texto, permite definir uma estrutura compatível com as entradas dos algoritmos de mineração.

A etapa pré-processamento pode ser dividida em quatro subetapas: remoção de StopWords, confluência, normalização de sinônimos e indexação. Na etapa remoção de stopwords os termos com pouca ou nenhuma relevância para o documento serão removidos. São palavras auxiliares ou conectivas, ou seja, não são discriminantes para o conteúdo do documento. São, em sua maioria, pronomes, preposições, artigos, numerais e conjunções. Para auxiliar na remoção das stopwords, geralmente, utiliza-se uma lista destas predefinida. Para facilitar o entendimento, na Figura 3 é apresentado um exemplo de remoção de stopwords.

Na etapa seguinte, confluência, realiza-se uma normalização morfológica, ou seja, realiza-se uma combinação das palavras que são variantes morfológicas em uma única forma de representação. Um dos procedimentos mais conhecidos de confluência é a radicalização (Stemming). Nela as palavras são reduzidas ao seu radical, ou seja, as palavras variantes morfológicamente serão combinadas em uma única representação, o radical. A radicalização pode ser efetuada com o auxílio de algoritmos de radicalização, sendo os mais utilizados o algoritmo de Porter (Porter Stemming Algorithm) e algoritmo de Orengo (Stemmer Portuguese ou RLSP). A Figura 4 exemplifica o processo de radicalização de um texto utilizando o algoritmo de Porter.

Existem dois problemas no processo de radicalização:

- **Overstemming:** quando a string removida não é um sufixo, mas sim parte do radical da palavra. Isso possibilita a combinação de palavras não relacionadas;
- **Understemming:** quando parte do sufixo não é removido, ocasionando numa falha de confluência de palavras relacionadas.

Após a confluência, na etapa de normalização de sinônimos, os termos que possuem significados similares serão agrupados em um único termo, por exemplo, as palavras ruído, tumulto e barulho serão substituídas ou representadas pelo termo barulho.

Na normalização de sinônimos, é formado um vocabulário controlado que se refere à utilização de termos adequados para representar um documento, sendo esses termos pré-definidos e específicos a um determinado assunto de uma área. Isso facilita a busca, pois os termos são comumente utilizados pelos usuários da área.

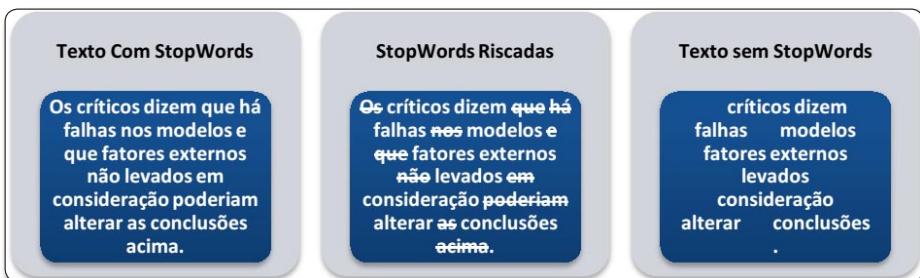


Figura 3. Exemplo de remoção de StopWords

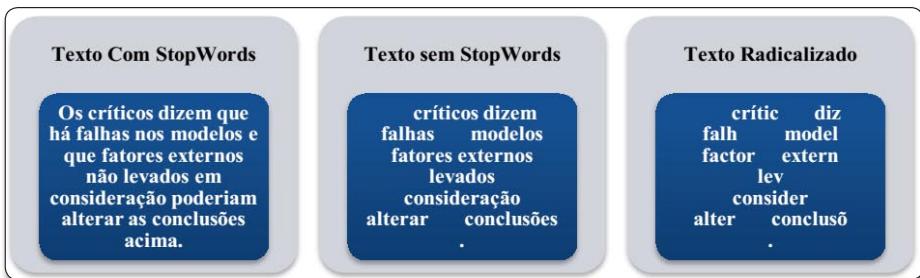


Figura 4. Exemplo de radicalização utilizando o algoritmo de Porter

E, por fim, na etapa indexação atribui-se uma pontuação para cada termo, garantindo uma única instância do termo no documento. No processo de atribuição de pesos devem ser considerados dois pontos: (a) quanto mais vezes um termo aparece no documento, mais relevante ele é para o documento; (b) quanto mais vezes um termo aparece na coleção de documentos, menos importante ele é para diferenciar os documentos.

Existem várias formas de determinar o peso de um termo (pontuação). Os principais métodos são:

- **Booleano ou Binário:** o peso para um determinado termo será 1 se o mesmo aparece no documento. Caso contrário, o peso será 0. Indica a presença ou ausência do termo no documento;
- **Frequência do Termo (*term frequency* ou *tf*):** o peso é a frequência do termo no documento. Consiste da razão entre a quantidade de vezes que o termo apareceu no documento e a quantidade total de termos contidos no documento, como é mostrado na Figura 5, onde n_i é a quantidade de ocorrências do termo i no documento e $|D|$ a quantidade total de termos no documento;

$$tf(\text{termo } i) = \frac{n_i}{|D|}$$

Figura 5. Fórmula para calcular a frequência do termo

- **Frequência do Documento (*Document Frequency* ou *df*):** é o número de documentos que possui um determinado termo;
- **Frequência Inversa do Documento (*Inverse Document Frequency* ou *idf*):** refere-se à importância de um termo em um conjunto de documentos. Quanto maior o idf , mais representativo é o termo para o documento. Consiste no logaritmo da razão entre o número total de documentos e a frequência do documento, conforme é demonstrado na Figura 6, onde $|N|$ é a quantidade

total de documentos e $df(termo\ i)$ a frequência do documento para o termo i ;

$$idf(termo\ i) = \log \frac{|N|}{df(termo\ i)}$$

Figura 6. Fórmula para calcular a frequência inversa do termo

- **tfidf (Term Frequency – Inverse Document Frequency):** o peso para o termo é associado na proporção da frequência do termo no documento e na proporção inversa do número de documentos na coleção em que o termo aparece pelo menos uma vez, ou seja, combina o tf com idf , como é mostrado na **Figura 7**, onde $tf(termo\ i)$ e $idf(termo\ i)$ são, respectivamente, o tf e idf do termo i . Obtém-se, assim, o índice de maior representatividade do termo.

$$tfidf(termo\ i) = tf(termo\ i) \times idf(termo\ i)$$

Figura 7. Fórmula para calcular o tfidf do termo

As subetapas do pré-processamento permitem uma redução da dimensionalidade do texto, pois um documento pode ser representado por um vetor de termos. Como um termo representa uma dimensão do texto, quanto maior a dimensionalidade do texto, mais complexa será a análise feita pelo algoritmo de mineração.

Assim como no KDD, o processo de mineração de texto possui diversas aplicações, como extração de palavras-chave, determinação de sistemas representacionais preferenciais, classificação de documentos por categoria, filtro de documentos, entre outras.

Similaridade de documentos

Um documento pode ser considerado um vetor de termos. Cada elemento do vetor é considerado uma coordenada dimensional e os documentos podem ser colocados num espaço euclidiano de n dimensões onde n é o número de termos. A posição do documento em cada dimensão é dada pelo peso (pontuação calculada na fase de indexação). A distância entre um documento e outro é o grau de similaridade. Documentos que possuem os mesmos termos acabam sendo colocados numa mesma região no espaço euclidiano, ou seja, são similares.

A similaridade entre dois documentos também pode ser obtida pelos termos que ocorrem em ambos, ou seja, pelos termos compartilhados. Os documentos mais similares são os que possuem mais termos em comum. No cálculo da similaridade, são ignorados os termos que ocorrem em um documento e que não ocorrem no outro. Em outras palavras, só interessam os termos que ocorrem nos dois, isto é, a ocorrência positiva desse em ambos.

Similaridade é considerada o coração do método de classificação K-Nearest-Neighbor. A diferença entre ambos é que no

K-Nearest-Neighbor consideram-se os k documentos mais similares. A depender do valor de k , podem ser considerados os documentos com score inferior aos de maior score para determinar a classe do novo documento.

Similaridade considera apenas os documentos com maior score e a classe do novo documento será a classe que mais ocorre nesses. É importante frisar que para o cálculo do grau de similaridade (score), devem ser apenas considerados os termos em comum.

Existem várias formas de calcular o grau de similaridade, isto é, as funções de similaridade. Depois de calcular os scores, podemos criar uma lista em forma de ranking, em que os documentos mais similares estão no topo da lista. As principais funções de similaridade são:

- **Contagem de palavras:** é considerada a função mais simples de mensurar a similaridade, pois se baseia apenas na contagem de termos que ocorrem em ambos documentos, isto é, as ocorrências positivas dos termos;
- **Contagem de palavras com bônus:** de forma análoga à contagem de palavras, serão contabilizados os termos em comum aos vetores com apenas um diferencial: para cada termo analisado, se esse termo ocorre em ambos documentos, será adicionado um bônus ao score conforme é visto na **Figura 8**, onde K é a quantidade total de termos do novo documento, $w(j)$ a pontuação para o termo j , $D(i)$ o documento i da coleção e a expressão $1/df(j)$ o bônus para o termo j . O bônus é considerado uma variação do idf . Se o termo ocorre em muitos documentos, o valor do bônus é baixo. Já se o termo aparece em poucos, o bônus é alto;

$$\begin{aligned} \text{Similarity}(D(i)) &= \sum_{j=1}^K w(j), \\ w(j) &= \begin{cases} 1 + 1/df(j), & \text{se o termo } j \text{ ocorre em ambos documentos} \\ 0, & \text{caso contrário} \end{cases} \end{aligned}$$

Figura 8. Fórmula para a contagem de palavras com bônus

- **Cosine similarity:** função de similaridade mais utilizada na área de recuperação de informação (RI) para comparar documentos. Representa o cosseno do ângulo formado por dois vetores, como é mostrado na **Figura 9**, onde d_1 e d_2 são os documentos cuja similaridade será calculada, $w_{d1}(j)$ o peso do termo j em d_1 , $w_{d2}(j)$ o peso do termo j em d_2 , $\sqrt{\sum(w_{d1}(j))^2}$ a normalização de d_1 e $\sqrt{\sum(w_{d2}(j))^2}$ a normalização de d_2 . Quanto mais próximo de zero for o valor do cosseno, menos similares são os documentos. Já quando for mais próximo de um, mais similares eles são;

$$\cos(d_1, d_2) = \frac{\sum (w_{d1}(j) \times w_{d2}(j))}{\sqrt{\sum (w_{d1}(j))^2} \times \sqrt{\sum (w_{d2}(j))^2}}$$

Figura 9. Fórmula para a cosine similarity

Mineração de texto: Análise comparativa de algoritmos

- **Distância euclidiana:** representa a menor distância entre dois vetores de termos no espaço euclidiano, como é visto na **Figura 10**, em que d_1 e d_2 são os documentos, K o número de termos, $w_{d1}(j)$ o peso do termo j em d_1 e $w_{d2}(j)$ o peso do termo j em d_2 ;

$$dist(d_1, d_2) = \sqrt{\sum_{j=1}^K (w_{d1}(j) - w_{d2}(j))^2}$$

Figura 10. Fórmula para a distância euclidiana

- **Distância de Manhattan:** é a soma das distâncias absolutas sem cada dimensão. Corresponde à distância a ser percorrida para se chegar de um ponto a outro, em que o caminho é percorrido em quadras, conforme é mostrado na **Figura 11**, onde d_1 e d_2 são os documentos, K o número de termos, $w_{d1}(j)$ o peso do termo j em d_1 e $w_{d2}(j)$ o peso do termo j em d_2 ;

$$dist(d_1, d_2) = \sum_{j=1}^K |w_{d1}(j) - w_{d2}(j)|$$

Figura 11. Fórmula para a distância manhattan

- **Produto escalar:** corresponde ao somatório do produto dos pesos de um termo em dois documentos, como é visto na **Figura 12**, onde d_1 e d_2 são os documentos, K o número de termos, $w_{d1}(j)$ o peso do termo j em d_1 e $w_{d2}(j)$ o peso do termo j em d_2 .

$$sim(d_1, d_2) = \sum_{j=1}^K w_{d1}(j) \times w_{d2}(j)$$

Figura 12. Fórmula para o produto escalar

Para facilitar o entendimento sobre similaridade, a **Figura 13** demonstra o cálculo da similaridade entre um novo documento e todos os documentos do dicionário, utilizando a função de similaridade contagem de palavras. Como podemos ver, foi calculado o score entre o novo documento e todos do dicionário por meio da contagem de palavras cuja ocorrência em ambos foi positiva, isto é, a contabilização delas que ocorre em ambos, ignorando as que ocorrem apenas em um e as ausentes em ambos. Existem dois documentos que possuem o maior score, grau de similaridade igual a 2.

Como os dois documentos com maior score possuem classe igual a um (última coluna do dicionário), a classe do novo documento também será um.

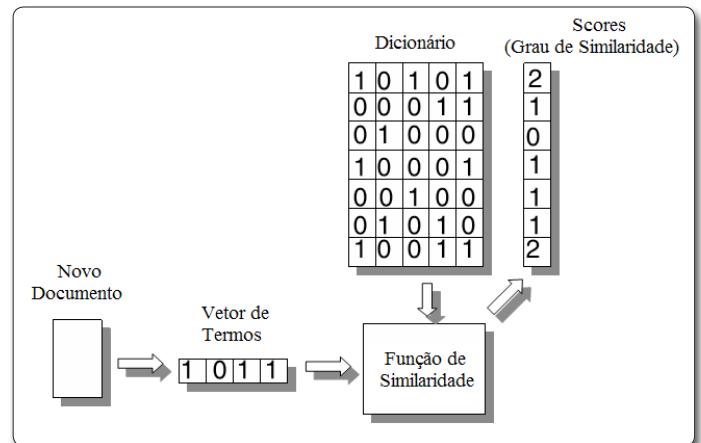


Figura 13. Cálculo de similaridade dos documentos

Avaliação de desempenho e qualidade

Existem diversas formas de se avaliar a capacidade de predição de um classificador para determinar a classe de vários registros. A "matriz de confusão" é a forma mais simples de analisar o desempenho e qualidade de um classificador em reconhecer registros de diferentes classes. Ela é um recurso que permite demonstrar o desempenho de um classificador, ou seja, a frequência com que os registros de classe X foram corretamente classificados como classe X ou, até mesmo, classificados erroneamente como outras classes.

Para n classes, a matriz de confusão é uma tabela de dimensão $n \times n$. Para cada classificação possível existe uma linha e coluna correspondente, ou seja, os valores das classificações serão distribuídos na matriz de confusão de acordo com os resultados, assim gerando a matriz de confusão para as classificações realizadas. As linhas correspondem às classificações corretas e as colunas representam as classificações realizadas pelo classificador. Por exemplo, na **Tabela 1**, o valor V1,1 corresponde ao número de registros de classe 1 que foram classificados com classe 1 pelo classificador.

Classe Atual	Classificado como			
	Classe 1	Classe 2	...	Classe n
Classe 1	V1,1	V1,2	...	V1,n
Classe 2	V2,1	V2,2	...	V2,n
...
Classe n	Vn,1	Vn,2	...	Vn,n

Tabela 1. Matriz de confusão para n classes

Quando existem apenas duas classes, uma é considerada como "positive" e a outra como "negative". Os valores da matriz de confusão são referenciados como *true positives* e *false positives* e *true negatives*, como é visto na **Tabela 2**.

Assim, existem quatro situações:

- **True Positive (TP):** é o número de instâncias de classe *positive* que foram classificadas como *positive*;
- **False Positive (FP):** é o número de instâncias de classe *negative* que foram classificadas como *positive*;

- **False Negative (FN):** é o número de instâncias de classe *positive* que foram classificadas como *negative*;
- **True Negative (TN):** é o número de instâncias de classe *negative* que foram classificadas como *negative*.

Actual class	Predicted class	
	Positive	Negative
Positive	True Positive (TP)	False Negative (FN)
Negative	False Positive (FP)	True Negative (TN)

Tabela 2. Matriz de confusão para duas classes

A avaliação de um classificador se dará pela análise dos valores nela contidos, bem como na verificação do somatório dos elementos das diagonais principal e secundária. Um bom classificador é aquele que possui a soma da diagonal principal maior que a da secundária. Um classificador é considerado ideal quando a soma da diagonal secundária é igual a zero, contudo esse será considerado um péssimo classificador se possuir o somatório da diagonal principal igual a zero.

De posse dos valores da matriz de confusão, podem ser utilizadas as métricas de avaliação de desempenho e qualidade de um classificador. As principais métricas de desempenho e qualidade são:

- **Acurácia (accuracy):** é o percentual de instâncias classificadas corretamente;
- **Precisão (precision):** é o percentual de instâncias classificadas como *positive* que são realmente *positive*;
- **Cobertura ou Revocação (recall):** é o percentual de instâncias *positive* que foram classificadas corretamente como *positive*;

- **Medida F (F1 Score):** é a medida que combina a precisão e revocação (cobertura), ou seja, é a média harmônica da precisão e revocação.

Ferramenta TextMining

O TextMining permite determinar se informações são ou não evidências de irregularidades, ou seja, se uma descrição está ou não de acordo com a lei e com o que se espera dos jurisdicionados. Desta forma, a ferramenta tem como objetivo tornar efetivo o trabalho do auditor na identificação de irregularidades. Suas principais funcionalidades são o gerenciamento de perfis, de dicionários e de classificações. Considera-se gerenciamento o conjunto de funções relacionadas ao cadastro, edição, consulta, exclusão e visualização de informações.

Iniciando pelo gerenciamento de perfis, este é um mecanismo que auxilia nas consultas por meio dos filtros anexados aos perfis. Conforme é mostrado na **Figura 14**, é por meio deles que o usuário poderá determinar dinamicamente os campos que deseja filtrar nas telas, nas quais poderá escolher o perfil. Na TextMining, está disponível para o usuário as funcionalidades de cadastro, consulta e exclusão.

A criação de um perfil poderá ocorrer só uma vez e pode ser compartilhado por todos os usuários. Como o custo da operação é muito baixo, se houver a necessidade de alteração dele, basta excluí-lo e criar outro novamente. Esta característica torna a aplicação flexível e genérica através da geração de perfis de consulta diferenciados para qualquer tabela e campos contidos na base de dados.

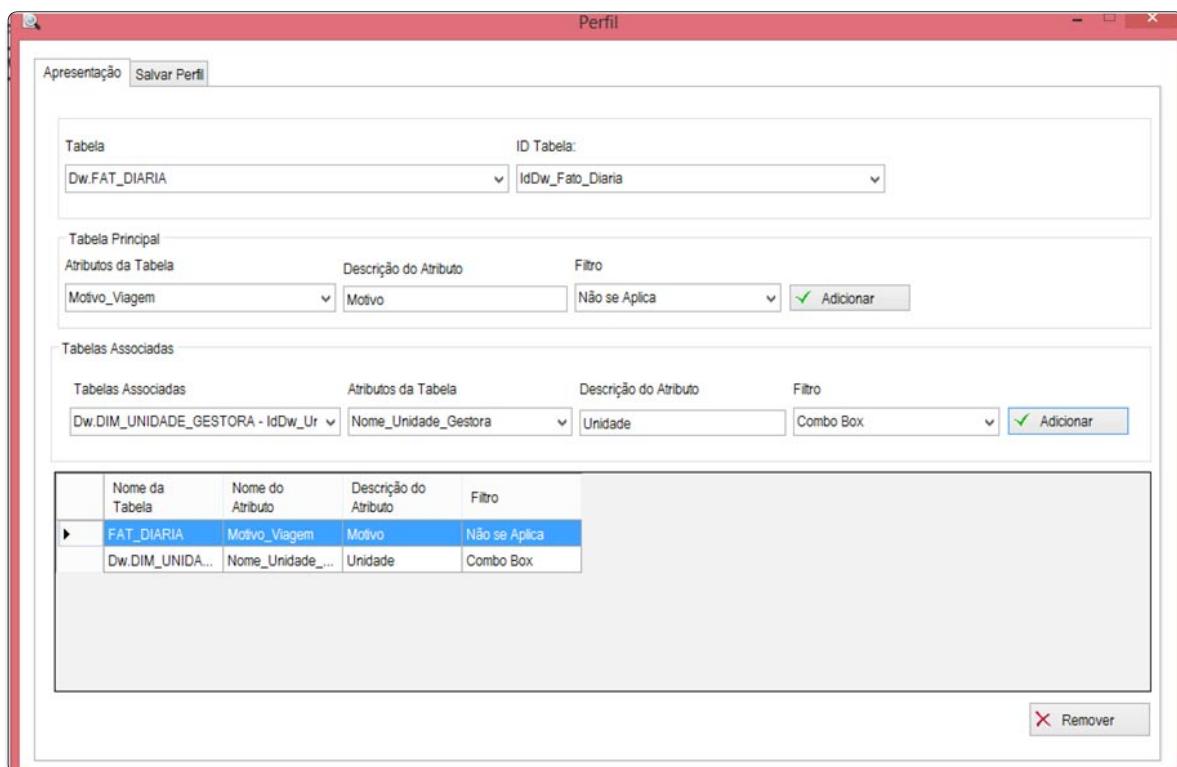


Figura 14. Tela Perfil (Ferramenta TextMining)

Mineração de texto: Análise comparativa de algoritmos

Dados estes entendimentos sobre perfis, outra funcionalidade importantíssima é o gerenciamento de dicionários, que são os modelos de conhecimento que servem de base para tornar possível a descoberta de evidências de fraudes semelhantes em toda base de dados ou em unidades e cidades específicas. Um dicionário é criado por meio da seleção de amostras que são dados selecionados pelo auditor como “Evidência” (possível evidência de irregularidade) e “Em Conformidade” (descrição que está de acordo com a lei), bem como o auditor pode informar amostras avulsas, as quais são especificadas manualmente e classificadas como “Evidência” ou “Em Conformidade”, como é mostrado na Figura 15.

A seleção de amostras para criação do dicionário deve ser balanceada, para cada evidência informada, deverá existir um ou mais registros que são exemplos de conformidade. Na ferramenta estão disponíveis para o usuário as funcionalidades de cadastro, consulta, edição, exclusão e desbloqueio de dicionários. É importante ressaltar que o dicionário criado poderá ser utilizado por todos os auditores, permitindo maior eficiência ao processo de auditoria.

A partir do perfil selecionado, dos filtros anexados a esse e do dicionário escolhido, o auditor poderá escolher os dados a serem classificados pela ferramenta, ou seja, o local em que serão buscadas novas evidências semelhantes às do dicionário criado.

A aplicação dispõe de dois algoritmos de mineração de texto, Naïve Bayes e Similaridade, para classificar os registros, como é mostrado na Figura 16. Naïve Bayes é um algoritmo de análise estatística e foi implementado de forma parametrizada, especificando um limiar mínimo para auxiliar na classificação dos registros. Para realizar a classificação de um registro, o algoritmo calcula a probabilidade desse registro ser ou não uma evidência de irregularidade. Este algoritmo dispõe de três formas para realizar o cálculo da probabilidade: “Híbrido”, “Frequência Inversa” e “Frequência”. Na primeira abordagem, é considerada a frequência do termo na amostra e o tf desse na sentença. Já na segunda é levado em conta o tfidf do termo na amostra e o tf na sentença. Por fim, na terceira, são consideradas as frequências do termo na amostra e na sentença.

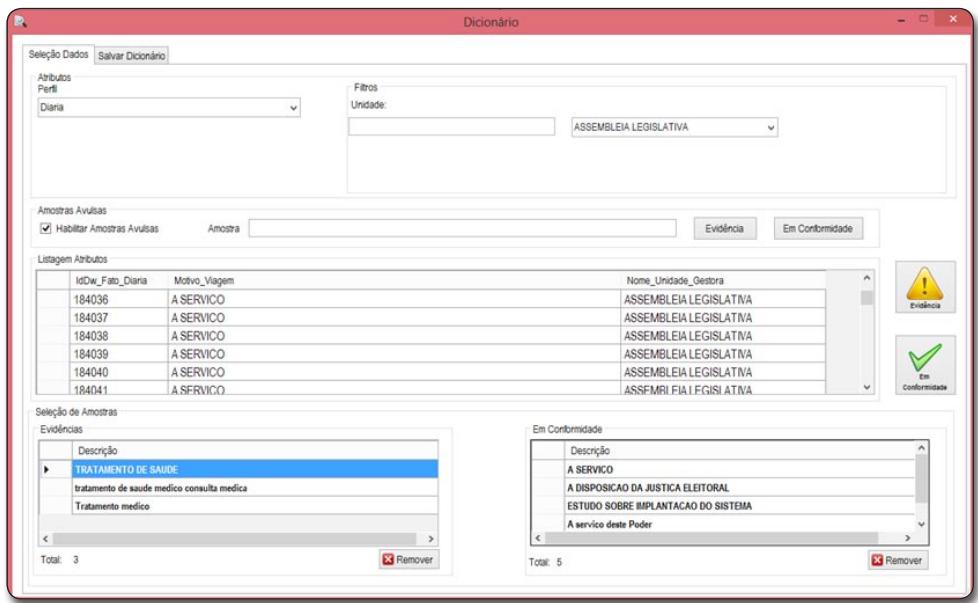


Figura 15. Tela Dicionário (Ferramenta TextMining)

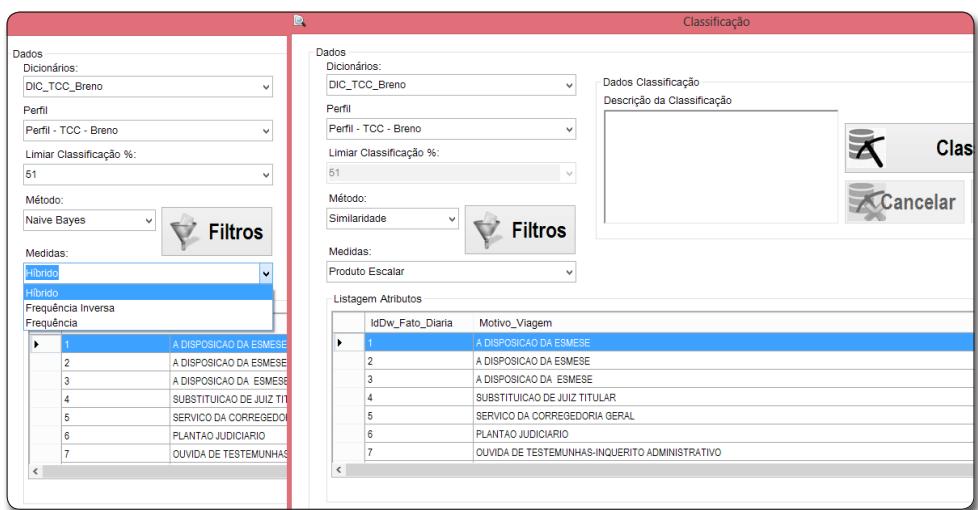


Figura 16. Telas Classificação sobrepostas mostrando os algoritmos disponíveis (Ferramenta TextMining)

Já o algoritmo de similaridade, também de análise estatística, calcula a similaridade entre uma sentença e um conjunto de amostras, por meio dos termos que ambos possuem em comum para determinar se a sentença é ou não uma evidência.

Na ferramenta, estão disponíveis para o usuário as funcionalidades de cadastro, consulta, edição e visualização de classificações.

Após a análise da aplicação foram efetuadas alterações no código objetivando melhoria no uso da ferramenta, inclusão de novas funcionalidades, prevenção e correção de problemas. A seguir temos as principais alterações realizadas:

- **Modelo de Dados:**

- Inclusão dos atributos “Metodo_Classificacao” e “Tempo_Classificacao” na tabela “DIM_CLASSIFICACAO”.

- **Módulo Dicionário:**

- A tela de criação de dicionários foi alterada para permitir a inclusão de amostras avulsas;
- A tela de consulta de dicionários foi alterada para que os botões “Editar” e “Excluir” ficassem desabilitados quando não existissem dicionários cadastrados;
- A tela “Dicionários Bloqueados” foi alterada para que o botão “Liberar” ficasse desabilitado quando não existissem dicionários bloqueados.

- **Módulo Classificação:**

- Criação da classe similaridade, algoritmo de classificação;
- Criação da classe abstrata Classificador, superclasse das classes Similaridade e NaiveBayes. A classe Classificador possui um método estático ClassificadorFactory que retorna um objeto do tipo Classificador, o qual pode ser uma instância das classes NaiveBayes ou Similaridade;
- Na tela “Classificação”, o algoritmo “Similaridade” foi incluído nas opções de métodos de classificação;
- Na tela “Classificação”, foram adicionados os percentuais 51 e 55 ao componente “Limiar Classificação %”;
- A tela “Classificação” foi alterada para os componentes “Dicionários”, “Perfil”, “Limiar Classificação %”, “Método”, “Medidas”, “Filtros”, “Classificar” e “Descrição da Classificação” serem desabilitados quando um processo classificatório fosse iniciado;
- A tela de consulta de classificações foi alterada para que os botões “Excluir” e “Detalhes” ficassem desabilitados quando não existissem classificações cadastradas;
- As alterações da tela “Dados da Classificação” foram:
 - Criação do componente “Tempo de Classificação” para visualizar o tempo da classificação realizada;
 - Atribuição do valor “---” para o componente “Limiar de Classificação %” quando o algoritmo de mineração utilizado não for o Naïve Bayes, porque este é o único que utiliza limiar;
 - Parametrização da tabela “Evidências”, em que o label e os valores da coluna do Limiar/Score serão formatados de acordo com o algoritmo utilizado na classificação. Por exemplo, caso o algoritmo seja Naïve Bayes, o label da coluna será “Limiar” e os valores da coluna estarão formatados em porcentagem, mas se for escolhido Similaridade, o label da coluna será “Score” e os valores da coluna estarão formatados em números com casas decimais.

Implementamos uma adaptação do algoritmo de similaridade de documentos. O método convencional de similaridade realiza um cálculo de similaridade entre todos os documentos do dicionário e o documento a ser classificado, apenas levando em conta os termos que ocorrem em ambos.

Para classificar o novo documento, o algoritmo convencional apenas considera as ocorrências do maior score, ignorando todos os outros. Nessa abordagem, existe a possibilidade de

não classificar um novo documento quando a quantidade de documentos com maior score para diferentes classes, é a mesma, como é mostrado na **Figura 17**.

Visando performance e melhoria do processo de classificação, foi desenvolvida uma adaptação do algoritmo de similaridade. As adaptações efetuadas foram:

- Será realizada uma poda, onde não serão analisados todos os documentos (amostras) do dicionário, somente aqueles que possuírem pelo menos um termo do documento (sentença) a ser classificado para determinar o grau de similaridade;
- Serão utilizados os outros scores quando não for possível classificar com o maior score.

Classe dos Documentos do Dicionário	Vetor de Scores	Classe dos Documentos do Dicionário	Vetor de Scores
1 0 1 1 0 0 1	2 1 0 1 1 1 2	1 0 1 1 0 0 0	2 1 0 1 1 1 2
a) Resultado da Classificação: Classe 1		b) Resultado da Classificação: Não Classifica	
Classe dos Documentos do Dicionário	Vetor de Scores	Classe dos Documentos do Dicionário	Vetor de Scores
1 0 1 1 0 0 1	3 1 0 1 1 1 2	1 0 1 1 0 0 1	1 1 0 1 1 1 1
c) Resultado da Classificação: Classe 1		d) Resultado da Classificação: Não Classifica	

Figura 17. Exemplo de quatro situações de classificação do algoritmo convencional de similaridade

$$score = \sum_{j=1}^k (tfidf_{amostra}(j) \times tf_{sentença}(j))$$

Figura 18. Fórmula para o cálculo do score utilizado no algoritmo implementado

O score para uma determinada amostra (documento do dicionário) pode ser obtido através do somatório da multiplicação do *tfidf* do termo da amostra com o *tf* da sentença (novo documento) para todos os termos comuns entre a amostra e a sentença, como pode ser visto na **Figura 18**, onde k é o número de termos que a amostra e a sentença possuem em comum.

Para facilitar o entendimento do algoritmo implementado, é apresentado a seguir seu passo a passo.

Mineração de texto: Análise comparativa de algoritmos

Entrada:

sc, sentença a ser classificada como “Evidência” ou “Em Conformidade”

dic, dicionário utilizado para classificar a sentença

Saída:

result, resultado da classificação da sentença, ou seja, objeto ResultadoGenerico contendo a classe e o score da sentença classificada

1. Calcular o *tf* (*term frequency*) para cada termo de *sc*.
2. Para cada amostra que contenha pelo menos um termo de *sc*, calcular o *score* para cada amostra, criar um objeto ResultadoGenerico para cada amostra analisada, contendo nesse objeto a classe da amostra e o *score* calculado e, por fim, armazenar o objeto no vetor de ResultadoGenerico.
3. Ordenar o vetor de ResultadoGenerico em ordem crescente pelo score.

4. Se o vetor estiver vazio Então

Retorne um objeto ResultadoGenerico com classe igual a falso e score igual a zero.

Senão

5. Se o vetor possuir apenas um elemento Então

Retorne o único objeto ResultadoGenerico dentro do vetor.

Senão

6. Se dentre os objetos ResultadoGenerico no vetor existe apenas uma ocorrência de um objeto com o maior score Então

Retorne o objeto ResultadoGenerico com maior *score*.

Senão

7. Para todas as ocorrências dos objetos ResultadoGenerico com maior *score*, realizar a contagem de objetos que possuem classe igual a falso (“Em Conformidade”) e também os que possuem classe igual a verdadeiro (“Evidência”).

8. Se a quantidade de verdadeiro for maior que a de falso Então

Retorne um objeto ResultadoGenerico com classe igual a verdadeiro e *score* igual ao maior *score*.

Senão

9. Se a quantidade de falso for maior que a de verdadeiro Então

Retorne um objeto ResultadoGenerico com classe igual a falso e *score* igual ao maior *score*.

Senão

10. Enquanto não for possível classificar a sentença (quantidade de verdadeiros e falsos forem iguais) e nem todos os objetos do vetor foram analisados, realizar os passos 6 a 9, considerando que o novo maior *score* será o *score* do objeto ResultadoGenerico que antecede a primeira ocorrência do objeto que possui o atual maior *score*.

11. Se não foi possível classificar analisando todos os elementos do vetor Então

Retorne um objeto ResultadoGenerico com classe igual a falso e *score* igual a zero.

Estudo de caso

A realização do estudo de caso teve por objetivo principal a validação dos resultados emitidos pela ferramenta TextMining para detecção de irregularidades nos pagamentos de diárias

contidos nos históricos de contas públicas sob custódia da organização ABC. Para atingir este objetivo, é necessária a efetivação dos seguintes passos:

- Selecionar os participantes e objetos do estudo de caso;
- Definir o dicionário a ser utilizado;
- Executar o processo classificatório nas amostras dos participantes envolvidos para cada algoritmo de mineração de texto;
- Verificar e validar os resultados obtidos por meio das métricas de Tempo Médio de Execução, Acurácia, Precisão, Cobertura e Medida F;
- Realizar alterações na ferramenta, se necessários.

Primeiramente foram selecionados os participantes e objetos, em seguida, a definição do dicionário utilizado, a determinação das métricas para a avaliação de desempenho e qualidade dos algoritmos e, por fim, foi realizada a execução do estudo.

Para a seleção dos participantes, foi necessário analisar dois critérios: os participantes devem ser unidades gestoras cadastradas e que possuam uma quantidade considerável de registros cadastrados. De acordo com o DW cedido, existem 481 unidades gestoras cadastradas, sendo assim, serão escolhidas, aleatoriamente, três unidades para a realização do estudo. Por questão de sigilo das informações, os nomes das unidades gestoras não serão revelados.

As unidades escolhidas, com a quantidade de registros especificados entre parênteses, foram: Unidade A (8872), Unidade B (625) e Unidade C (1855). É importante ressaltar que para as unidades A e C, também foram escolhidas dentro da quantidade de registros, aleatoriamente, amostras de 500 registros para o estudo. É fundamental frisar que a base de treinamento será constituída pela Unidade A, já a base de teste será formada pelas unidades B e C.

Após a escolha das unidades, é fundamental determinar o atributo na tabela de fato a ser minerado, ou seja, o campo descritivo. De acordo com a Figura 19, existem cinco campos descritivos: IdDw_Fato_Diaria, Matricula_Funcionario, Destino_Viagem, Motivo_Viagem e Numero_Empenho. Dentre estes, para detectar irregularidades no pagamento de diárias, o atributo mais significativo é Motivo_Viagem, porque o mesmo representa a justificativa da concessão de uma diária.

Dicionário utilizado

Diária é uma espécie de auxílio financeiro ou ajuda de custo para um colaborador prestar algum serviço fora da localidade do órgão ao qual esteja vinculado. Em outras palavras, é um auxílio recebido pelo colaborador com o intuito de custear seus gastos para a realização de serviço fora do local de trabalho. A concessão de diárias é diversificada, pois abrange gastos referentes à capacitação, viagens para reuniões com superiores, entre outras.

Diante do exposto, é proibida a concessão de diárias para fins que não sejam relacionados à prestação de serviço. Existem inúmeras justificativas consideradas evidências de irregularidades para concessão de diárias como realização de uma viagem particular. Assim, para a definição do

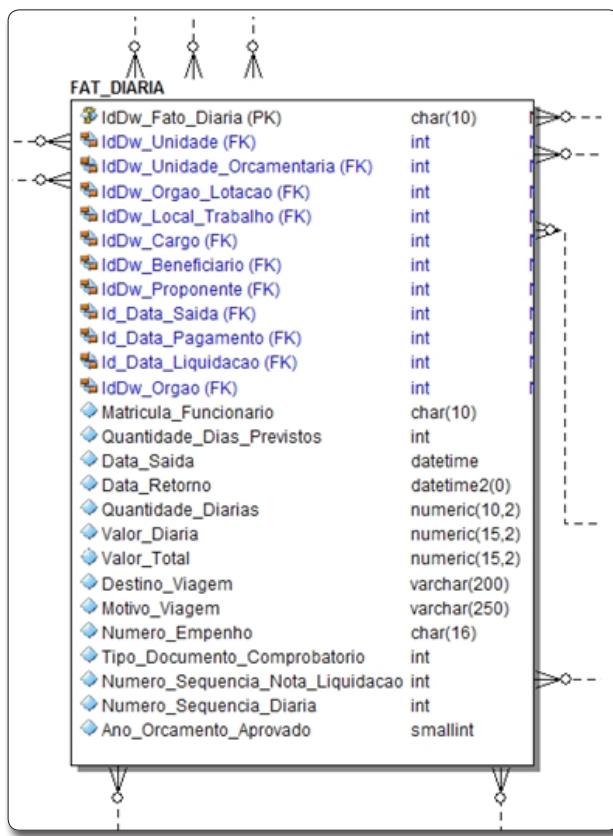


Figura 19. Tabela de Fato de Diárias

dicionário a ser utilizado neste estudo, optou-se em restringir esse conjunto para evidências relacionadas ao uso de diárias para tratamento de saúde.

O modelo de conhecimento (dicionário a ser utilizado) possuirá, no total, 60 sentenças constituídas de amostras da própria base e de amostras avulsas para classificar registros. Para as amostras da própria base foram escolhidas, aleatoriamente, 40 sentenças da Unidade A, sendo 20 classificadas como “Evidência” e as outras 20 como “Em Conformidade”.

Para as amostras avulsas, foram definidas 20 sentenças, sendo 10 classificadas como “Evidência” e as outras 10 como “Em Conformidade”. As sentenças avulsas classificadas como “Em Conformidade” são similares às da própria base, as quais foram formuladas por meio de uma análise das amostras dos dados das unidades gestoras envolvidas.

Já das sentenças avulsas classificadas como “Evidência”, apenas duas são similares às da própria base, por meio da análise da amostra dos dados da Unidade A. Para formular as sentenças avulsas restantes, com o intuito de obter termos da Medicina sobre procedimentos, tratamentos e cirurgias médicas, profissionais da saúde, doenças e exames, foram realizadas algumas pesquisas sobre “Medicina”.

Com a ajuda de uma especialista na área da saúde, foram selecionados, por categoria, apenas os termos mais comuns e relevantes, a exemplo de doenças mais comuns e exames de rotinas mais solicitados.

AMOSTRAS DA PRÓPRIA BASE

Sentença – “Em Conformidade”	Sentença – “Evidência”
A DISPOSIÇÃO DA JUSTIÇA ELEITORAL	ACOMPANHAMENTO DE TRATAMENTO DE SAÚDE DE SUA FILHA
A SERVIÇO DA ASSEMBLEIA	ACOMPANHANTE DA SRA DEP. CELIA FRANCO P/TRATAMENTO MÉDICO
A SERVIÇO DESTE PODER	ACOMPANHAR A DEPUTADA PARA TRATAMENTO DE SAÚDE
A TRABALHO	ACOMPANHAR A FILHA EM TRATAMENTO MÉDICO
ACOMPANHAR O SR.PRIMEIRO SECRETARIO	ATENDER PROCEDIMENTOS MÉDICOS
ACOMPANHAR PROCESSOS E REUNIÕES PARLAMENTARES	CONSULTA MÉDICA
ASSUNTO DE INTERESSE DESTE PODER	DESPESAS MÉDICA
AUTORIZADA PELO PRIMEIRO SECRETÁRIO	FAZER EXAMES PARA LIBERAÇÃO DE TRANSPLANTE DE RINS
ENCONTRO DO PARTIDO PROGRESSISTA	PARA A FUNCIONARIA A TRATAMENTO DE SAÚDE
ESTUDO SOBRE IMPLANTAÇÃO DO SISTEMA	PARA A SERVIDORA REALIZAR TRATAMENTO CLÍNICO
INTEGRAR COMITIVA DO GOVERNO DO ESTADO	REVISÃO MÉDICA
PARA O SR. DEPUTADO PARTICIPAR DE REUNIÃO DO PARTIDO	SUBMETER-SE A CONSULTA MÉDICA
PARTICIPAR DE REUNIÃO DE CUNHO POLÍTICO PARTIDÁRIO	SUBMETER-SE A EXAMES MÉDICOS
PARTICIPAR DA CONVENÇÃO NACIONAL DO DEM	SUBMETER-SE A TRATAMENTO MÉDICO
PARTICIPAR DA POSSE DO PRESIDENTE DA PETROBRAS DISTRIBUIDORA	TRATAMENTO MÉDICO
PARTICIPAR DE ATO PÚBLICO CONTRA REFORMA SINDICAL	TRATAMENTO DE SAÚDE
PARTICIPAR DO I CONGRESSO INTERMUNICIPAL DE SAÚDE	TRATAMENTO DE SAÚDE
REUNIÃO DE CUNHO POLÍTICO PARTIDÁRIO	TRATAMENTO DE SAÚDE
TRATAR DE ASSUNTO DE INTERESSE DESTE PODER	TRATAMENTO DE SAÚDE
VISITAR A SUPERINTENDÊNCIA DA CAIXA ECONÔMICA FEDERAL	TRATAMENTO DE SAÚDE

Tabela 3. Amostras da base

Mineração de texto: Análise comparativa de algoritmos

AMOSTRAS "AVULSAS"	
Sentença – "Em Conformidade"	Sentença – "Evidência"
Complementação de diária para funcionário realizar trabalho.	25-HIDROXIVITAMINA D OU 25(OH)D; ACIDO ÚRICO NO SANGUE; ALBUMINA; ALTERAÇÕES DO FERRO E DE SUA CAPACIDADE DE FIXAÇÃO; AUDIOMETRIA VON BEKESY; BILIRRUBINA NA URINA; BILIRRUBINA NO SANGUE (DIRETA, INDIRETA E TOTAL). PROVAS DE FUNÇÃO HEPÁTICA (BILIRRUBINAS, ELETROFORESE DE PROTEÍNAS. FA, TGO, TGP E GAMA-PGT); CÁLCIO NO SANGUE; CARDIOLIPINA, AUTO-ANTICORPOS IGG; ...
Conduzir pacientes para hospital	ABLATIVA; ABORTO; ACUPUNTURA; ALOPATIA; AMPUTAÇÃO; ANESTESIA; ANTI-SEPSIA; APENDICECTOMIA; ASSEPSIA; AUSCULTAÇÃO; AUTÓPSIA; BARIÁTRICA; BIÓPSIA; CABEÇA; CARDÍACA; CAUTERIZAÇÃO; CHECK-UP; CIRURGIA; COLUNA; COSTURA; DRENO; ...
Pagamento de diária para servidor ou funcionário realizar serviços fora desta unidade	AMBULATÓRIO; ASSISTÊNCIA MÉDICA; ATENDIMENTO MÉDICO; CARDIOGRAMA; CLÍNICA; CONSULTA MÉDICA; CONSULTÓRIO; DIAGNÓSTICO MÉDICO; DESPESA MÉDICA; ELETROENCEFALOGRAMA; EMERGÊNCIA; EXAMES MÉDICOS; HISTÓRICO DE SAÚDE; HOSPITAL; LAUDO; PERÍCIA MÉDICA; POLICLÍNICA; PROCEDIMENTO DE SAÚDE; PRONTO-SOCORRO; QUADRO CLÍNICO; RADIOGRAFIA; REVISÃO MÉDICA; VACINA
Viagem para realização de serviço desta unidade	REALIZAÇÃO DE EXAMES MÉDICOS DA ESPOSA E FILHOS.
Viagem para tratar de assuntos de saúde pública e obter recursos financeiros.	REALIZAR PROCEDIMENTO CIRÚRGICO.

Tabela 4. Amostras avulsas

Nas Tabelas 3 e 4 são apresentadas as amostras que constituem o modelo de conhecimento.

Medidas de desempenho e qualidade para avaliação dos algoritmos

Para analisar o desempenho e qualidade dos algoritmos de mineração de texto em questão será utilizado o recurso “matriz de confusão”, bem como as métricas de acurácia, cobertura, precisão e medida F e tempo de execução.

De acordo com o contexto deste trabalho, devemos considerar quatro situações:

- NSCCE: número de sentenças classificadas corretamente como “Evidência” (*True Positive*).
- NSCCC: número de sentenças classificadas corretamente como “Em Conformidade” (*True Negative*).
- NSCEE: número de sentenças classificadas erroneamente como “Evidência” (*False Positive*).
- NSCEC: número de sentenças classificadas erroneamente como “Em Conformidade” (*False Negative*).

A matriz de confusão que contempla essas situações pode ser vista na Tabela 5. Com a matriz de confusão criada, podemos definir as métricas a serem utilizadas:

- Acurácia é o percentual de sentenças classificadas corretamente pelo classificador;
- Cobertura é o percentual de evidências que foram classificadas corretamente como “Evidência”;
- Precisão é o percentual de sentenças classificadas como “Evidência” que são realmente evidências;
- Medida F, também conhecida como Média Harmônica da Precisão e Cobertura, é a medida que combina a precisão e cobertura;

Classificação Correta	Classificado como	
	Evidência	Em Conformidade
Evidência	NSCCE	NSCEC
Em Conformidade	NSCEE	NSCCC

Tabela 5. Matriz de confusão utilizada

- Tempo de Execução é o tempo de duração de uma classificação, compreendida pela diferença entre o tempo de término e o tempo de início da classificação.

A etapa de execução consistirá na realização do processo classificatório nas amostras dos participantes envolvidos para cada algoritmo de mineração de texto, utilizando o modelo de conhecimento definido. Foram efetuadas três classificações nas amostras dos participantes envolvidos para cada algoritmo. É necessário frisar que para cada execução do Naïve Bayes foi utilizado cada método desse algoritmo (“Híbrido”, “Frequência Inversa” e “Frequência”), bem como o limiar de 51 % foi utilizado em todas as execuções do Naïve Bayes. Após o término do processo classificatório, as matrizes de confusão foram geradas a partir dos resultados obtidos das classificações efetuadas, assim como foram coletadas as métricas para cada algoritmo.

Resultados

Após a realização do estudo de caso, foram coletados os valores das matrizes de confusão de cada execução para cada algoritmo e unidade escolhida. Com os valores das matrizes de confusão foi possível coletar as métricas para avaliar todas as abordagens. Vale ressaltar que para um mesmo algoritmo e unidade gestora,

a matriz de confusão foi a mesma para as três execuções (processos classificatórios). Nas **Tabelas 6 e 7** é apresentado um resumo dos valores das matrizes de confusão por algoritmo e unidade.

Inicialmente foram analisados os resultados das três classificações realizadas para cada algoritmo na Unidade A. Conforme é visto nas **Tabelas 8 a 10**, os algoritmos Naïve Bayes – Híbrido (N.B.H.) e Naïve Bayes – Frequência (N.B.F.) são as melhores abordagens para essa unidade, pois possuem as melhores porcentagens de acurácia (100%), precisão (100%), cobertura (100%) e medida F (100%). Similaridade (SIM) obteve um melhor desempenho do que os demais na métrica tempo de execução. Comparando similaridade e Naïve Bayes – Frequência Inversa (N.B.F.I.), similaridade supera esse nas métricas de cobertura, medida F e tempo de execução, mas ambos possuem a mesma porcentagem de acurácia. É importante verificar que o algoritmo similaridade possui precisão inferior em relação às demais abordagens.

A próxima unidade analisada foi a Unidade B. Também foram analisados os resultados das três classificações realizadas para cada algoritmo na referida unidade. De acordo com as **Tabelas 11 a 13**, o algoritmo similaridade obteve um melhor desempenho do que os demais apenas na métrica de tempo de execução. Já o Naïve Bayes – Frequência Inversa obteve um melhor desempenho na métrica acurácia (98,08%). Já Similaridade obteve a menor porcentagem de acurácia. É importante verificar que todas as abordagens tiveram um péssimo desempenho na métrica precisão (valor abaixo de 50%), mas Naïve Bayes – Híbrido e Naïve Bayes – Frequência tiveram desempenho melhor que os demais.

Apesar dos péssimos resultados, Naïve Bayes – Frequência Inversa foi a melhor abordagem, pois o mesmo classificou, erroneamente, um número muito inferior de evidências do que os outros algoritmos, como é mostrado nas **Tabelas 6 e 7**. Em outras palavras, comparando-se o resultado da soma entre NSCCE e NSCCC (soma da diagonal principal da

Unidades	Valores da Matriz de Confusão – Diagonal Principal							
	NSCCE (TP)				NSCCC (TN)			
	N.B. F.I.	N.B. H.	N.B. F.	SIM.	N.B. F.I.	N.B. H.	N.B. F.	SIM.
Unidade A	136	137	137	137	363	363	363	362
Unidade B	0	1	1	0	613	603	603	588
Unidade C	3	3	3	2	457	449	449	430

Tabela 6. Valores da matriz de confusão por algoritmo e unidade gestora – Diagonal Principal

Unidades	Valores da Matriz de Confusão – Diagonal Secundária							
	NSCEE (FP)				NSCEC (FN)			
	N.B. F.I.	N.B. H.	N.B. F.	SIM.	N.B. F.I.	N.B. H.	N.B. F.	SIM.
Unidade A	0	0	0	1	1	0	0	0
Unidade B	11	21	21	36	1	0	0	1
Unidade C	40	48	48	67	0	0	0	1

Tabela 7. Valores da matriz de confusão por algoritmo e unidade gestora – Diagonal Secundária

Execuções	Métricas de Desempenho e Qualidade							
	Acurácia				Precisão			
	N.B. F.I.	N.B. H.	N.B. F.	SIM.	N.B. F.I.	N.B. H.	N.B. F.	SIM.
1ª Exec.	99,80 %	100 %	100 %	99,80 %	100 %	100 %	100 %	99,28 %
2ª Exec.	99,80 %	100 %	100 %	99,80 %	100 %	100 %	100 %	99,28 %
3ª Exec.	99,80 %	100 %	100 %	99,80 %	100 %	100 %	100 %	99,28 %
Média	99,80 %	100 %	100 %	99,80 %	100 %	100 %	100 %	99,28 %

Tabela 8. Comparativo das métricas acurácia e precisão para cada algoritmo na Unidade A

Execuções	Métricas de Desempenho e Qualidade							
	Cobertura				Medida F			
	N.B. F.I.	N.B. H.	N.B. F.	SIM.	N.B. F.I.	N.B. H.	N.B. F.	SIM.
1ª Exec.	99,27 %	100 %	100 %	100 %	99,63 %	100 %	100 %	99,64 %
2ª Exec.	99,27 %	100 %	100 %	100 %	99,63 %	100 %	100 %	99,64 %
3ª Exec.	99,27 %	100 %	100 %	100 %	99,63 %	100 %	100 %	99,64 %
Média	99,27 %	100 %	100 %	100 %	99,63 %	100 %	100 %	99,64 %

Tabela 9. Comparativo das métricas cobertura e medida F para cada algoritmo na Unidade A

Execuções	Métricas de Desempenho e Qualidade			
	Tempo de Execução			
	N.B. F.I.	N.B. H.	N.B. F.	SIM.
1ª Exec.	135,88 s	133,49 s	146,79 s	84,78 s
2ª Exec.	135,24 s	139,28 s	153,68 s	83,39 s
3ª Exec.	134,64 s	134,81 s	156,64 s	82,22 s
Média	135,25 s	135,86 s	152,37 s	83,46 s

Tabela 10. Comparativo da métrica tempo de execução para cada algoritmo na Unidade A

Mineração de texto: Análise comparativa de algoritmos

Execuções	Métricas de Desempenho e Qualidade							
	Acurácia				Precisão			
	N.B. F.I.	N.B. H.	N.B. F.	SIM.	N.B. F.I.	N.B. H.	N.B. F.	SIM.
1ª Exec.	98,08 %	96,64 %	96,64 %	94,08 %	0,00 %	4,55 %	4,55 %	0,00 %
2ª Exec.	98,08 %	96,64 %	96,64 %	94,08 %	0,00 %	4,55 %	4,55 %	0,00 %
3ª Exec.	98,08 %	96,64 %	96,64 %	94,08 %	0,00 %	4,55 %	4,55 %	0,00 %
Média	98,08 %	96,64 %	96,64 %	94,08 %	0,00 %	4,55 %	4,55 %	0,00 %

Tabela 11. Comparativo das métricas acurácia e precisão para cada algoritmo na Unidade B

Execuções	Métricas de Desempenho e Qualidade							
	Cobertura				Medida F			
	N.B. F.I.	N.B. H.	N.B. F.	SIM.	N.B. F.I.	N.B. H.	N.B. F.	SIM.
1ª Exec.	0,00 %	100 %	100 %	0,00 %	甫	8,70 %	8,70 %	甫
2ª Exec.	0,00 %	100 %	100 %	0,00 %	甫	8,70 %	8,70 %	甫
3ª Exec.	0,00 %	100 %	100 %	0,00 %	甫	8,70 %	8,70 %	甫
Média	0,00 %	100 %	100 %	0,00 %	甫	8,70 %	8,70 %	甫

Tabela 12. Comparativo das métricas cobertura e medida F para cada algoritmo na Unidade B

Execuções	Métricas de Desempenho e Qualidade							
	Tempo de Execução							
	N.B. F.I.	N.B. H.	N.B. F.	SIM.				
1ª Exec.	256,01 s	268,49 s	276,44 s	159,55 s				
2ª Exec.	259,67 s	258,26 s	266,54 s	158,14 s				
3ª Exec.	254,64 s	271,76 s	291,13 s	159,16 s				
Média	256,77 s	266,17 s	278,04 s	158,95 s				

Tabela 13. Comparativo da métrica tempo de execução para cada algoritmo na Unidade B

Execuções	Métricas de Desempenho e Qualidade							
	Acurácia				Precisão			
	N.B. F.I.	N.B. H.	N.B. F.	SIM.	N.B. F.I.	N.B. H.	N.B. F.	SIM.
1ª Exec.	92,00 %	90,40 %	90,40 %	86,40 %	6,98 %	5,88 %	5,88 %	2,90 %
2ª Exec.	92,00 %	90,40 %	90,40 %	86,40 %	6,98 %	5,88 %	5,88 %	2,90 %
3ª Exec.	92,00 %	90,40 %	90,40 %	86,40 %	6,98 %	5,88 %	5,88 %	2,90 %
Média	92,00 %	90,40 %	90,40 %	86,40 %	6,98 %	5,88 %	5,88 %	2,90 %

Tabela 14. Comparativo das métricas acurácia e precisão para cada algoritmo na Unidade C

Execuções	Métricas de Desempenho e Qualidade							
	Cobertura				Medida F			
	N.B. F.I.	N.B. H.	N.B. F.	SIM.	N.B. F.I.	N.B. H.	N.B. F.	SIM.
1ª Exec.	100 %	100 %	100 %	66,67 %	13,04 %	11,11 %	11,11 %	5,56 %
2ª Exec.	100 %	100 %	100 %	66,67 %	13,04 %	11,11 %	11,11 %	5,56 %
3ª Exec.	100 %	100 %	100 %	66,67 %	13,04 %	11,11 %	11,11 %	5,56 %
Média	100 %	100 %	100 %	66,67 %	13,04 %	11,11 %	11,11 %	5,56 %

Tabela 15. Comparativo das métricas cobertura e medida F para cada algoritmo na Unidade C

matriz de confusão) de cada algoritmo, o resultado de Naïve Bayes – Frequência Inversa foi superior aos resultados das demais abordagens, portanto, Naïve Bayes – Frequência Inversa foi o algoritmo que apresentou melhor desempenho e qualidade na classificação das sentenças dessa unidade.

Encerrando a primeira análise, a próxima unidade analisada foi a Unidade C. Também foram analisados os resultados das três classificações realizadas para cada algoritmo nessa unidade. Conforme é visto nas Tabelas 14 a 16, similaridade obteve um melhor desempenho do que os demais apenas na métrica de tempo de execução, bem como obteve os menores percentuais nas outras métricas. Já o Naïve Bayes – Frequência Inversa obteve as melhores porcentagens de acurácia, precisão, cobertura e medida F, seguido das abordagens Naïve Bayes – Híbrido e Naïve Bayes – Frequência. Sendo assim, o algoritmo Naïve Bayes – Frequência Inversa foi o melhor método de classificação para as sentenças dessa unidade. Contudo, é importante observar que, apesar do ótimo desempenho, Naïve Bayes – Frequência Inversa classificou, erroneamente, uma quantidade considerável de sentenças como “Evidência”, como é mostrado na Tabela 7.

Encerrando a análise comparativa dos algoritmos, foi feita uma análise por métrica utilizada. Verificando o gráfico da Figura 20, correspondente à métrica acurácia, é notável os ótimos desempenhos dos Naïve Bayes – Híbrido e Naïve Bayes – Frequência na Unidade A e o empate de ambos em todas unidades gestoras. Também é perceptível o empate entre Naïve Bayes – Frequência Inversa e Similaridade na Unidade A. Em média, Naïve Bayes – Frequência Inversa possui a melhor porcentagem de acurácia, consequentemente, é a melhor abordagem em termos de acurácia.

No gráfico apresentado na Figura 21 podemos observar o bom desempenho do Naïve Bayes – Frequência Inversa na maioria das unidades, apesar do péssimo desempenho na Unidade B. Em média, Naïve Bayes – Híbrido e Naïve

Bayes – Frequência foram melhores do que Naïve Bayes – Frequência Inversa. Mesmo assim, Naïve Bayes – Frequência Inversa é a melhor abordagem em termos de precisão, pois, como foi dito anteriormente, classificou erroneamente um número inferior de sentenças em relação aos demais algoritmos.

Observando o gráfico apresentado na **Figura 22**, concluímos o ótimo desempenho de Naïve Bayes – Híbrido e Naïve Bayes – Frequência, possuindo 100% em todas as unidades. Assim como é perceptível o baixo desempenho de similaridade na métrica cobertura. Portanto, Naïve Bayes – Híbrido e Naïve Bayes – Frequência, por possuírem as melhores porcentagens de cobertura, são os melhores algoritmos em termos desta métrica.

Analizando o gráfico da **Figura 23**, é notável a qualidade de Naïve Bayes – Híbrido e Naïve Bayes – Frequência, apesar de possuir a média harmônica um pouco inferior à de Naïve Bayes – Frequência Inversa na Unidade C. Já similaridade possui, em média, a menor porcentagem de medida F. Assim, Naïve Bayes – Híbrido e Naïve Bayes – Frequência possuem, em média, as melhores porcentagens de medida F, consequentemente, são os melhores algoritmos para esta medida.

Observando o gráfico da **Figura 24** fica evidente o ótimo desempenho do algoritmo de similaridade por possuir os menores tempos de execução em todas as unidades gestoras.

Na tentativa de descobrir o motivo de todos algoritmos terem classificado erroneamente uma quantidade considerável de evidências nas unidades B e C, foram analisadas as classificações de duas conformidades que foram consideradas como evidências por todas as abordagens: “INAUGURACAO DE CONJUNTO HABITACIONAL,CLINICA DE SAUDE E CEN” (Unidade B) e “LEVAR PACIENTES P/REALIZACAO DE EXAMES” (Unidade C). Para classificar a primeira sentença, foi considerado apenas um termo cujo radical é “saud”, sendo 06 e 09 as quantidades de amostras “Em Conformidade” e “Evidência”, respectivamente.

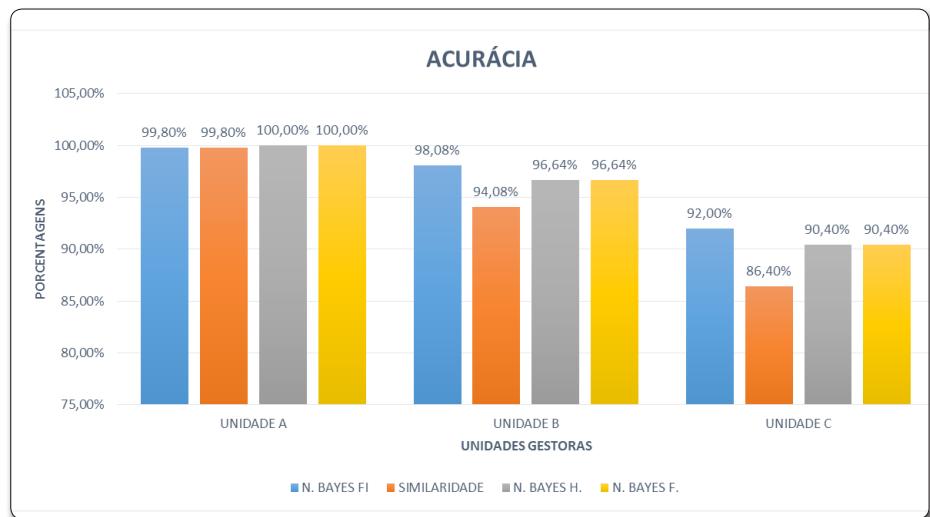


Figura 20. Gráfico da métrica acurácia

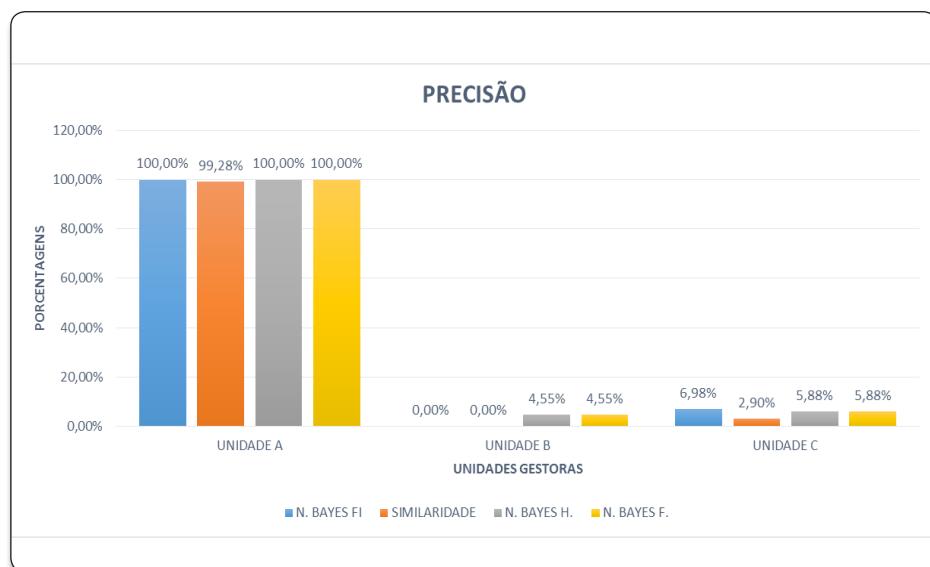


Figura 21. Gráfico da métrica precisão

Execuções	Métricas de Desempenho e Qualidade			
	Tempo de Execução			
	N.B. F.I.	N.B. H.	N.B. F.	SIM.
1ª Exec.	184,19 s	201,97 s	245,13 s	117,37 s
2ª Exec.	212,91 s	185,20 s	311,77 s	114,61 s
3ª Exec.	187,70 s	194,69 s	256,87 s	119,11 s
Média	194,93 s	193,95 s	271,26 s	117,03 s

Tabela 16. Comparativo da métrica tempo de execução para cada algoritmo na Unidade C

Mineração de texto: Análise comparativa de algoritmos

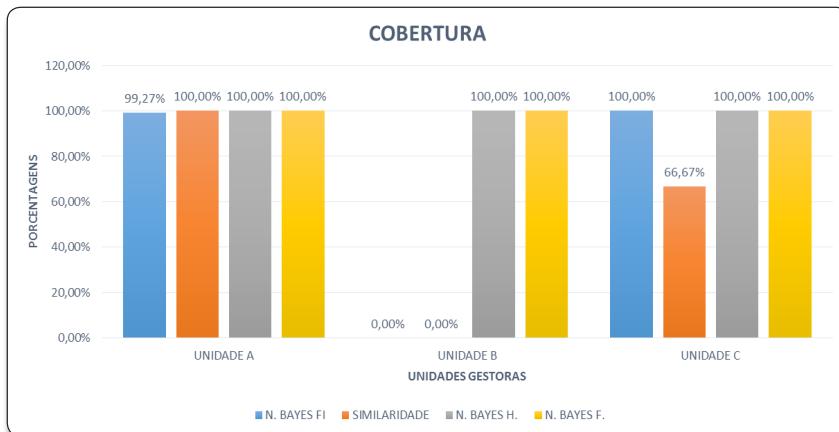


Figura 22. Gráfico da métrica cobertura

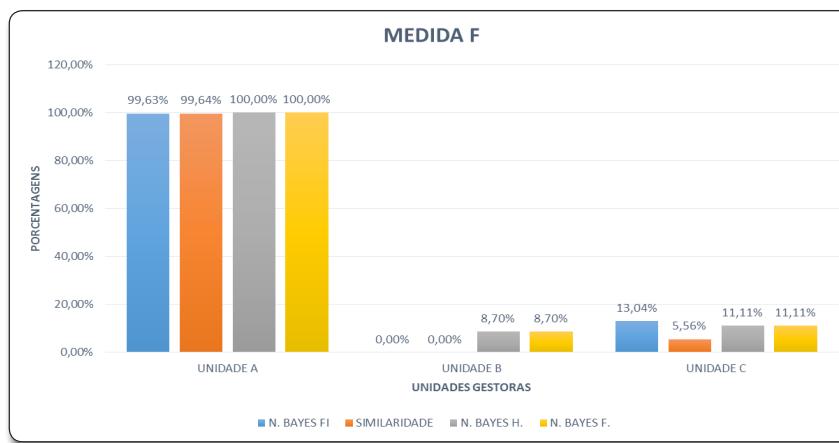


Figura 23. Gráfico da métrica medida F

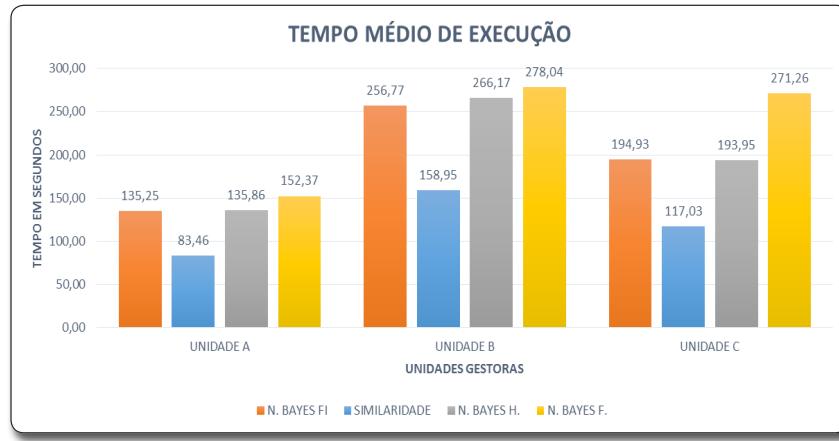


Figura 24. Gráfico da métrica tempo médio de execução

Já para a segunda sentença, foram considerados dois termos de radicais “pacient” e “exam”, sendo 01 a quantidade de “Em Conformidade” para o radical “pacient” e 06 a quantidade de “Evidência” para o radical “exam”. Portanto, o processo de Stemming influenciou na classificação errônea dessas sentenças, podendo até ter influenciado nas demais sentenças classificadas de forma errada.

Finalizando a análise, na maioria das métricas e unidades analisadas, conclui-se que Naïve Bayes – Frequência Inversa, para o contexto abordado neste artigo, foi o algoritmo que obteve melhor desempenho e qualidade para classificar sentenças, consequentemente, possibilitando melhores resultados para apoiar a decisão dos auditores na detecção de irregularidades no pagamento de diárias.

De posse do melhor algoritmo, esse pode ser utilizado para tornar mais efetivo o trabalho do auditor na identificação de irregularidades, auxiliando-o na tomada de decisão.

Autor



Breno Santana Santos

breno1005@hotmail.com

Mestrando em Ciência da Computação na linha de pesquisa em Engenharia de Software, Programa de Pós-graduação em Ciência da Computação (PROCC), Universidade Federal de Sergipe. Graduado em Bacharelado em Sistemas de Informação, Campus Professor Alberto Carvalho, Universidade Federal de Sergipe



Autor



Methanias Colaço Júnior

mjrse@hotmail.com

Possui doutorado pela Universidade Federal da Bahia. Atua especialmente na interface entre a Tecnologia da Informação e as estratégias de Gestão e Marketing. Atualmente é professor da Universidade Federal de Sergipe. Possui 1 livro publicado.



Você gostou deste artigo?

Dê seu voto em

www.devmedia.com.br/sqlmagazine/feedback

Ajude-nos a manter a qualidade da revista!



Oracle Data Guard com Fast-Start Failover

Aprenda a automatizar a ativação da base de contingência

Ambientes de banco de dados de missão crítica e que necessitam prover alta disponibilidade dependem de infraestruturas de hardware e software adequadas ao negócio. Nestes ambientes a indisponibilidade de um banco de dados pode resultar em grandes prejuízos para a corporação. Pensando nisso, a Oracle disponibiliza diversos recursos que ajudam a reduzir drasticamente o risco de indisponibilidade nos ambientes de banco de dados.

Dentre as principais soluções de alta disponibilidade (ou HA – *High Availability*) oferecidas pela Oracle, podemos citar duas consideradas chaves e que atuam cada uma em determinado cenário de desastre, ou seja, são complementares:

- **Oracle Real Application Clusters:** é formado por um cluster de duas ou mais máquinas que compartilham um (ou mais) storage(s), conforme a **Figura 1**. Esta solução garante a continuidade do negócio em caso de falha em algum dos servidores que compõem o cluster. Este modelo é bastante utilizado e muito eficiente na maioria das falhas mais comuns e costuma ser implementado localmente no data center, mas é possível a implementação de nós do cluster separados geograficamente, desde que a infraestrutura seja adequada e a distância não seja muito grande (latência de rede baixa);

- **Oracle Data Guard:** permite a replicação física (ou lógica) dos dados de uma instância de banco de dados para outra infraestrutura independente (vide **Figura 2**). O Data Guard costuma complementar o Real Application Clusters em algumas situações de falha que um ambiente RAC não poderia evitar. Alguns exemplos são falhas nos componentes de storage, catástrofes naturais, incêndios, etc. Com o Oracle Data Guard é possível manter servidores redundantes a milhares de quilômetros de distância, ou mesmo possuir diversos sites remotos recebendo a replicação de um banco principal e ainda permitir que o banco réplica possa ser consultado (*read only*) desde que seja adquirida a option Active Data Guard.

Fique por dentro

Este artigo apresenta o produto Oracle Data Guard utilizando a funcionalidade Fast-Start Failover para automatizar a recuperação do ambiente após uma falha. O texto abordará os benefícios e os pontos de atenção que devem ser observados ao montar a infraestrutura, concluindo com um exemplo prático da implementação de um ambiente virtual Data Guard. Neste exemplo, vamos ver desde a montagem dos componentes da infraestrutura, a instalação e configuração do sistema operacional Linux, passando pela configuração do database, replicação para standby, configuração do Fast-Start Failover, até os testes de funcionamento do produto. O conteúdo apresentado é útil ao trabalharmos na configuração de soluções de alta disponibilidade para nosso ambiente de produção. A importância da tecnologia apresentada também está no fato dela permitir automatizar a ativação da base de contingência permitindo minimizarmos o tempo que o banco permaneça fora do ar.

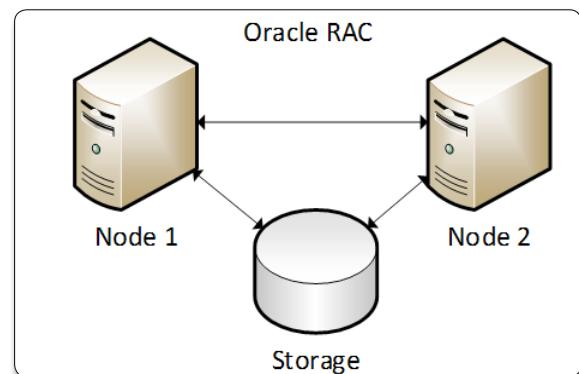


Figura 1. Modelo Oracle RAC

Neste artigo vamos explorar melhor o standby físico, mais especificamente usando a funcionalidade Fast-Start Failover do Data Guard. Para tanto, iremos explicar como simular um ambiente completo de contingencia usando máquinas virtuais, que será composto pelos seguintes componentes:

Oracle Data Guard com Fast-Start Failover

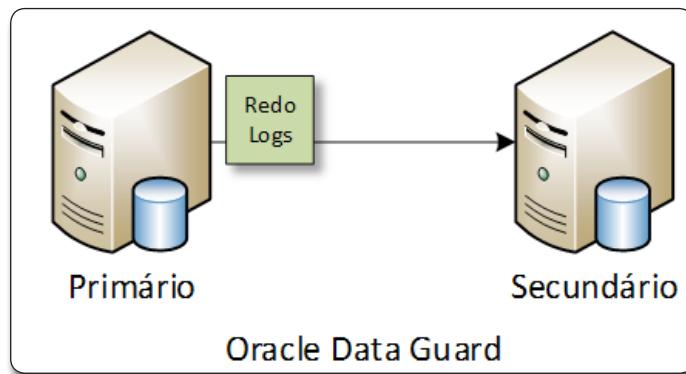


Figura 2. Modelo Oracle Data Guard

- **Data Guard:** é a base da infraestrutura. Precisamos de uma base de dados principal e pelo menos uma base de dados em standby, recebendo e aplicando transações;
- **Data Guard Broker:** é o gerenciador do Data Guard. Com ele é possível automatizar boa parte da configuração do Data Guard e é ele quem guarda informações chave das configurações de replicação;
- **Flashback Database:** garante rápida recuperação da base standby no momento de um failover, pois permite retornar o estado da base de dados a um momento anterior à falha para sincronizar com as transações da (nova) base de produção e iniciar a aplicação de redo logs;
- **FSFO Observer:** é um equipamento à parte das máquinas de produção e standby. Consiste de uma máquina com um Oracle Client instalado (independente de SO) que mantém um serviço DG-MGR CL (Data Guard Broker Command Line Interface) e “observa” o comportamento dos componentes do Data Guard. Em caso de falha do ambiente produtivo, avalia a necessidade de realizar um failover e instanciar novamente a base que falhou, seguindo algumas condições pré-configuradas. Este componente é considerado de baixa necessidade de recursos computacionais e independente de sistema operacional, necessitando apenas que seja homologado para executar o Oracle Client. A Oracle recomenda que a máquina Observer esteja o mais próximo possível das estações que acessarão o banco de dados, para que ele tenha a real visão de um cliente no caso de uma falha da infraestrutura. Além disso, não se recomenda que o observer fique nem no datacenter primário, nem no secundário, sob risco de ocorrer o chamado “Split Brain”, onde os dois databases iniciam como primários. A Figura 3 ilustra o modelo fast-start failover configurado com um observer.

Condições para um Fast-Start Failover

Imagine que o failover de uma base de produção é um processo drástico e a decisão de realizar ou não este failover deve ser bem pensada, mesmo que o procedimento ocorra automaticamente. Para isso, é necessário aplicar ajustes finos e definir exatamente em que situações espera-se que o banco de dados primário seja movido de um servidor para outro.

Em um ambiente com Fast-Start Failover configurado com os parâmetros default, o observer inicia o processo de failover somente se todas as seguintes condições forem satisfeitas:

- O Observer está rodando;
- Ambos o Observer e o Standby perdem contato com o Primário;
- O Observer tem contato com o Standby;
- As condições pré-definidas pelo DBA foram atingidas;
- O threshold de timeout foi atingido.

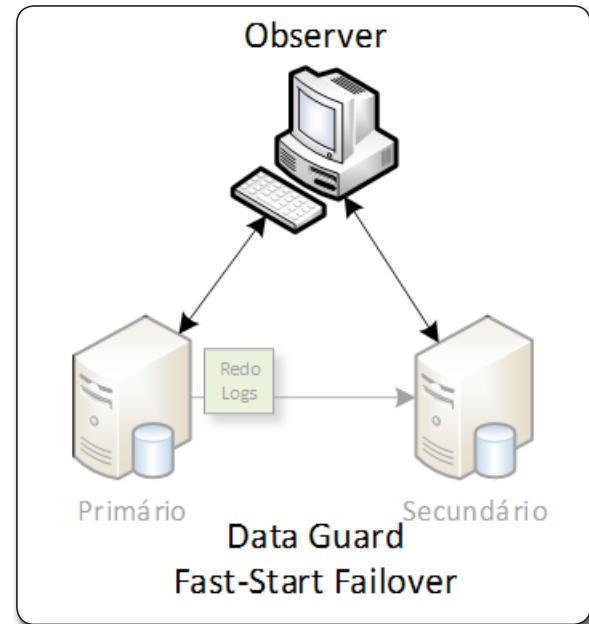


Figura 3. Modelo Fast-Start Failover

Condições para um failover configurado pelo usuário

Caso as configurações de failover automático não sejam adequadas ao ambiente a ser montado, é possível ainda que o usuário (DBA) defina algumas situações em que ele queira que o failover ocorra, a saber:

- Datafile off-line por falhas de I/O;
- Control file corrompido;
- Dicionário corrompido;
- Log Files inacessíveis por falhas de I/O;
- Archiver Stuck;
- Erros ORA-XXX (configurável);
- Uso da procedure DBMS_DG.INITIATE_FS_FAILOVER.

Roteiro de implementação do Fast-Start Failover

A implementação do Fast-Start Failover pode ser resumida com as seguintes atividades:

- A criação de uma base principal, Enterprise Edition, com Flashback database (vide BOX 1) habilitado;

BOX 1. Flashback database

Flashback database permite desfazer transações recentes em um banco de dados. Esta funcionalidade é essencial para que, após um failover, a antiga base produtiva tenha desfeitas as transações que não foram replicadas para a nova produção, sincronizando a base com um SCN válido na nova base de produção e habilitando a nova base standby a continuar recebendo transações de redo log sem a necessidade de nova carga completa.

- Um clone dos datafiles da base principal para outra máquina;
- A configuração do Data Guard utilizando o broker, garantindo que a replicação está ocorrendo corretamente;
- A configuração do observer em uma terceira máquina;
- Testes de failover.

Para configurar um ambiente de testes serão necessários:

- Hardware:
 - 2 VMs com OEL 6, 2GB de memória e 1 vCPU;
 - 1 VM com OEL 6, 1GB de memória e 1 vCPU.
- Software:
 - Oracle Database 12.1.0.2 EE;
 - Oracle Database 12.1.0.2 Client (para o observer).

Note que a implementação do ambiente foi realizada de forma simplificada, utilizando Oracle VM. Na prática, em ambiente físico, certamente serão necessárias configurações e passos adicionais para colocar o ambiente em funcionamento, porém os passos demonstrados aqui são completos o suficiente para a implementação do Data Guard.

Passo 1: Criação da máquina “Base”

Foi criada uma máquina virtual base, com um disco de 20 GB para as partições de sistema e outro disco de 20 GB montado com o nome “/u01”, onde serão instalados os binários e armazenados os datafiles do nosso banco de testes. Neste caso não precisamos de muito espaço, pois o objetivo não é armazenar dados, mas apresentar o funcionamento da solução.

Nesta máquina foi instalado o sistema operacional Oracle Enterprise Linux na versão 6.6. Antes de escolher o sistema operacional para sua máquina virtual, no entanto, verifique se ele é homologado para a versão de banco de dados que pretende instalar.

Após a instalação básica do Linux, foi instalado o pacote *oracle-rdbms-server-12cR1-preinstall-1.0-14.el6.x86_64*. Para uma instalação de testes, este pacote automatiza a instalação de pré-requisitos e a configuração de parâmetros de kernel e limites necessários para o Oracle Database 12c. Além disso, este pacote cria automaticamente o usuário Oracle. Depois de instalado, só é necessário redefinir a senha do usuário com o comando *passwd oracle*. Em produção, recomenda-se conferir os parâmetros definidos e ajustar em caso de necessidade.

Pronto, nossa máquina Base está instalada e configurada adequadamente, pronta para ser replicada.

Passo 2: Criação da máquina “Primária”

A máquina primária é a máquina considerada principal no ambiente Data Guard.

No nosso caso, é um clone da máquina Base criada anteriormente. Vale ressaltar que as máquinas não precisam necessariamente ser taxadas como Primária e Secundária, pois em um ambiente ideal, as duas máquinas deveriam ser idênticas, e a base produtiva read/write poderia estar em qualquer um dos dois “lados” do Data Guard, de forma transparente. Neste exemplo chamaremos de Primária (dg1) e Secundária (dg2) somente para facilitar a identificação de cada uma na montagem do ambiente.

Ao clonar máquinas para este ambiente, devemos nos atentar para os seguintes detalhes:

1. Um clone “linkado” será mais rápido e utilizará menos espaço em disco;
2. Ao clonar a máquina, sempre devemos selecionar a opção reiniciar o MAC Address;
3. No primeiro startup da máquina, limpe as linhas do arquivo */etc/udev/rules.d/70-persistent-net.rules* referentes à(s) interface(s) de rede da máquina clonada. Remova também as linhas que definem UID e MACADDR no arquivo */etc/sysconfig/network-scripts/ifcfg-ethX*;
4. Ajuste o nome da máquina em */etc/sysconfig/network* e reinicie a máquina;
5. Após o reboot, ajuste informações de IP caso necessário.

Após o clone, o primeiro passo é instalar o Oracle Database 12c 12.1.0.2 Enterprise Edition. Neste caso selecionamos a feature Multitenant, mas como ela não é escopo do artigo, não faria diferença se fosse selecionada outra opção. Na verdade, durante a instalação, a única opção que nos importa mesmo é que seja selecionada a instalação do Enterprise Edition, pois o Standard Edition não oferece a feature Data Guard.

Depois da instalação do binário, deve-se proceder com a criação do database e a configuração do listener, sem mistérios. No entanto, ainda não é a hora de configurar a inicialização automática da base.

```

[oracle@dg1 ~]$ sqlplus / as sysdba
SQL*Plus: Release 12.1.0.2.0 Production on Thu Oct 15 17:57:47 2015
Copyright (c) 1982, 2014, Oracle. All rights reserved.

Connected to an idle instance.

SQL> startup
ORACLE instance started.

Total System Global Area 713031680 bytes
Fixed Size                2928488 bytes
Variable Size              532676760 bytes
Database Buffers          171966464 bytes
Redo Buffers               5459968 bytes
Database mounted.
Database opened.
SQL> alter pluggable database all open;

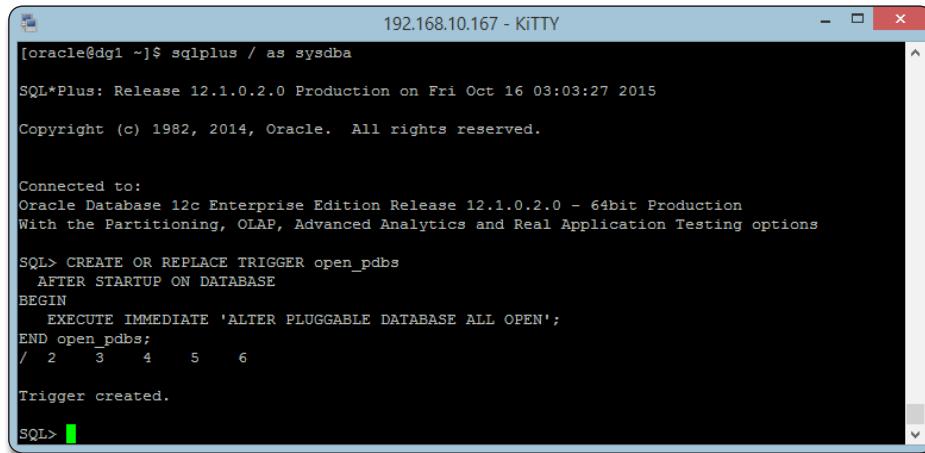
Pluggable database altered.

SQL>

```

Figura 4. Iniciando base primária

Oracle Data Guard com Fast-Start Failover



```
[oracle@dg1 ~]$ sqlplus / as sysdba
SQL*Plus: Release 12.1.0.2.0 Production on Fri Oct 16 03:03:27 2015
Copyright (c) 1982, 2014, Oracle. All rights reserved.

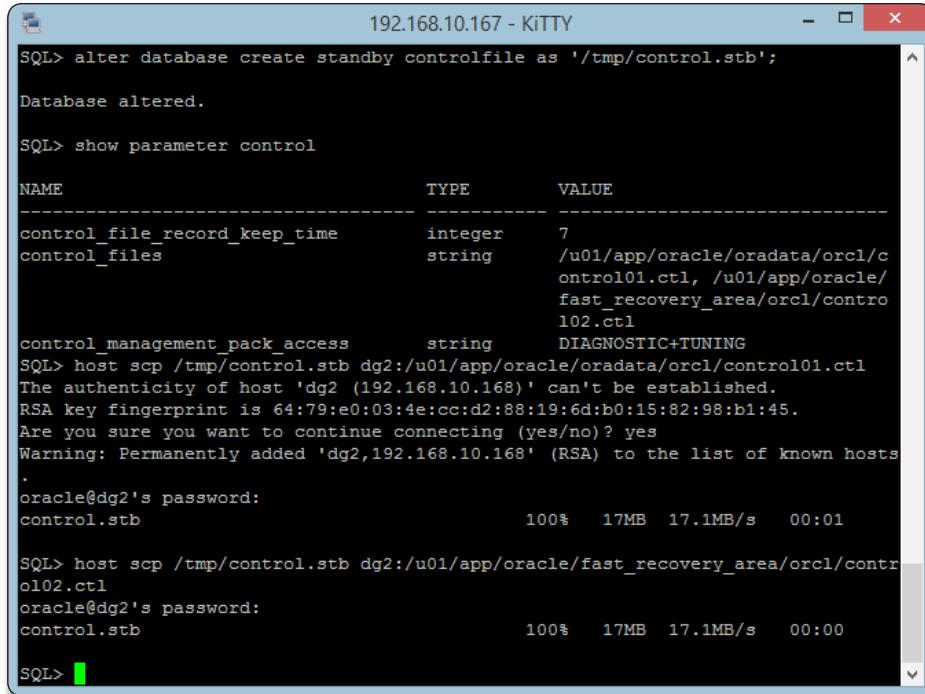
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing options

SQL> CREATE OR REPLACE TRIGGER open_pdbs
  AFTER STARTUP ON DATABASE
BEGIN
  EXECUTE IMMEDIATE 'ALTER PLUGGABLE DATABASE ALL OPEN';
END open_pdbs;
/ 2 3 4 5 6

Trigger created.

SQL>
```

Figura 5. Trigger de inicialização dos PDBs

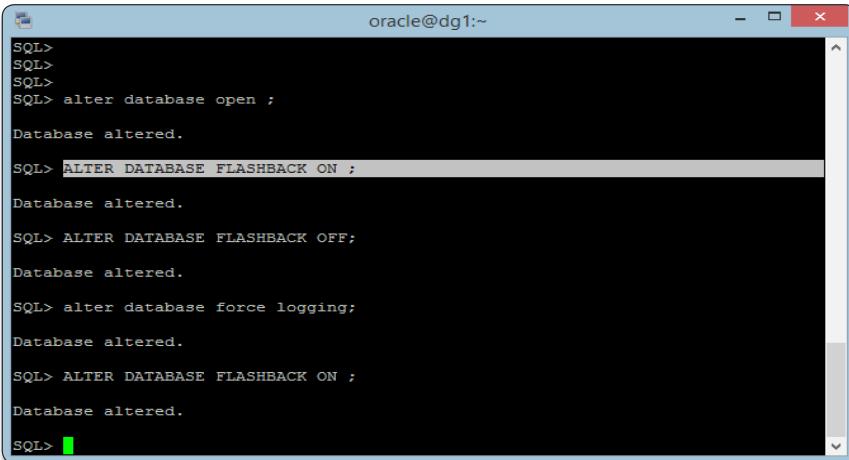


```
192.168.10.167 - KITTY
SQL> alter database create standby controlfile as '/tmp/control.stb';
Database altered.

SQL> show parameter control
NAME          TYPE        VALUE
-----
control_file_record_keep_time    integer     7
control_files                   string      /u01/app/oracle/oradata/orcl/control01.ctl, /u01/app/oracle/fast_recovery_area/orcl/control02.ctl
control_management_pack_access   string      DIAGNOSTIC+TUNING
SQL> host scp /tmp/control.stb dg2:/u01/app/oracle/oradata/orcl/control01.ctl
The authenticity of host 'dg2 (192.168.10.168)' can't be established.
RSA key fingerprint is 64:79:e0:03:4e:cc:d2:88:19:6d:b0:15:82:98:b1:45.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'dg2,192.168.10.168' (RSA) to the list of known hosts .
oracle@dg2's password:
control.stb                         100%  17MB  17.1MB/s  00:01

SQL> host scp /tmp/control.stb dg2:/u01/app/oracle/fast_recovery_area/orcl/control02.ctl
oracle@dg2's password:
control.stb                         100%  17MB  17.1MB/s  00:00
SQL>
```

Figura 6. Criando standby control file



```
oracle@dg1:~
SQL>
SQL>
SQL>
SQL> alter database open ;
Database altered.

SQL> ALTER DATABASE FLASHBACK ON ;
Database altered.

SQL> ALTER DATABASE FLASHBACK OFF;
Database altered.

SQL> alter database force logging;
Database altered.

SQL> ALTER DATABASE FLASHBACK ON ;
Database altered.

SQL>
```

Figura 7. Habilitando Flashback e Force logging

Passo 3: Criação da máquina secundária

Neste momento pouparemos um tempo usando uma máquina virtual. Usaremos um clone da máquina primária, que já possui o SO instalado e configurado, o Oracle Database instalado e ainda teremos uma cópia dos datafiles do banco primário.

Caso não fosse possível realizar este clone, teríamos que instalar SO, o Oracle Database e realizar um clone da base primária usando o comando *rman "duplicate for standby from active database"*.

Na máquina clonada, não esqueça de seguir as recomendações referentes a mac address e ajustes de rede já discutidos anteriormente. Com isso, neste momento já temos uma máquina “base” e duas máquinas com banco de dados Oracle.

Passo 4: Ajustes para o Data Guard

Agora entramos efetivamente na configuração do Data Guard. Para começar, vamos realizar uma configuração de replicação básica utilizando as duas bases clonadas nos passos anteriores.

Dito isso, iniciamos a base “primária”, conforme a Figura 4.

Para que no próximo startup da base de dados não seja necessário iniciar o Pluggable DataBase (PDB), criaremos uma trigger que fará este trabalho (vide Figura 5).

Em seguida, criamos um control file para ser usado no standby e copiamos ele para a máquina secundária, substituindo o control file original (Figura 6).

Para garantir que todas as transações da base primária sejam registradas em redo logs, é necessário habilitar o force logging na base de dados. E para utilizar o Fast-Start Failover é necessário habilitar o Flashback Database. Estes dois procedimentos devem ser executados tanto na base primária quanto na secundária, como demonstrado na Figura 7.

Agora devemos ajustar o TNSNAMES de ambas as máquinas para permitir a conexão individual a cada uma das instâncias. Note que as entradas *ORCL_PRD* e *ORCL_DTG* direcionam as conexões especificamente para as bases existentes respectivamente nas máquinas DG1 e DG2. Já a entrada “*ORCL*” é a que deve ser distribuída para os usuários de aplicação,

pois realiza o trabalho de “failover” de conexão caso o Data Guard tenha que ser acionado, ou seja, não importa em que lado a base produtiva esteja, o TNSNAMES ORCL sempre atenderá à produção. Podemos ver na **Figura 8** um exemplo de arquivo TNSNAMES criado para um ambiente Data Guard.

Após essa etapa, o teste de conexão é muito importante para garantir que todas as entradas TNSNAMES são funcionais, caso contrário teremos problemas durante a configuração do Data Guard no Broker. A **Figura 9** apresenta uma forma de testar a configuração recém realizada.

Nota

Sempre teste as entradas TNSNAMES conectando efetivamente na máquina. O TNSPING somente testa se o listener está no ar. Não testa se SERVICE_NAME ou SID estão ativos, nem se o database está pronto para receber conexões.

Passo 5: Configuração do Data Guard

Neste momento iniciamos a base standby na máquina Secundária em modo *mount*. Neste modo a memória SGA é alocada, os arquivos de controle são lidos e os datafiles estão prontos para receber transações via aplicação de logs.

A criação de standby logfiles em ambos os databases é necessária para que a base standby consiga receber e aplicar transações da base primária sem a necessidade de geração de archivelogs. Desse modo, as transações geradas no redo log da produção são aplicadas quase que simultaneamente na base standby. A **Figura 10** demonstra a criação de standby redo logs.

Nota

O tamanho dos standby redo logs deve sempre ser correspondente ao tamanho dos redo logs existentes na base primária. A quantidade recomendada é pelo menos a quantidade de grupos existentes na produção + 1 em ambientes single. Em ambientes RAC deve-se usar a seguinte equação:

(número máximo de redo log groups por thread + 1) * número máximo de threads.

Em seguida, ajustamos os parâmetros *DG_BROKER_START* e *DB_UNIQUE_*

NAME de cada instância. Neste momento também podemos definir o parâmetro *STANDBY_FILE_MANAGEMENT* para AUTO, para que novos datafiles sejam criados automaticamente no standby quando são criados na produção.

O parâmetro *DG_BROKER_START* deve ser alterado para TRUE, para que o Data

Guard Broker seja iniciado. Já o parâmetro *DB_UNIQUE_NAME* deve identificar cada instância com um nome único no ambiente. Neste caso, colocamos *ORCL_PRD* e *ORCL_DTG*, respectivamente. Para alterar este último parâmetro, no entanto, é necessário definir “scope=spfile”, pois ele não é dinâmico. Feito isso, reinicie

```

ORCL =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = DG1)(PORT = 1521))
    (ADDRESS = (PROTOCOL = TCP)(HOST = DG2)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = orcl)
    )
  )

ORCL_PRD =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = DG1)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SID = orcl)
    )
  )

ORCL_DTG =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = DG2)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SID = orcl)
    )
  )

```

Figura 8. Configuração de arquivo TNSNAMES

```

[oracle@dgl ~]$ sqlplus system@orcl
SQL*Plus: Release 12.1.0.2.0 Production on Fri Oct 16 03:05:38 2015
Copyright (c) 1982, 2014, Oracle. All rights reserved.

Enter password:
Last Successful login time: Fri Oct 16 2015 03:05:13 -03:00

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing options

SQL> exit
Disconnected from Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing options

[oracle@dgl ~]$ sqlplus system@orcl_prd
SQL*Plus: Release 12.1.0.2.0 Production on Fri Oct 16 03:05:46 2015
Copyright (c) 1982, 2014, Oracle. All rights reserved.

Enter password:
Last Successful login time: Fri Oct 16 2015 03:05:40 -03:00

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing options

SQL> exit
Disconnected from Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing options

```

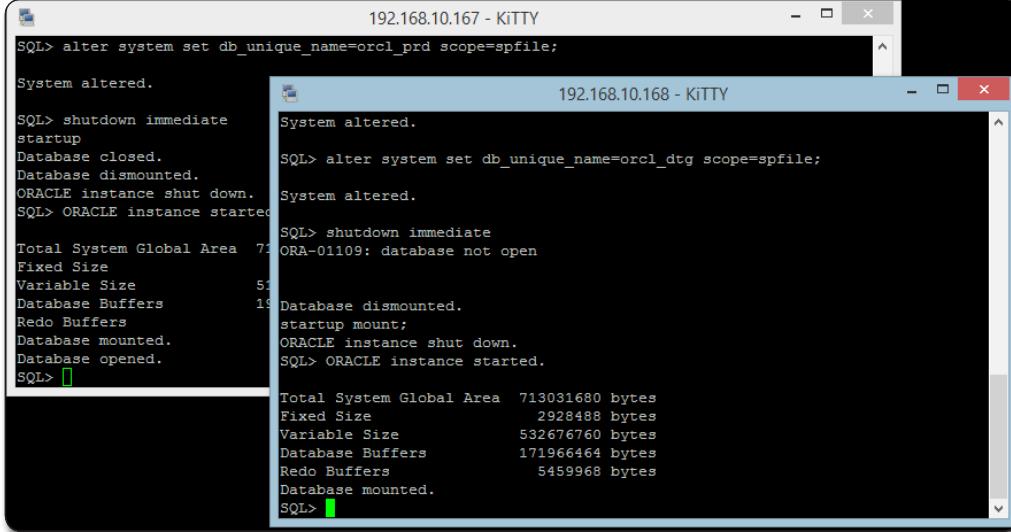
Figura 9. Testando TNSNAMES

Oracle Data Guard com Fast-Start Failover



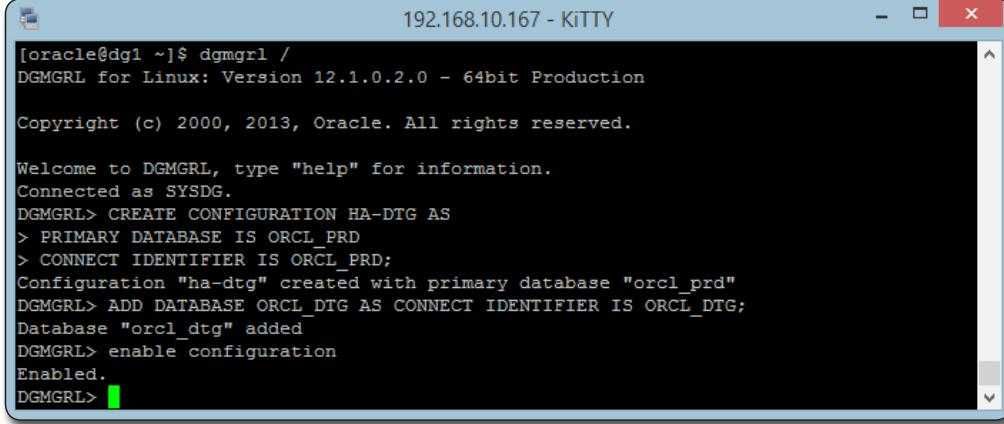
```
oracle@dg1:~  
SQL> ALTER DATABASE ADD STANDBY LOGFILE GROUP 4 '/u01/app/oracle/oradata/orcl redo04  
.log' size 200m;  
  
Database altered.  
  
SQL> ALTER DATABASE ADD STANDBY LOGFILE GROUP 5 '/u01/app/oracle/oradata/orcl redo05  
.log' size 200m;  
  
Database altered.  
  
SQL> ALTER DATABASE ADD STANDBY LOGFILE GROUP 6 '/u01/app/oracle/oradata/orcl redo06  
.log' size 200m;  
  
Database altered.  
  
SQL>
```

Figura 10. Adicionando standby logfiles



```
192.168.10.167 - KiTTY  
SQL> alter system set db_unique_name=orcl_prd scope=spfile;  
  
System altered.  
  
SQL> shutdown immediate  
startup  
Database closed.  
Database dismounted.  
ORACLE instance shut down.  
SQL> ORACLE instance started.  
  
Total System Global Area 71  
Fixed Size 2928488 bytes  
Variable Size 51  
Database Buffers 19  
Redo Buffers 5  
Database mounted.  
Database opened.  
SQL>  
  
192.168.10.168 - KiTTY  
System altered.  
  
SQL> alter system set db_unique_name=orcl_dtg scope=spfile;  
  
System altered.  
  
SQL> shutdown immediate  
ORA-01109: database not open  
  
Database dismounted.  
startup mount  
ORACLE instance shut down.  
SQL> ORACLE instance started.  
  
Total System Global Area 713031680 bytes  
Fixed Size 2928488 bytes  
Variable Size 532676760 bytes  
Database Buffers 171966464 bytes  
Redo Buffers 5459968 bytes  
Database mounted.  
SQL>
```

Figura 11. Alterando parâmetros e reiniciando instâncias



```
[oracle@dg1 ~]$ dgmgrl /  
DGMGRL for Linux: Version 12.1.0.2.0 - 64bit Production  
  
Copyright (c) 2000, 2013, Oracle. All rights reserved.  
  
Welcome to DGMGRL, type "help" for information.  
Connected as SYSDG.  
DGMGRL> CREATE CONFIGURATION HA-DTG AS  
> PRIMARY DATABASE IS ORCL_PRD  
> CONNECT IDENTIFIER IS ORCL_PRD;  
Configuration "ha-dtg" created with primary database "orcl_prd"  
DGMGRL> ADD DATABASE ORCL_DTG AS CONNECT IDENTIFIER IS ORCL_DTG;  
Database "orcl_dtg" added  
DGMGRL> enable configuration  
Enabled.  
DGMGRL>
```

Figura 12. Criando configuração do Broker.

as instâncias, como indicado na Figura 11.

Agora que ajustamos os parâmetros necessários para o Data Guard, podemos configurá-lo efetivamente. Para isso, usamos o aplicativo DGMGRL, que está disponível em `$ORACLE_HOME/bin`. Se o ambiente estiver parametrizado adequadamente, é só chamá-lo via linha de comando.

A partir desse aplicativo, configuraremos o Broker usando o comando `create configuration`. Em seguida, adicionaremos o database secundário e habilitaremos toda a configuração realizada usando o comando `enable configuration`, como pode ser verificado na Figura 12.

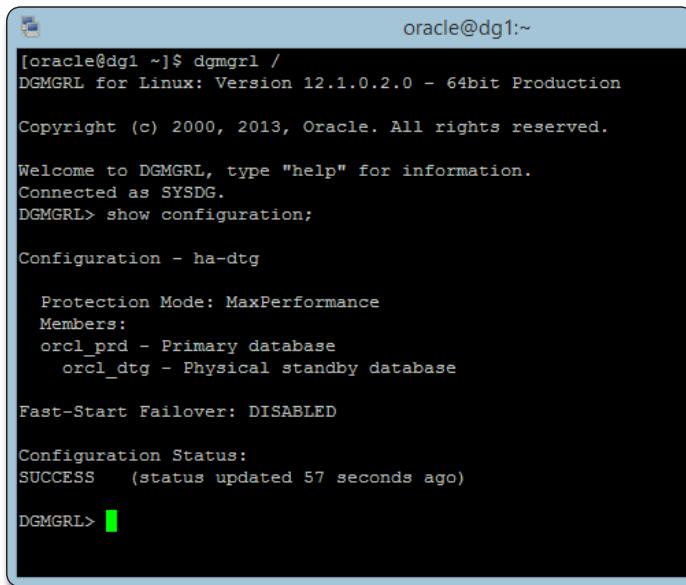
Ao habilitar a configuração, os parâmetros necessários para iniciar a replicação das transações entre base primária e secundária são ajustados automaticamente no spfile das duas bases de dados. Em muitos casos, estas definições são suficientes para termos o ambiente de alta disponibilidade funcional.

Caso tudo esteja configurado corretamente, neste momento a

base primária se comunicará com a base secundária, identificando quais transações devem ser transferidas, e fará a sincronização automática entre elas.

Após algum tempo, se consultarmos a configuração, receberemos o status de base sincronizada, isto é, todas as transações geradas no redo log de produção foram aplicadas na base standby. Os comandos necessários podem ser conferidos na Figura 13.

Podemos ainda conferir o status individual de cada instância, em mais detalhes, usando o comando `show database`. Este comando mostrará a role do database e seu estado. No caso da base standby,



```
oracle@dg1:~$ dgmgrl /
DGMGRl for Linux: Version 12.1.0.2.0 - 64bit Production

Copyright (c) 2000, 2013, Oracle. All rights reserved.

Welcome to DGMGRl, type "help" for information.
Connected as SYSDG.
DGMGRl> show configuration;

Configuration - ha-dtg

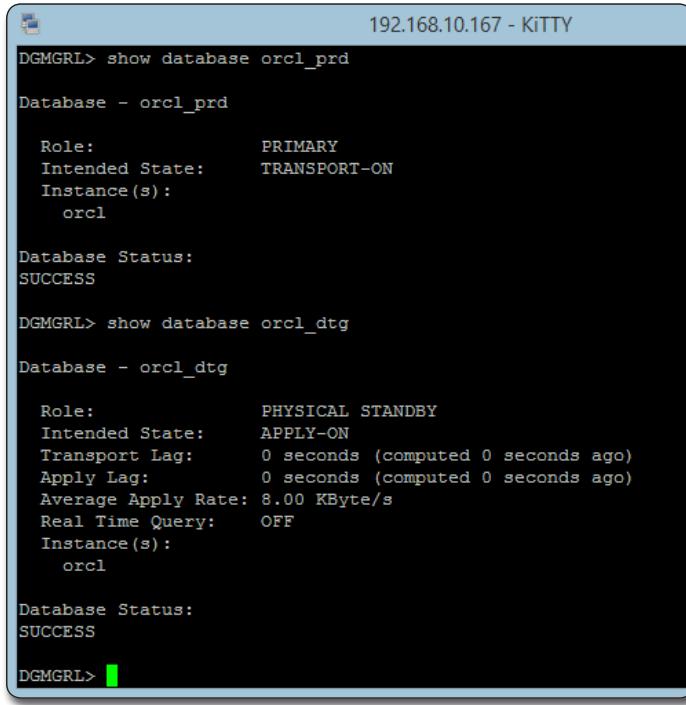
  Protection Mode: MaxPerformance
  Members:
    orcl_prd - Primary database
    orcl_dtg - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS (status updated 57 seconds ago)

DGMGRl>
```

Figura 13. Verificando a configuração



```
192.168.10.167 - KiTTY
DGMGRl> show database orcl_prd

Database - orcl_prd

  Role:          PRIMARY
  Intended State: TRANSPORT-ON
  Instance(s):
    orcl

  Database Status:
  SUCCESS

DGMGRl> show database orcl_dtg

Database - orcl_dtg

  Role:          PHYSICAL STANDBY
  Intended State: APPLY-ON
  Transport Lag:  0 seconds (computed 0 seconds ago)
  Apply Lag:    0 seconds (computed 0 seconds ago)
  Average Apply Rate: 8.00 KByte/s
  Real Time Query: OFF
  Instance(s):
    orcl

  Database Status:
  SUCCESS

DGMGRl>
```

Figura 14. Verificando mais detalhes do banco de dados

o comando apresenta ainda quão defasada está a base em relação à produção. Alguns exemplos podem ser conferidos na Figura 14.

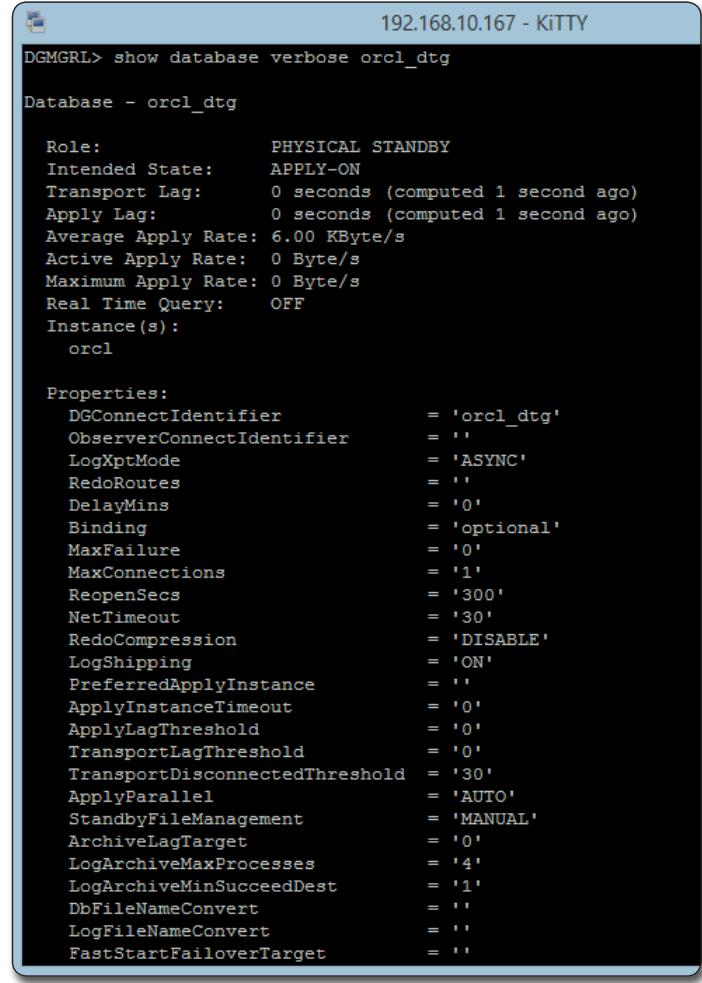
Caso seja preciso conferir todos os parâmetros de configuração do broker do database, podemos utilizar o parâmetro `verbose`, conforme exposto na Figura 15.

Passo 6: Teste de switchover

Antes de continuar a configuração é prudente realizar alguns comandos “switchover” para conferir se o Data Guard está funcionando corretamente. Para isso, novamente usaremos o DGMGRL, apesar de poder ser feito de outras formas como, por

Listagem 1. Configuração de uma entrada do listener.ora da máquina primária.

```
01 SID_LIST_LISTENER =
02 (SID_LIST=
03   (SID_DESC=
04     (GLOBAL_DBNAME = ORCL_PRD_DGMGRl)
05     (ORACLE_HOME = /u01/app/oracle/product/12.1.0.1/dbhome)
06     (SID_NAME= orcl)
07   )
08 )
```



```
192.168.10.167 - KiTTY
DGMGRl> show database verbose orcl_dtg

Database - orcl_dtg

  Role:          PHYSICAL STANDBY
  Intended State: APPLY-ON
  Transport Lag:  0 seconds (computed 1 second ago)
  Apply Lag:    0 seconds (computed 1 second ago)
  Average Apply Rate: 6.00 KByte/s
  Active Apply Rate: 0 Byte/s
  Maximum Apply Rate: 0 Byte/s
  Real Time Query: OFF
  Instance(s):
    orcl

  Properties:
    DGConnectIdentifier           = 'orcl_dtg'
    ObserverConnectIdentifier     = ''
    LogXptMode                   = 'ASYNC'
    RedoRoutes                   = ''
    DelayMins                     = '0'
    Binding                       = 'optional'
    MaxFailure                   = '0'
    MaxConnections                = '1'
    ReopenSecs                    = '300'
    NetTimeout                    = '30'
    RedoCompression               = 'DISABLE'
    LogShipping                   = 'ON'
    PreferredApplyInstance        = ''
    ApplyInstanceTimeout          = '0'
    ApplyLagThreshold             = '0'
    TransportLagThreshold         = '0'
    TransportDisconnectedThreshold = '30'
    ApplyParallel                 = 'AUTO'
    StandbyFileManagement        = 'MANUAL'
    ArchiveLagTarget              = '0'
    LogArchiveMaxProcesses        = '4'
    LogArchiveMinSucceedDest      = '1'
    DbFileNameConvert              = ''
    LogFileNameConvert              = ''
    FastStartFailoverTarget        = ''
```

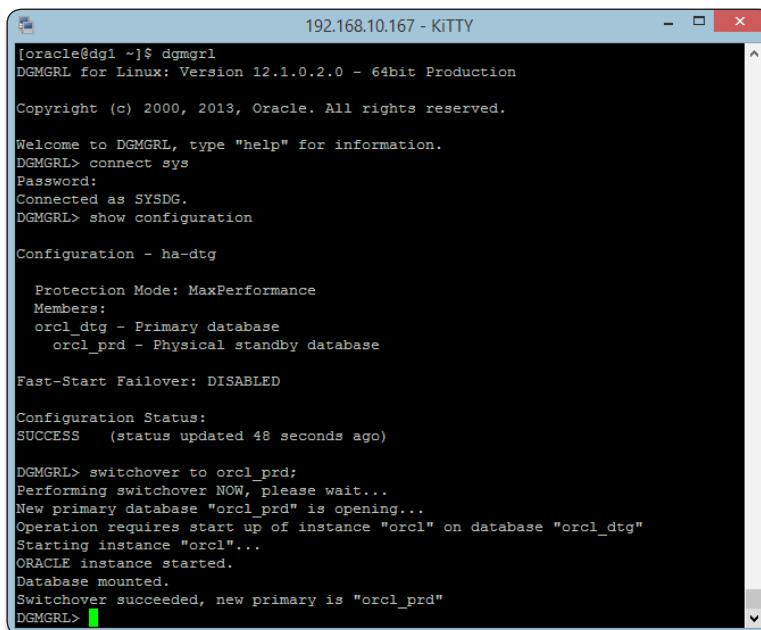
Figura 15. Verificando os parâmetros

Oracle Data Guard com Fast-Start Failover

exemplo, com o Enterprise Manager Cloud Control, caso estivesse disponível.

Antes de iniciar o switchover, verifique se o serviço <DB_UNIQUE_NAME>_DGMGRL está registrado no listener. Isso pode ser feito utilizando o comando `lsnrctl status` ou analisando o arquivo de configuração `listener.ora`. No exemplo, temos que configurar a entrada apresentada na **Listagem 1** do arquivo `listener.ora` da máquina primária.

Na máquina secundária devemos repetir a configuração substituindo `ORCL_PRD_DGMGRL` por `ORCL_DTG_DGMGRL`.



```
192.168.10.167 - KITTY
[oracle@dg1 ~]$ dgmgrl
DGMGRL for Linux: Version 12.1.0.2.0 - 64bit Production
Copyright (c) 2000, 2013, Oracle. All rights reserved.

Welcome to DGMGRL, type "help" for information.
DGMGRL> connect sys
Password:
Connected as SYSDG.
DGMGRL> show configuration

Configuration - ha-dtg

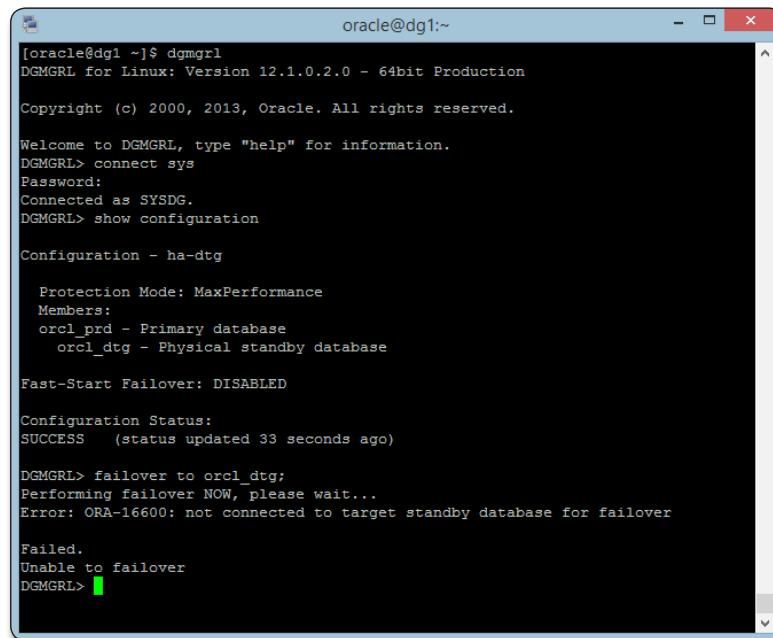
  Protection Mode: MaxPerformance
  Members:
    orcl_dtg - Primary database
    orcl_prd - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS (status updated 48 seconds ago)

DGMGRL> switchover to orcl_prd;
Performing switchover NOW, please wait...
New primary database "orcl_prd" is opening...
Operation requires start up of instance "orcl" on database "orcl_dtg"
Starting instance "orcl"...
ORACLE instance started.
Database mounted.
Switchover succeeded, new primary is "orcl_prd"
DGMGRL>
```

Figura 16. Testando switchover



```
oracle@dg1:~
[oracle@dg1 ~]$ dgmgrl
DGMGRL for Linux: Version 12.1.0.2.0 - 64bit Production
Copyright (c) 2000, 2013, Oracle. All rights reserved.

Welcome to DGMGRL, type "help" for information.
DGMGRL> connect sys
Password:
Connected as SYSDG.
DGMGRL> show configuration

Configuration - ha-dtg

  Protection Mode: MaxPerformance
  Members:
    orcl_prd - Primary database
    orcl_dtg - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS (status updated 33 seconds ago)

DGMGRL> failover to orcl_dtg;
Performing failover NOW, please wait...
Error: ORA-16600: not connected to target standby database for failover

Failed.
Unable to failover
DGMGRL>
```

Figura 17. Falha de teste de failover

Caso tenha que alterar algo no arquivo `listener.ora`, não esqueça de reiniciar o listener.

Lembro que este é o nome de serviço padrão que o Broker configura automaticamente. É possível que o nome de serviço necessário no seu ambiente seja diferente. Para confirmar o nome de serviço, pode-se usar o comando `show database verbose <DB_UNIQUE_NAME>` e verificar o parâmetro `StaticConnect-Identifier` da configuração.

Outro detalhe importante para que o switchover ocorra corretamente é garantir que se conecte ao DGMGRL com um usuário `SYSDBA` (no caso o `SYS`) e utilize uma senha válida para ambos os databases do Data Guard.

O comando switchover realiza a transição de serviço de um database para o outro de modo que não sejam perdidas transações. Neste processo o Data Guard finaliza a base de produção corrente, garante que todas as transações foram enviadas para a base de contingência e inicia o standby como `read/write`. Seguindo os passos descritos anteriormente, espera-se ainda que a base que estava em produção suba automaticamente como standby, pronta para receber as transações da nova base produtiva. Na **Figura 16** podemos conferir o comando switchover sendo executado pela linha de comando do DGMGRL.

Ao final, se tudo der certo, teremos a base `ORCL_DTG` como principal `read/write` e a base `ORCL_PRD` como standby no modo `mount` ou `read only`.

Feito o switchover corretamente, podemos então realizar novo switchover para voltar a base principal para `ORCL_PRD`. É recomendado executar o switchover quantas vezes forem necessárias até que se tenha a configuração ideal do ambiente.

Passo 7: Teste de Failover

O comando de failover realiza a transição de serviço de um database para o outro de modo forçado. Esse tipo de transição de modo forçado é necessário quando a base de produção corrente falhou e não está disponível para que se recupere as últimas transações e se baixe a instância de modo normal – ocorre um `shutdown abort` ou a simples queda “fria” do serviço.

Este comando deve sempre ser executado conectando-se o DGMGRL ao database que está atualmente em standby, caso contrário o comando de failover retornará o erro apresentado na **Figura 17**.

Ao executar o comando failover simulamos a real necessidade de transição de serviço forçada. Desse modo o banco de produção é derrubado e o banco standby assume a produção subindo automaticamente. A **Figura 18** apresenta esse procedimento sendo executado.

Após a execução do comando, o banco que foi interrompido abruptamente deve ser reconstruído. Para isso, realize-se o rollback das transações que não foram aplicadas no banco standby para então iniciar a execução das transações

de redo do novo banco de produção. Neste momento o Flashback Database entra em cena e realiza este trabalho. Caso não estivesse habilitado seria preciso uma nova carga clone no banco de produção para retomar a aplicação de redo logs no novo standby.

A reconstrução pode ser realizada com a execução do comando `reinstate database <DB_UNIQUE_NAME>` no DGMGRL, mas antes é necessário iniciar o database em modo mount (vide Figura 19).

```
oracle@dg2:/u01/app/oracle/fast_recovery_area/ORCL_DTG/archivelog
```

```
Welcome to DGMGRL, type "help" for information.
DGMGRL> connect sys
Password:
Connected as SYSDG.
DGMGRL> failover to orcl_dtg;
Performing failover NOW, please wait...
Failover succeeded, new primary is "orcl_dtg"
DGMGRL> show configuration

Configuration - ha-dtg

Protection Mode: MaxPerformance
Members:
  orcl_dtg - Primary database
    orcl_prd - Physical standby database (disabled)
      ORA-16661: the standby database needs to be reinstated

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS  (status updated 93 seconds ago)

DGMGRL>
```

Figura 18. Teste de failover utilizando o DGMGRL

```
oracle@dg2:/u01/app/oracle/fast_recovery_area/ORCL_DTG/archivelog
```

```
Welcome to DGMGRL, type "help" for information.
DGMGRL> connect sys
Password:
Connected as SYSDG.
DGMGRL> show configuration

Configuration - ha-dtg

Protection Mode: MaxPerformance
Members:
  orcl_dtg - Primary database
    orcl_prd - Physical standby database (disabled)
      ORA-16661: the standby database needs to be reinstated

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS  (status updated 17 seconds ago)

DGMGRL> reinstate database orcl_prd;
Reinstating database "orcl_prd", please wait...
Reinstatement of database "orcl_prd" succeeded
DGMGRL>
```

Variable Size	511705240 bytes
Database Buffers	192937984 bytes
Redo Buffers	5459968 bytes
Database mounted.	

Figura 19. Reconstruindo a base standby após failover

Passo 8: Configuração do Fast-Start Failover

Agora que já temos o Data Guard configurado com o DG Broker e o Flashback Database habilitado, é hora de colocarmos o Observer para gerenciar o estado dos bancos de dados. Para isso, realizamos um novo clone da máquina base, que já possui o Linux instalado. Novamente, não esqueça de definir os ajustes de rede necessários, conforme já explicado.

Chamaremos esta nova máquina de “observer” e ela será responsável por observar o funcionamento do Data Guard configurado nas máquinas dg1 e dg2. Em observer, instalaremos o Oracle Client 12c como usual e copiaremos o arquivo *tnsnames.ora* de um dos servidores do Data Guard para a pasta *\$ORACLE_HOME/network/admin*. Isso é necessário para que o observer tenha acesso a todas as bases de dados envolvidas.

Em seguida, execute o DGMGRL, se conecte ao Data Guard com o usuário SYSDBA e inicie o observer com o seguinte comando (vide Figura 20):

```
Start observer;
```

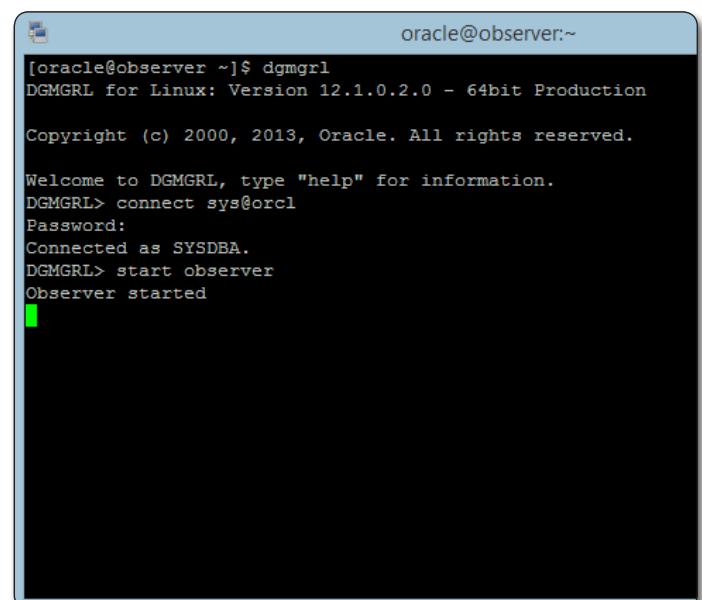
O prompt ficará travado após essa chamada. Para evitar que isso aconteça podemos utilizar o comando apresentado a seguir, o qual possibilita iniciar o observer e direcionar as mensagens de log para um arquivo (vide Figura 21):

```
nohup dmgmgr sys/****@orcl "start observer" -logfile /u01/app/oracle/observer.log &
```

Com o ambiente definido podemos habilitar o Fast-Start Failover usando o comando *enable fast_start failover*. A Figura 22 apresenta

este comando sendo executado e o status da configuração após habilitado.

Podemos verificar o status do fast-start failover com o comando *show fast_start failover*. Este apresenta as situações opcionais em que o failover ocorrerá e os thresholds configurados. No caso da Figura 23, o failover será acionado automaticamente caso o dicionário de dados ou control files estejam corrompidos ou ainda se um datafile ficar off-line por motivo de falha no arquivo.

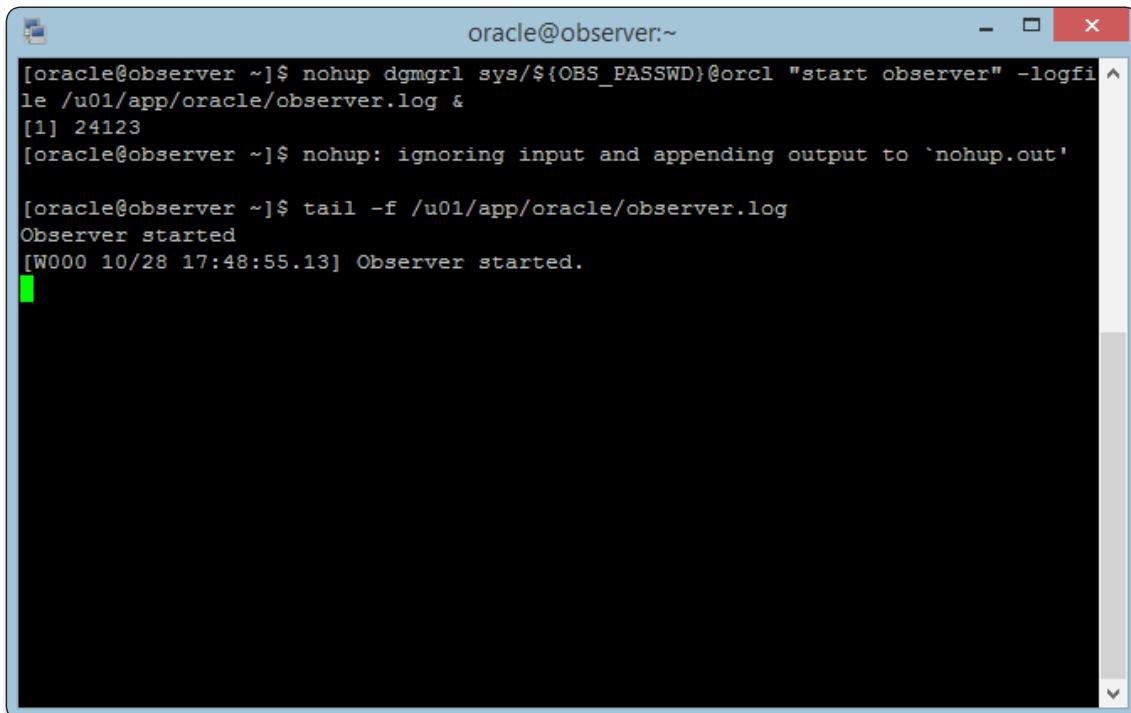


```
[oracle@observer ~]$ dmgmgr
DGMGRL for Linux: Version 12.1.0.2.0 - 64bit Production

Copyright (c) 2000, 2013, Oracle. All rights reserved.

Welcome to DGMGRL, type "help" for information.
DGMGRL> connect sys@orcl
Password:
Connected as SYSDBA.
DGMGRL> start observer
Observer started
```

Figura 20. Iniciando o observer



```
[oracle@observer ~]$ nohup dmgmgr sys/${OBS_PASSWD}@orcl "start observer" -logfile /u01/app/oracle/observer.log &
[1] 24123
[oracle@observer ~]$ nohup: ignoring input and appending output to `nohup.out'

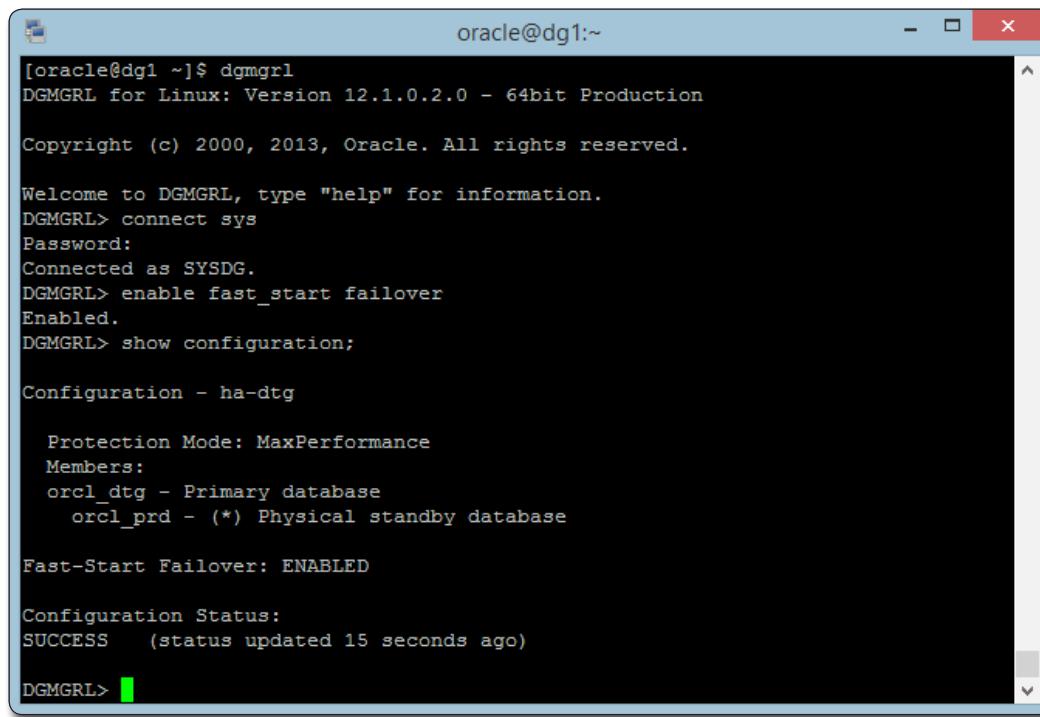
[oracle@observer ~]$ tail -f /u01/app/oracle/observer.log
Observer started
[W000 10/28 17:48:55.13] Observer started.
```

Figura 21. Iniciando observer com log

Passo 9: Teste de Fast-Start Failover

Para testar uma falha na base de produção, vamos matar o processo *pmon* da base primária usando o comando *kill*. Feito isso, a base primária se encerrará com erro e a base secundária será iniciada automaticamente pelo observer.

Na Tabela 1 é apresentada a sequência de acontecimentos esperados em cada um dos componentes da infraestrutura. Ao final teremos a base primária na máquina DG2 e a máquina DG1 estará com a base de dados em modo standby. O Data Guard é uma alternativa bastante útil e segura de incrementar a alta



```
[oracle@dg1 ~]$ dgmgrl
DGMGRL for Linux: Version 12.1.0.2.0 - 64bit Production

Copyright (c) 2000, 2013, Oracle. All rights reserved.

Welcome to DGMGRL, type "help" for information.
DGMGRL> connect sys
Password:
Connected as SYSDG.
DGMGRL> enable fast_start failover
Enabled.
DGMGRL> show configuration;

Configuration - ha-dtg

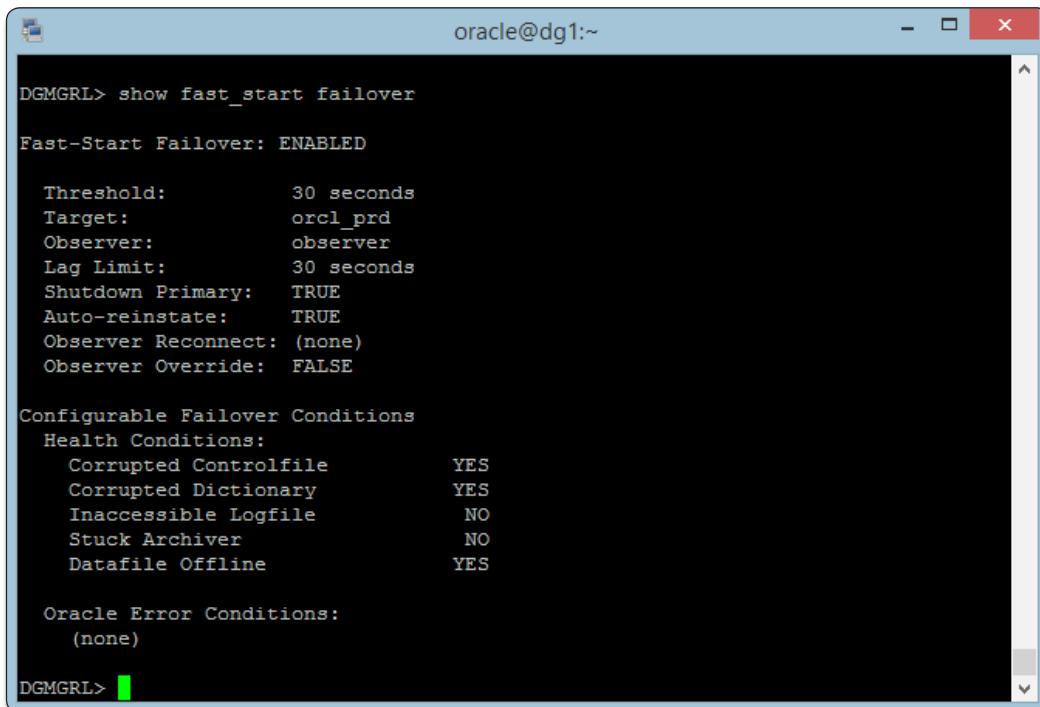
  Protection Mode: MaxPerformance
  Members:
    orcl_dtg - Primary database
    orcl_prd - (*) Physical standby database

Fast-Start Failover: ENABLED

Configuration Status:
SUCCESS      (status updated 15 seconds ago)

DGMGRL>
```

Figura 22. Habilitando o Fast-Start Failover



```
DGMGRL> show fast_start failover

Fast-Start Failover: ENABLED

  Threshold:          30 seconds
  Target:            orcl_prd
  Observer:          observer
  Lag Limit:         30 seconds
  Shutdown Primary: TRUE
  Auto-reinstate:   TRUE
  Observer Reconnect: (none)
  Observer Override: FALSE

Configurable Failover Conditions
  Health Conditions:
    Corrupted Controlfile      YES
    Corrupted Dictionary       YES
    Inaccessible Logfile       NO
    Stuck Archiver             NO
    Datafile Offline           YES

  Oracle Error Conditions:
    (none)

DGMGRL>
```

Figura 23. Verificando parâmetros FSFO

Oracle Data Guard com Fast-Start Failover

DG1	DG2	Observer
Processo PMON encerrado com o comando kill. O banco de dados primário é encerrado.		
	Banco de dados ORCL_DTG é iniciado como primário.	Log do observer apresenta a mensagem: "Initiating Fast-Start Failover to database "orcl_dtg"... Performing failover NOW, please wait... Failover succeeded, new primary is "orcl_dtg"."
Manualmente, montamos o database ORCL_PRD (após verificar eventual problema, se for o caso).		
Inicia-se o procedimento de recuperação da base ORCL_PRD usando o Flashback Database.		Log do observer apresenta a mensagem: "Initiating reinstatement for database "orcl_prd"... Reinstating database "orcl_prd", please wait..."
Banco de dados ORCL_PRD é iniciado como standby.		Log do observer apresenta a mensagem: "Reinstatement of database "orcl_prd" succeeded".

Tabela 1. Sequência de acontecimentos em uma simulação de falha em que o observer realiza o failover automaticamente

disponibilidade de bancos de dados Oracle, e o Fast-Start Failover facilita o trabalho de Failover em caso de problemas com o ambiente primário de forma bastante segura. Esta funcionalidade faz parte do pacote oferecido pelo Oracle Enterprise Edition, bastando que os processadores do servidor standby também sejam licenciados como Enterprise Edition.

Como complemento ainda pode ser habilitada a option *Active Data Guard*, que permite manter aberto em read only o banco de dados secundário (standby), enquanto o mesmo recebe transações do banco primário. Esta option paga é especialmente interessante em bancos de dados que possuem aplicações que realizam somente leitura, como geração de relatórios ou análises de BI ou cargas (externas) de bancos DW.

Links:

Guide to Oracle Data Guard Fast-Start Failover

<http://www.oracle.com/technetwork/articles/smiley-fsfo-084973.html>

Client Failover Best Practices for Highly Available Oracle Databases: Oracle Database 11g Release 2

<http://www.oracle.com/au/products/database/maa-wp-11gr2-client-failover-173305.pdf>

Database High Availability Best Practices

https://docs.oracle.com/database/121/HABPT/config_dg.htm#HABPT4876

Autor



Ivan Ricardo Schuster

ivanrs79@gmail.com



Com mais de 15 anos de carreira como DBA, já trabalhou em diversos projetos de Bancos de Dados de alta criticidade, apresentando soluções para as mais diversas falhas de infraestrutura, em ambientes de alta disponibilidade e alta complexidade. Possui diversas certificações Oracle, tais como OCP 10g, 11g e 12c, OCE RAC e Exadata, trabalha atualmente na Teiko Tecnologia da Informação Ltda. como consultor DBA.

Você gostou deste artigo?

Dê seu voto em www.devmedia.com.br/sqlmagazine/feedback

Ajude-nos a manter a qualidade da revista!



Programador Java: Por onde começar?

Descubra nesse vídeo como entrar na carreira Java com o pé direito!



DEVMEDIA

<http://www.devmedia.com.br/programador-java-por-onde-comecar/33638>

Somos tão apaixonados por tecnologia que o nome da empresa diz tudo.

Porta 80 é o melhor que a Internet pode oferecer para sua empresa.

Já completamos 8 anos e estamos a caminho dos 80, junto com nossos clientes.

Adoramos tecnologia. Somos uma equipe composta de gente que entende e gosta do que faz, assim como você.



Estrutura

100% NACIONAL.
Servidores de primeira linha, links de alta capacidade.

Supporte diferenciado

Treinamos nossa equipe para fazer mais e melhor. Muito além do esperado.

Serviços

Oferecemos a tecnologia mais moderna, serviços diferenciados e antenados com as suas necessidades.

1-to-1

Conhecemos nossos clientes. Atendemos cada necessidade de forma única.
[Conheça!](#)



Porta 80
WEB HOSTING

Hospedagem | Cloud Computing | Dedicados | VoIP | Ecommerce |
Aplicações | Streaming | Email corporativo

porta80.com.br | comercial@porta80.com.br | twitter.com/porta80

SP 4063-8616 | RJ 4063-5092 | MG 4063-8120 | DF 4063-7486