Universidade do Minho
Escola de Engenharia

Agente de Monitorização de Bases de Dados Oracle

Administração e Exploração de Bases de Dados
Mestrado Engenharia Informática

Realizado por:
José Oliveira PG33873
Tiago Araujo A71346
Manuel Maciel A68410

Índice

1 Introdução

Nos últimos anos tem vindo a crescer a necessidade de aquisição de Database Activity Monitors (DAM) derivado do facto de que qualquer base de dados está propicia a atividades maliciosas e que podem, muitas vezes, encontrar-se camufladas entre tantos outros processos que estejam a decorrer no momento (MogulI).

A vantagem de um DAM é que este é independente do Database Management System (DBMS) o que significa que não irá interferir com o comportamento do mesmo.

Existe também uma extensão para DAM que se denomina Database Activity Monitoring and Prevention (DAMP) que para além de monitorizar a actividade de uma base de dados pode também alertar ou até mesmo prevenir certas ameaças ao sistema. Na secção 3 falaremos um pouco mais de DAM's.

Há duas maneiras mais comuns de monitorizar uma base de dados, pode ver a informação num determinado espaço de tempo seja este real ou aproximado ou podem ser criados "triggers" que capturam informação de acordo com certos eventos a decorrer (ver (IBM Corporation, 2013)).

A ideia proposta neste projeto é implementar um DAM que permita monitorização de forma simples e intuitiva dos principais parâmetros de avaliação de performance de uma base de dados.

A extração da informação foi feita com algumas queries de Structured Query Language (SQL) executadas num programa escrito em python, o ambiente gráfico foi montado de forma conveniente e eficiente com HTML5 para que o administrador não tivesse qualquer problema na sua utilização.

Este documento está organizado da forma seguinte. O capítulo 3 fornece alguns conceitos relacionados com o trabalho. No capítulo 4 explica-se todo o processo de desenvolvimento do agente de monitorização. E no capítulo 5 são estabelecidas algumas conclusões.

2. Trabalho Relacionado

Como visto em (Armerding, 2017) várias empresas foram vítimas de ciberataques como SQL injection, algumas destas empresas como eBay e Yahoo viram comprometidas milhões de contas de utilizadores que continham nomes verdadeiros, endereços de email, datas de nascimento e números de telefone.

A empresa britânica Equifax (guardian, 2017) admitiu que foram roubadas informações pessoais a cerca de 700.000 consumidores britânicos incluindo detalhes parciais de cartões de crédito e outros. A IBM (IBM Corporation, 2013, pág.31) possui uma ferramenta de interface web que ajuda a monitorizar a base de dados, identificando problemas, a sua causa, evitando problemas de desempenho e resolvendo os mesmos. Este tipo de monitorização facilita também a deteção de muitas falhas de segurança o que poderá facilitar o processo de mitigação das mesmas.

3. Alguns conceitos

3.1. O que é um Database Activity Monitor DAM ?

Um Database Activity Monitor é, tal como o nome sugere, um monitor de base de dados que visa reunir informação e guardar, seja esta em tempo real ou próximo, das atividades SQL a decorrer numa dada base de dados podendo, se programado para tal, emitir alertas.
O comportamento de um DAM é independente da base de dados onde recolhe informação, o que significa que não influencia as restantes atividades diretamente.
Segundo (Mogull) os DAMs têm 5 funcionalidades principais:

1. Habilidade para monitorizar base de dados de forma independente
2. Guardar a informação recolhida fora da base de dados
3. Consegue agregar e correlacionar atividades de múltiplos DBMSs
4. Fortalecer separação de deveres dos administradores de base de dados (DBA)
    a. Uma auditoria deve incluir atividade do DBA
5. Emitir alertas no caso de uma violação de política estabelecida.

No entanto, o nosso DAM não vai cobrir as funcionalidades todas, ao que podemos então propor uma nova lista de funcionalidades:

1. Habilidade para monitorizar base de dados de forma independente
2. Guardar a informação recolhida fora da base de dados
3. Apresentar informação ao utilizador por via de aplicação Cliente-Servidor com interface gráfica html5.

3.2. Em que consiste computação Cliente-Servidor ?

Segundo (Lewandowski, 1998), sistemas de computação Cliente-Servidor têm duas partes lógicas sendo estas:

1. Servidor que fornece um determinado serviço
2. Cliente que pede serviços ao servidor

Estes dois pontos quando juntos formam um sistema de computação Cliente-Servidor que relaciona duas ou mais Threads de execução a com uma relação consumidor-fornecedor. O cliente será o consumidor, que fará certos pedidos ao seu fornecedor (Servidor) para recolher informação ou qualquer outro tipo de serviço e que, quando fornecida/o, poderá continuar o seu trabalho.

Um bom exemplo de um sistema Cliente-Servidor é a rede de Automated Teller Machines (ATM) em que um utilizador usa a ATM como cliente que serve de interface a um servidor secundário e que comunica diretamente com um servidor central que, por sua vez, administra todos os outros servidores secundários.

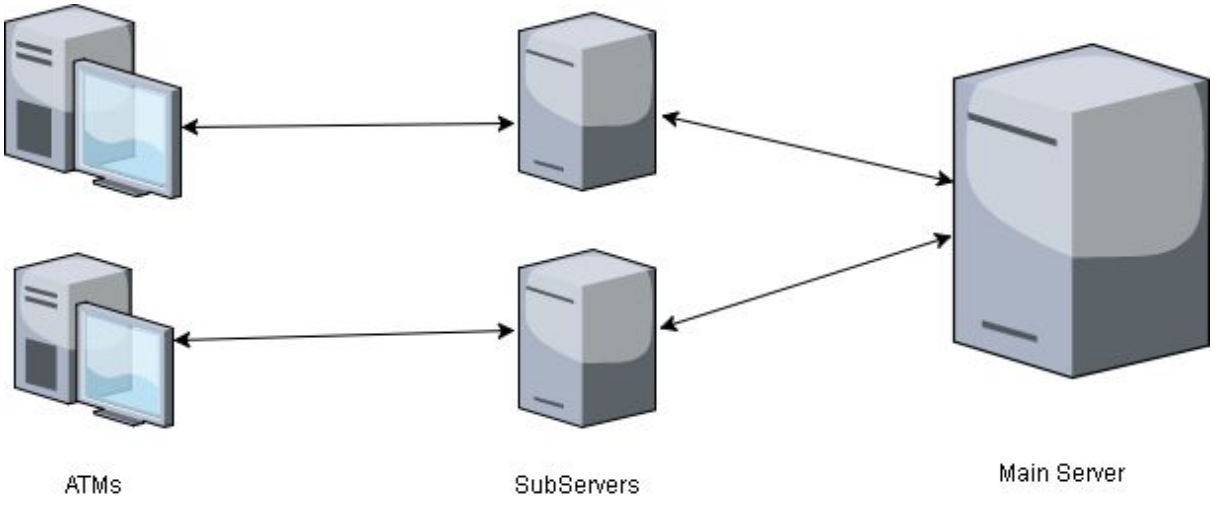


Fig1. Arquitetura Cliente-Servidor

3.2.1 O cliente

O cliente, num sistema de Cliente-Servidor, tem geralmente uma interface gráfica (GUI) que apresenta os serviços disponibilizados pelo servidor como objetos independentes. Um cliente pode também oferecer serviços a outros clientes bem como estar conectado a múltiplos servidores uma vez que não existem limites

3.2.2 O servidor

O servidor tradicional (Lewandowski, 1998) fica em *standby* à espera de pedidos de clientes mas os sistemas mais compactos buscam informação proveniente do cliente e operam conforme a mesma.

O resultado do pedido é encapsulado do lado do servidor, de forma a isolar o mesmo de todas as restantes atividades do sistema, e depois será entregue por uma via escondida ao cliente.

Geralmente, a forma de como um servidor tem de fornecer serviços está inteiramente dependente da arquitetura do sistema, por exemplo, o servidor pode dividir uma tarefa em subtarefas e atribuir as mesmas aos seus sub-servidores.

3.2.3 Middleware

O software que facilita a comunicação entre cliente e servidor denomina-se *Middleware*. É um pedaço de software que se localiza entre o servidor e a aplicação. Essencialmente, é uma ferramenta que facilita a vida de desenvolvedores de sistemas distribuídos (Nunn). Grande vantagem de um *middleware* é poder usar um único ambiente de desenvolvimento tendo em conta que, num sistema mais complexo, podemos ter Sistemas Operativos (OS) variados a usar diferentes linguagens de programação.

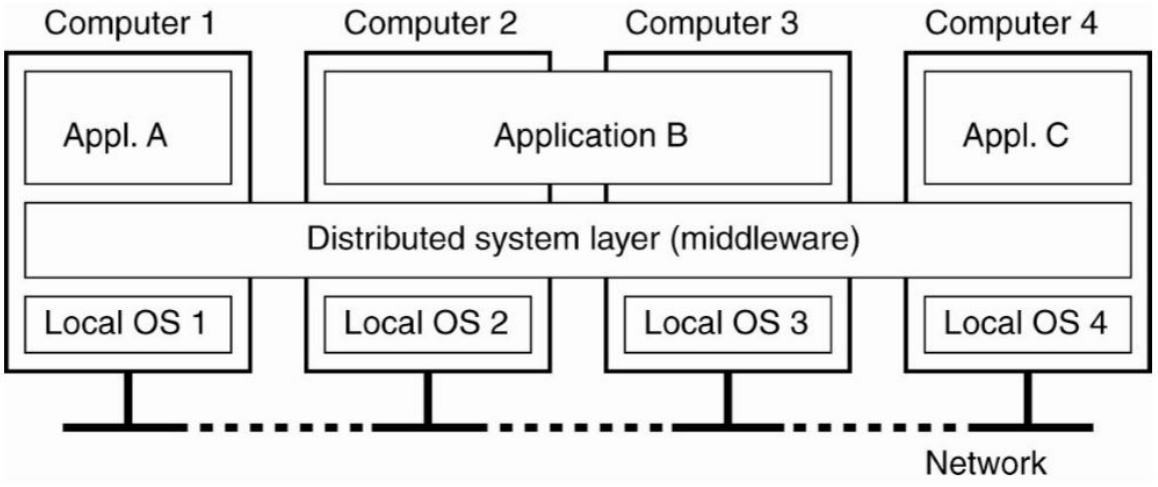


Fig2. Sistema distribuido atribuído por *middleware*.
A camada de *middleware* estende-se pelas várias máquinas e oferece a cada aplicação a mesma interface

Há muitas falhas que poderão ser tratadas automaticamente sem ajuda de um programador ou aplicação e, dependendo do tipo de middleware, poderá oferecer Transparência de Acesso o que significa que oculta diferenças na representação dos dados e no modo como os recursos poderão ser acessados pelo utilizador.

Existem 5 tipos de *middleware* descritos em (Inspirel, 2015):

1. **Object-Oriented Middleware**
   Segundo (Nunn), estes sistemas são uma extensão de *procedural middleware* á qual foram adicionadas características das novas linguagens orientadas a objetos (OOP - Object-Oriented Programming)

2. **Service-Oriented Middleware (SOM)**
   Baseado em (Al-Jaroodi et al, 2010), as funcionalidades de um SOM podem ser divididas em 8 categorias:
   1. Suporte de tempo de execução (runtime) para fornecedores de serviços
   2. Suporte para que consumidores possam descobrir e usar serviços registados
   3. Abstrações para ocultar a heterogeneidade de ambientes subjacentes através de linguagens e protocolos de suporte
   4. Serviços configuráveis
   5. Transparência de serviços para aplicações cliente

6. Descobrir e modificar mecanismos de fornecedores de serviços
7. Interoperabilidade com uma variedade de dispositivos e sistemas
8. Eficiênte manipulação de volumes de dados e cargas de comunicação massivos.

3. **Data-centric middleware**
   Como o nome sugere, *data-centric middleware* foca-se na partilha de dados e informações sobre o estado do sistema (Shacker, 2010). O *middleware* está ciente dos dados a serem partilhados entre aplicações e constrói mensagens para comunicar os updates e mudanças do estado do sistema.
   Estas mensagens não são genéricas como no caso de *message-centricity*, em vez disso, são derivadas do modelo de sistema de dados.

4. **Message-oriented middleware (MOM)**
   Facilita comunicação através de uma troca de mensagens. Estas mensagens são pequenos blocos de informação que podem ser usados para fornecer notificações de evento e pedidos de execução de serviços. Este tipo de *middleware* está bem equipado para notificações de evento distribuídas.

Excepcionalmente, em (Nunn) temos também um tipo de *middleware* denominado *Procedural Middleware* que é geralmente usado para providênciar Chamadas de Procedimento Remoto (RPC) que servem essencialmente para definir componentes de servidor com uma Linguagem de Descrição de Interface (IDL) que, de seguida, executa o processo de comunicação de rede. RPC tem ligações a vários OSs e linguagens de programação tornando-se uma solução simples para sistemas distribuídos multiplataforma.

4. Desenvolvimento

Este projeto foi realizado com Sistema Operativo Windows 10 versão 10.0.15063 e para tal foram usadas algumas ferramentas de trabalho para ligação a uma base de dados Oracle e extração de informação.

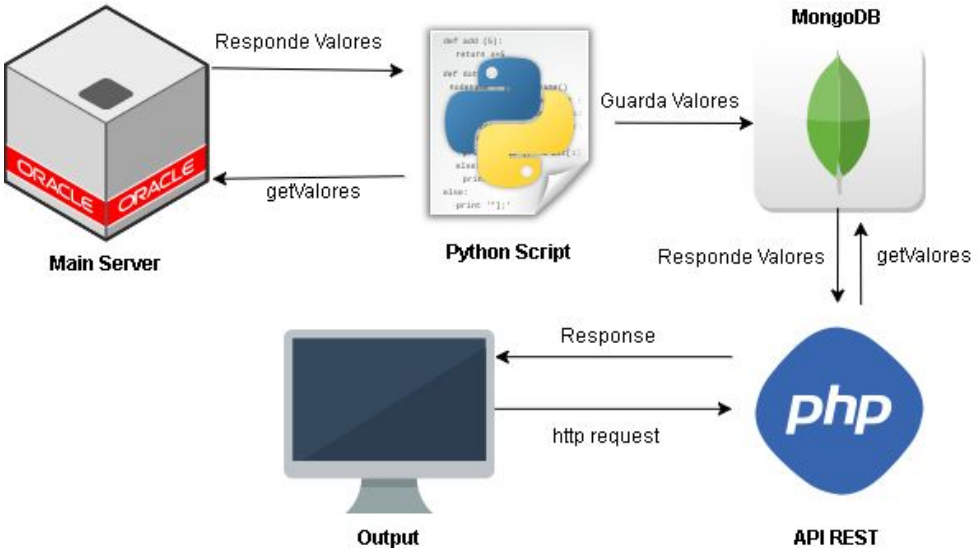


Fig3. Fluxograma da solução.

4.1. Extração de informação da BD Oracle

A linguagem de programação usada na extração da informação foi python na versão 3.6.2 com o pacote cx_Oracle na versão 6.0.3 (note que a 6.0b2[beta] não funciona) e o programa usado para criação do script foi o spyder do Anaconda versões 3.2.3 e 1.6.8 respetivamente.
As queries usadas extraíram informações relacionadas com o espaço usado,com os users, com as sessões, com os wait e com as operações IO. Foram também retiradas algumas estatísticas.
Depois de obter as informações necessárias, o script irá proceder ao envio das mesmas para a nova base de dados local não relacional feita em MongoDB.

4.2. Inserção de dados na MongoDB

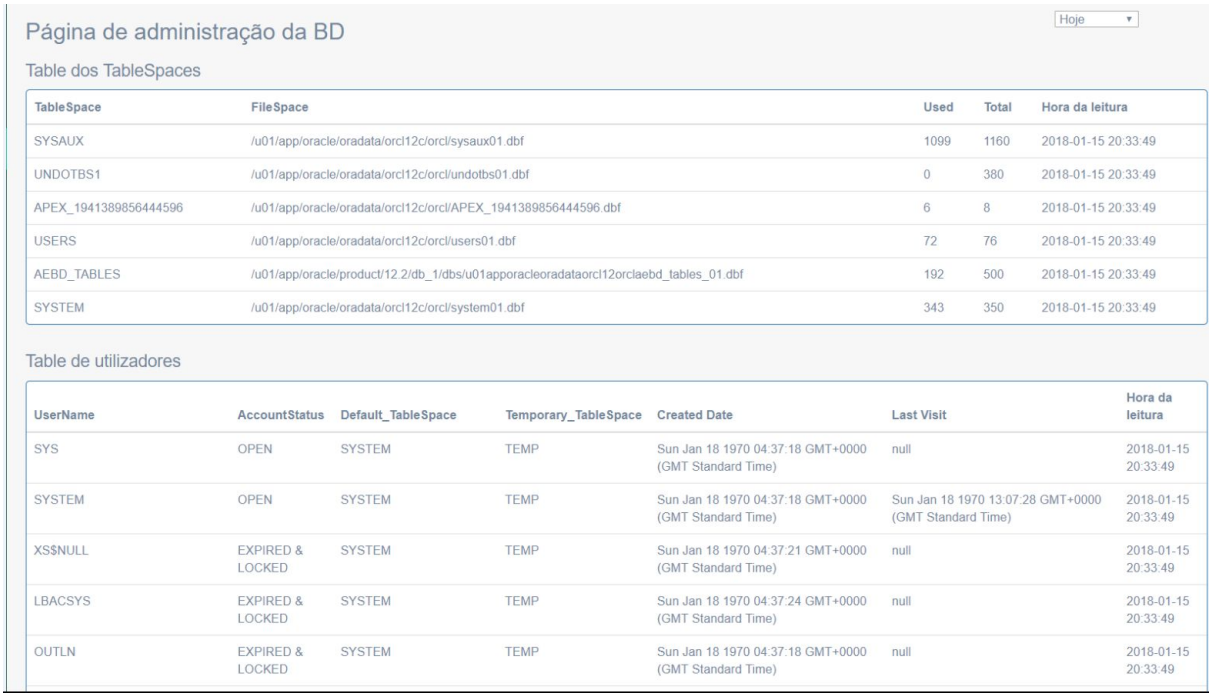
Para armazenar os dados extraídos da base de dados oracle, usamos uma base de dados não relacional, mais precisamente o MongoDB. Tomamos esta decisão pois as tabelas da nossa base dados não iram ter qualquer relação entre elas, é um modelo que não obriga a base de dados a ter um esquema definido, a qualquer momento o administrador pode

decidir guardar os dados numa estrutura diferente sem ter de alterar os dados anteriormente armazenados, e por último é um sistema mais simples que uma base de dados Oracle.
Os dados extraídos foram inseridos na base de dados não relacional em MongoDB através do uso do pacote pyMongo, versão 3.6.0. Foi criada a base de dados "*admin_database*" que engloba as seguintes colecções onde os dados foram inseridos:

- *space_collection*: para cada tablespace indica o espaço usado em MB, o espaço livre em MB,o espaço total em MB, a percentagem livre e a data em que a query foi executada.
- *users_collection*: para cada utilizador indica o *username*, o estado da conta, o tablespace *default* e temporário, a data em que o utilizador foi criado, a última vez que fez *login* e a data em que a query foi executada.
- *session_collection*: para cada sessão indica o SID, o Serial, o uso de cpu em segundos e a data em que a query foi executada.
- *stat_collection*: apresenta uma lista de estatísticas cada um com o seu nome,id e valor tal como a data em que a query foi executada.
- *wait_collection*: apresenta uma lista de eventos e o *wait time* que lhes está associado e a data em que a query foi executada.
- *IO_collection*: para cada datafile indica o número de *physical reads*, a percentagem de *reads*, o número de *physical writes*, a percentagem de *writes*, o total de *block IOs* e a data em que a query foi executada

4.3. Criação do website

O website foi criado utilizado um back office escrito em javascript, utilizador um bootstrap para auxiliar a construção do mesmo e o desenvolvimento dos gráficos. Ao carregar uma página HTML do nosso website é chamada uma função do back office que por cada gráfico presente na página envia um pedido HTTP através de Ajax à base de dados não relacional, como resposta o back office recebe em json os dados necessário para a construção do gráfico em questão, é utilizado uma das livrarias do bootstrap para gerar e apresentar o gráfico ao utilizador. No website temos quatro páginas distintas, a *BdManagement* que tem duas tabelas onde apresenta as bases de dados no sistema e sua respectiva informação, e outra onde apresenta os users que foram criados no sistema e sua respectiva informação.



Página de administração da BD

Hoje

Table dos TableSpaces

TableSpace	FileSpace	Used	Total	Hora da leitura
SYSAUX	/u01/app/oracle/oradata/orcl12c/orcl/sysaux01.dbf	1099	1160	2018-01-15 20:33:49
UNDOTBS1	/u01/app/oracle/oradata/orcl12c/orcl/undotbs01.dbf	0	380	2018-01-15 20:33:49
APEX_1941389856444596	/u01/app/oracle/oradata/orcl12c/orcl/APEX_1941389856444596.dbf	6	8	2018-01-15 20:33:49
USERS	/u01/app/oracle/oradata/orcl12c/orcl/users01.dbf	72	76	2018-01-15 20:33:49
AEBD_TABLES	/u01/app/oracle/product/12.2/db_1/dbs/u01apporacleoradataorcl12orclaebd_tables_01.dbf	192	500	2018-01-15 20:33:49
SYSTEM	/u01/app/oracle/oradata/orcl12c/orcl/system01.dbf	343	350	2018-01-15 20:33:49

Table de utilizadores

UserName	AccountStatus	Default_TableSpace	Temporary_TableSpace	Created Date	Last Visit	Hora da leitura
SYS	OPEN	SYSTEM	TEMP	Sun Jan 18 1970 04:37:18 GMT+0000 (GMT Standard Time)	null	2018-01-15 20:33:49
SYSTEM	OPEN	SYSTEM	TEMP	Sun Jan 18 1970 04:37:18 GMT+0000 (GMT Standard Time)	Sun Jan 18 1970 13:07:28 GMT+0000 (GMT Standard Time)	2018-01-15 20:33:49
XS$NULL	EXPIRED & LOCKED	SYSTEM	TEMP	Sun Jan 18 1970 04:37:21 GMT+0000 (GMT Standard Time)	null	2018-01-15 20:33:49
LBACSYS	EXPIRED & LOCKED	SYSTEM	TEMP	Sun Jan 18 1970 04:37:24 GMT+0000 (GMT Standard Time)	null	2018-01-15 20:33:49
OUTLN	EXPIRED & LOCKED	SYSTEM	TEMP	Sun Jan 18 1970 04:37:18 GMT+0000 (GMT Standard Time)	null	2018-01-15 20:33:49

Fig4. Tabelas na página de administração da BD

Depois temos a página IO_SESSIONS que mostra quatro gráficos, sendo o primeiro responsável por mostrar os datafiles do sistema e o seu IO até ao momento escolhido, outro que é responsável por mostrar os o tempo que cada utilizador gastou no CPU até ao momento, depois existe um gráfico que mostra as três maiores razões de esperas no sistema e compara com as restantes.

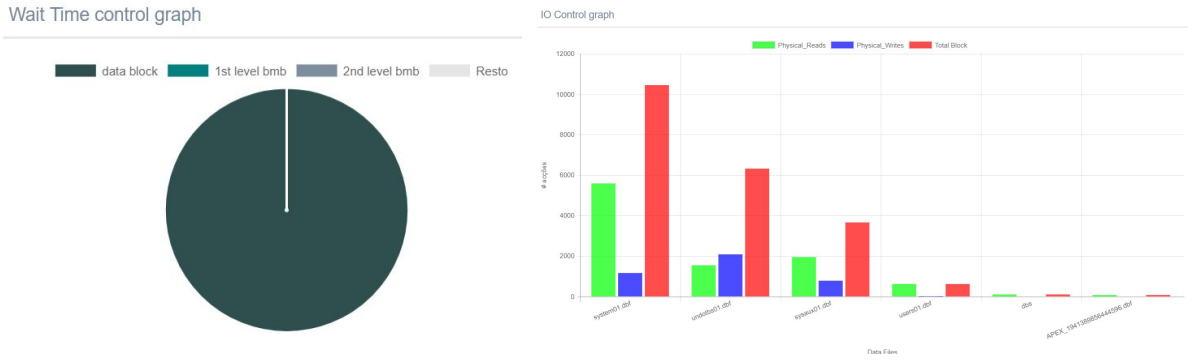


Fig5. Gráfico de wait time e gráfico de IO

Para além das páginas mencionadas em cima existe também uma que mostra a utilização do sistema nos últimos 10 minutos.

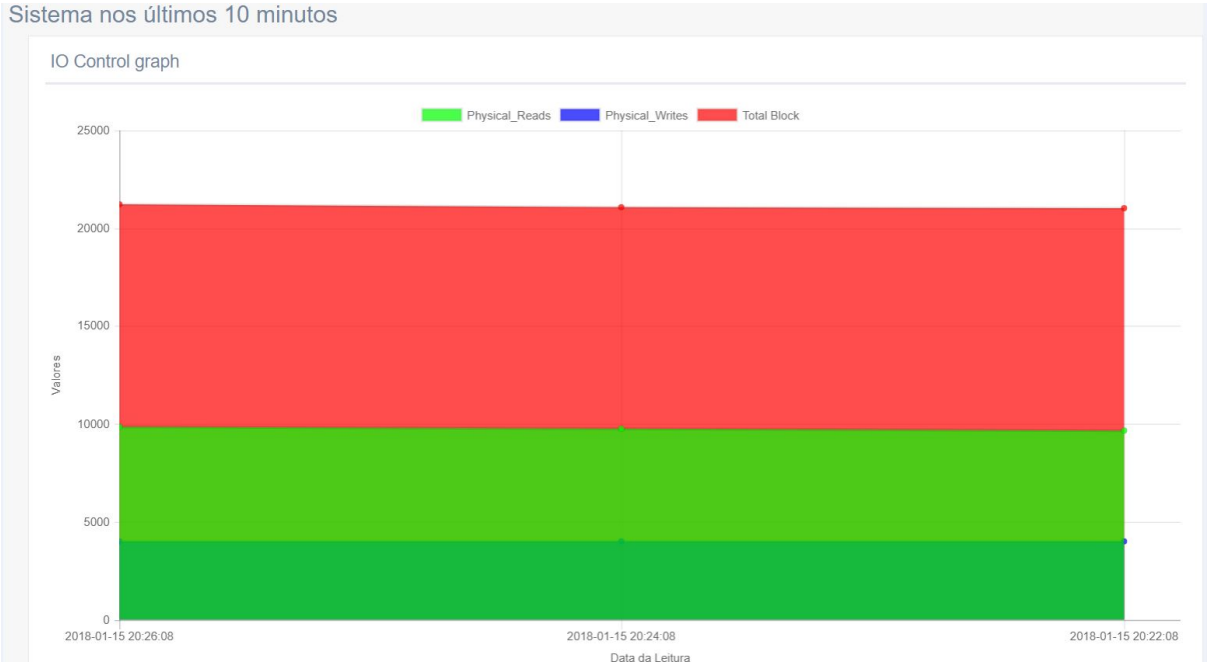


Fig6. Gráfico da evolução das escritas/leituras no últimos 10 minutos

No exemplo acima dá um gráfico linear porque os valores não se alteraram ao longo das leituras.Por último temos uma página que mostra as escritas e leituras de um determinado datafile num determinado momento.

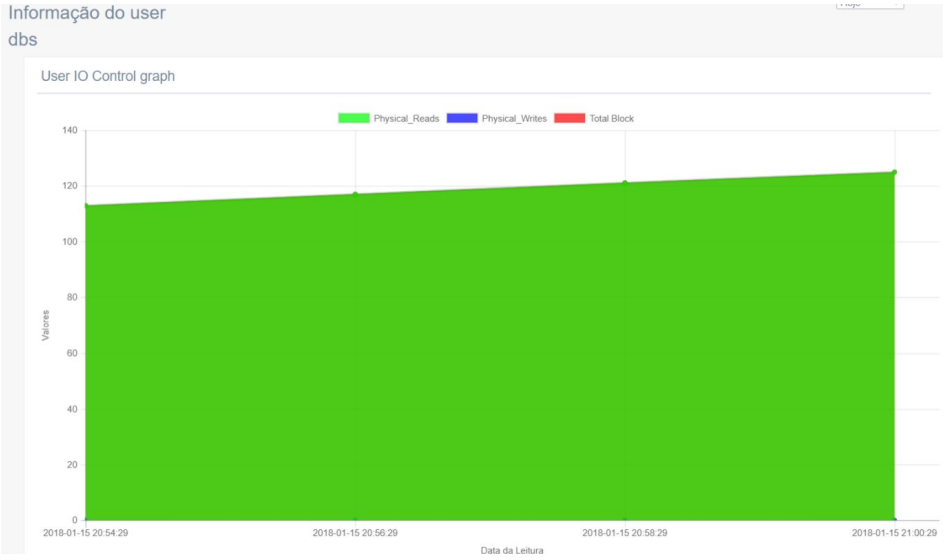


Fig7. Gráfico das escritas/leituras de um utilizador(dbs)

5. Conclusões e trabalho futuro

O agente de monitorização de Bases de Dados desenvolvido encontra-se a funcionar com todos os requisitos propostos, contudo futuramente poderiam ser incluídas novas funcionalidades para complementar o sistema e acomodar melhor as preferências do utilizador.
Em suma, achamos que o trabalho e a experiência foram positivas.
O script responsável pela extração e inserção da informação não se encontra a correr em tempo real mas sim num curto espaço de tempo, de 2 em 2 minutos, evitando assim o custo desnecessário de estar constantemente a fazer acessos à base de dados Oracle.
A GUI do sistema foi projetada convenientemente para ser bastante acessível até para utilizadores mais inexperientes

Referências

Anaconda 3.5 (64bit Windows) Install cx_Oracle 2016
https://stackoverflow.com/questions/34637836/anaconda-3-5-64bit-windows-install-cx-oracle

ODPI-C Instalation 2016
https://oracle.github.io/odpi/doc/installation.html

OWASP TOP 10
Accessed in 13/10/2017
https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf

Periorellis, Panos, Securing Web Services: Practical Usage of Standards and Specifications
IGI Global, 31/10/2007
Accessed in 21/11/2017
https://books.google.pt/books?id=zX2N7fWTJOUC&pg=PA319&lpg=PA319&dq=web+application+bad+practices+consequences&source=bl&ots=iQScE7EaUe&sig=o1L5u7SwABf2W5_yV7iexfHg1RQ&hl=pt-PT&sa=X&ved=0ahUKEwiD5dnO9M_XAhVIuxQKHZQJA10Q6AEIUTAF#v=onepage&q=web%20application%20bad%20practices%20consequences&f=false

Armerding, Taylor. The 16 biggest data breaches of the 21st century
Accessed in 21/11/2017
https://www.csoonline.com/article/2130877/data-breach/the-16-biggest-data-breaches-of-the-21st-century.html

EQUIFAX
https://www.theguardian.com/technology/2017/oct/11/personal-details-of-almost-700000-britons-hacked-in-cyber-attack

Avi Silberschatz, Henry F. Korth and S. Sudarshan, Database System Concepts, Mc Grow Hill, 2010

IBM Coporation, Database Monitoring Guide and Reference, 2013
ftp://ftp.software.ibm.com/ps/products/db2/info/vr105/pdf/en_US/DB2Monitoring-db2f0e1050.pdf

Tore Risch, Monitoring Database Objects , 1989
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.100.1209&rep=rep1&type=pdf

Rich Mogull, Understanding and Selecting a Database Activity Monitoring Solution
https://securosis.com/assets/library/reports/DAM-Whitepaper-final.pdf

Peter Lin, So you want high performance
https://tomcat.apache.org/articles/performance.pdf

Lisbeth Bergholt, Jacob Steen Due, Thomas Hohn, Jørgen Lindskov Knudsen,Kirsten Hjerrild Nielsen, Thomas Sonne Olesen, Emil Hahn Pedersen
Database Management Systems: Relational, Object-Relational, and Object-Oriented Data Models
http://www.cit.dk/cot/reports/reports/Case4/05-v1.1/cot-4-05-1.1.pdf

Global Network of Hackers Steal $45 Million From ATMs
Published 7:23 AM ET Fri, 10 May 2013
Accessed 15-01-2018
https://www.cnbc.com/id/100726799

SCOTT M. LEWANDOWSKI 1-March-1998
Frameworks for Component-Based Client/Server Computing
http://homepages.gsd.inesc-id.pt/~ler/docencia/tm0405/papers/OrbLewandowski.pdf

Inspirel, Types of middleware, 2015
http://www.inspirel.com/articles/Types_Of_Middleware.html

Robert Nunn, Distributed Software Architectures Using Middleware
http://www0.cs.ucl.ac.uk/staff/ucacwxe/lectures/3C05-02-03/aswe18-essay.pdf

Jameela Al-Jaroodi, Nader Mohamed, and Junaid Aziz, 2010
Service Oriented Middleware: Trends and Challenges
http://middleware-tech.net/papers/ITNG2010_SOM.pdf

David T. Liu, Michael J. Franklin , Jim Garlick, Ghaleb M. Abdulla (2005)
Scaling Up Data-Centric Middleware on a Cluster Computer
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.507.8073&rep=rep1&type=pdf

Curt Schacker, 2010
Data-Centric Middleware A fundamental improvement in developing, maintaining and deploying mission-critical distributed systems Curt Schacker, VP Worldwide Field Operations
https://d2vkrkwbbxbylk.cloudfront.net/sites/default/files/archive/RTI_Data_Centric_Middleware.pdf

Mubina Malik1 and Trisha Patel2, 2016
DATABASE SECURITY - ATTACKS AND CONTROL METHODS
http://aircconline.com/ijist/V6N2/6216ijist18.pdf

Template bootStrap utilizado
https://github.com/puikinsh/gentelella