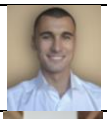
Sistemas de Informação Distribuídos

Licenciaturas em Engenharia Informática e Informática e Gestão de Empresas
2017-2018, Segundo Semestre

Monitorização de Culturas em Laboratório

Auditoria e Migração

Identificação do grupo autor da especificação (Etapa A): __G11__

Número	Nome	Foto
74232	Alberto Ramos	
72948	André Carvalho	
73105	Bruno Carreira	
74486	Daniel Reis	
Especificação: ☐ ODBC ☒ Ficheiro		

Identificação do grupo autor da implementação (Etapas B e C): __G23__

Número	Nome	Foto
62109	André Vieira	
16482	Paulo Vieira	
69565	Rodolfo Arnaldo	
73553	Rui Tomé	
65345	Tiago Rodrigues	
Especificação: ☒ ODBC ☐ Ficheiro Implementação: ☐ ODBC ☒ Ficheiro		

Instruções

Estas instruções são de cumprimento obrigatório. Relatórios que não cumpram as indicações serão penalizados na nota final.

- Podem (e em várias situações será necessário) ser adicionadas novas páginas ao relatório, mas não podem ser removidas páginas. Se uma secção não for relevante, fica em branco, não pode ser removida;
- Todas as secções têm que iniciar-se no topo de página (colocar uma quebra de página antes);
- A paginação tem de ser sequencial e não ter falhas;
- O índice tem de estar actualizado;
- Na folha de rosto (anterior) têm de constar toda a informação solicitada, nomeadamente todas as fotografias de todos os elementos dos dois grupos. É obrigatório que caiba tudo numa única página;
- A formatação das "zonas" (umas sombreadas outras não sombreadas) não pode ser alterada;
- Nas etapas A e B (até secção 1.4 inclusive), o grupo que primeiro edita o documento (Etapa A) **apenas escreve nas zonas não sombreadas**, e o outro grupo apenas escreve nas zonas sombreadas;
- A etapa C é apenas preenchida pelo grupo que recebe o presente documento do outro grupo. Nas secções 2.1, 2.2, 2.3 e 2.6 deve colocar nas zonas não sombreadas a especificação que entregou ao outro grupo (sem alteração, *copy e paste*),
- As restantes secções são preenchidas normalmente pelo grupo que recebe o presente documento do outro grupo.

Índice

Monitorização de Culturas em Laboratório

Um laboratório de investigação de um departamento biológico necessita de um sistema para monitorizar a evolução de culturas. Nomeadamente pretende acompanhar a temperatura e humidade a que as culturas estão sujeitas, bem como detectar/antecipar potenciais problemas.

Cada cultura tem um único investigador responsável e apenas ele pode actualizar e consultar os dados de medições das suas culturas. Esta *protecção de dados* é um aspecto importante do sistema.

Sobre cada cultura são regularmente efectuadas (manualmente) medições com base num conjunto de variáveis que variam consoante a cultura. Para cada cultura o sistema conhece o intervalo de valores normal para cada variável, logo, o sistema poderá emitir alertas caso surja um valor anormal.

Por exemplo, para as culturas hidropónicas de pimento e tomate, fazem-se medições do nível de concentração de mercúrio e chumbo. Se, por exemplo, a concentração de chumbo no pimento reduzir significativamente – menos de 25 mg/litro – significa que a planta ajuda a absorver os metais indesejáveis. (*Culturas = pimento e tomate (hidropónico), variáveis = mercúrio, chumbo.)*

Outro exemplo. Numa solução onde convivem bactérias e antibióticos, se o número de bactérias cresce pouco então é porque são sensíveis ao antibiótico (logo, sabemos como as matar se forem prejudiciais). Se o número de colónias de bactérias *Bacillus subtilis,* colocadas junto de antibiótico penicilina, aumentar em mais de 30% em 2 horas é porque o antibiótico não é eficaz. *(Cultura = Bacillus subtilis, variável = penicilina.)*

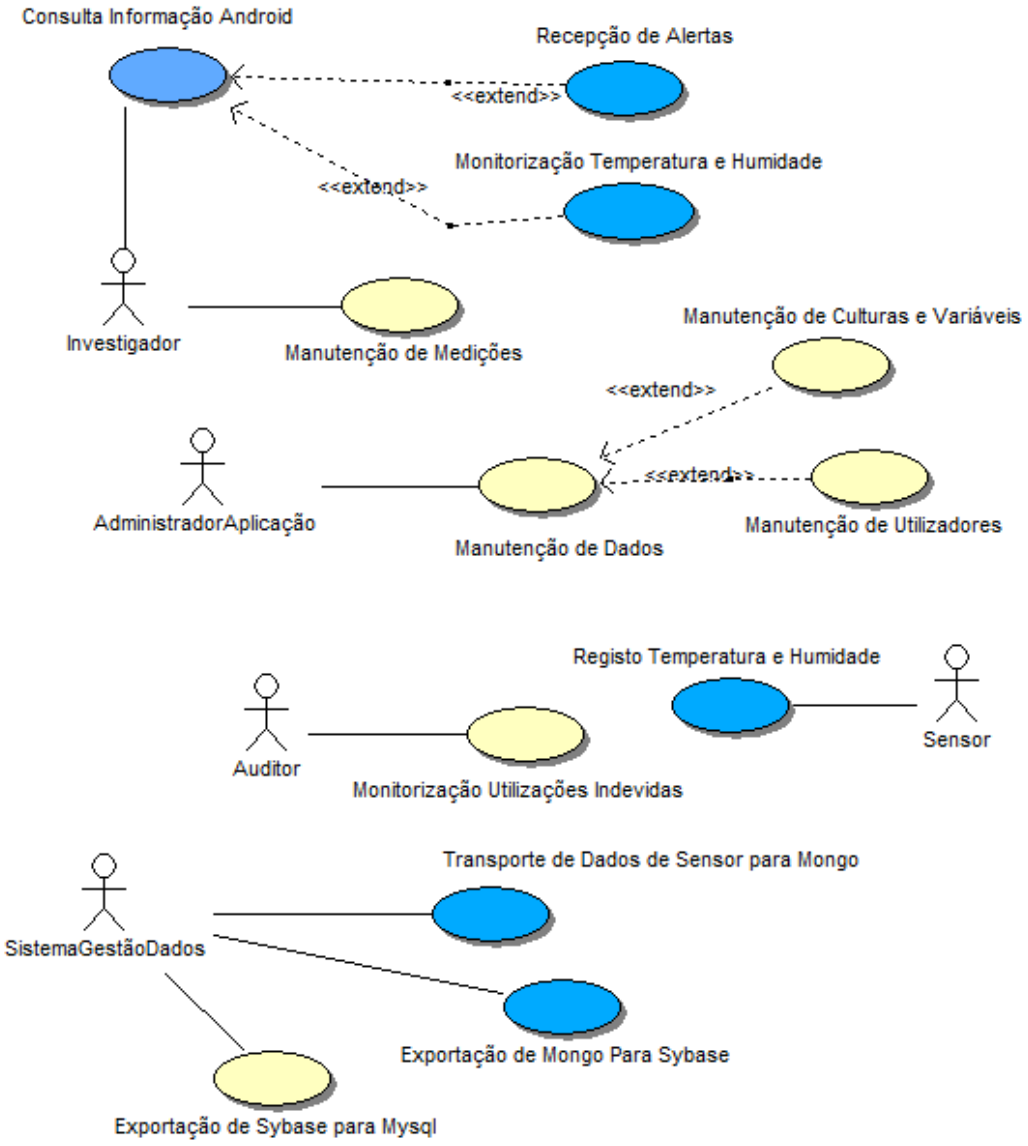
Existe um sensor que periodicamente *lê* a temperatura e humidade no laboratório. Os dados são registados na base de dados (classe HumidadeTemperatura), e pretende-se que sejam utilizados para emitir alertas (o sistema sabe o intervalo de valores de humidade e temperatura ideal para cada cultura) e para tentar *explicar* eventuais valores anómalos de variáveis (por exemplo, "detecta-se que sempre que a temperatura desce bruscamente – mais do que 5 graus em menos de uma hora – a concentração de ferro no pimento apresenta valores anormalmente baixos").

Cada investigador deverá ter a possibilidade de, através de um telemóvel, monitorizar a evolução da temperatura e humidade (não apenas a última leitura, mas a evolução da última hora ou horas) e receber alertas relativos a variações bruscas nós valores das variáveis das suas culturas.

É necessário guardar no sybase o registo de todas as operações de escrita sobre todas as tabelas (qua dados foram alterados/inseridos/apagados, quando e por quem) e registo de operações de consulta sobre a tabela Medições. Esse registo de alterações (*log) é exportado* incrementalmente (apenas informação nova) e periodicamente para uma base de dados autónoma (mysql). Através dessa base de dados (apenas de consulta) um auditor pode analisar se ocorreram utilizações abusivas dos dados (por exemplo, verificar se um investigador tentou

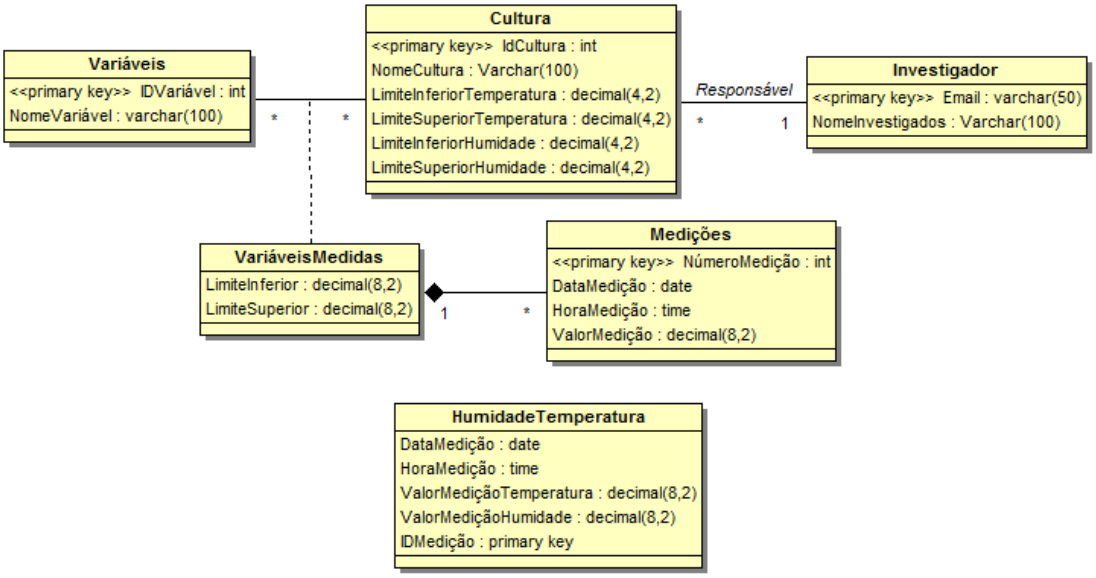
ler medições de culturas que não as suas, quem é que alterou limites de Temperatura de uma cultura, etc.).

Diagrama de Use Case Global

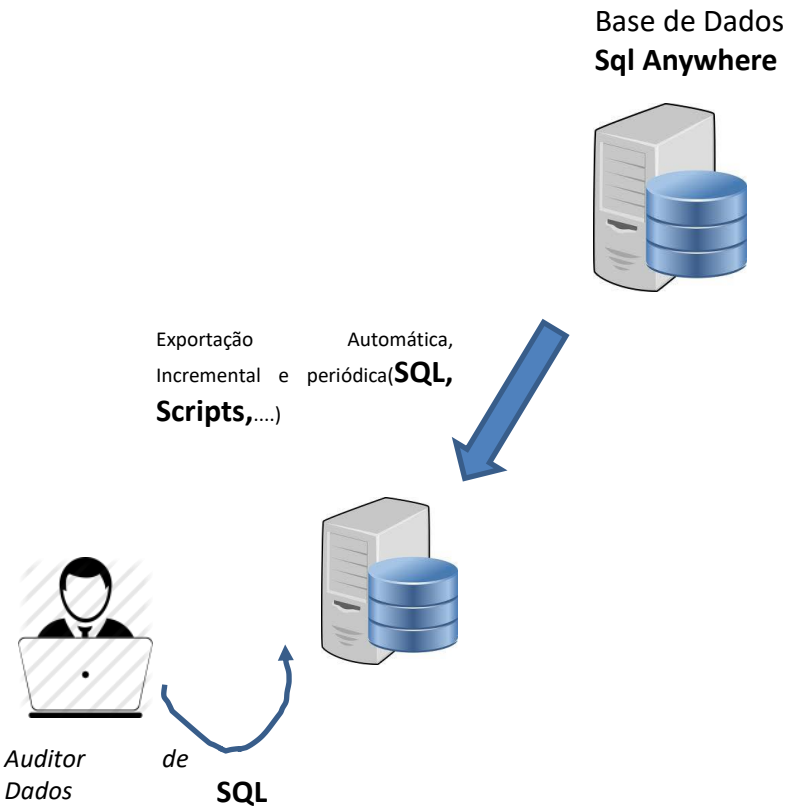


No presente relatório não são contemplados os use case assinalados com cor mais escura. Nenhum use case pressupõe a programação de formulários. As manutenções são efectuadas através de comandos SQL e/ou Storedd Procedures/Triggers (interactive sql), recorrendo a utilizadores e grupos de utilizadores do Sql Anywhere.

Diagrama de Classes de Suporte à Base de Dados

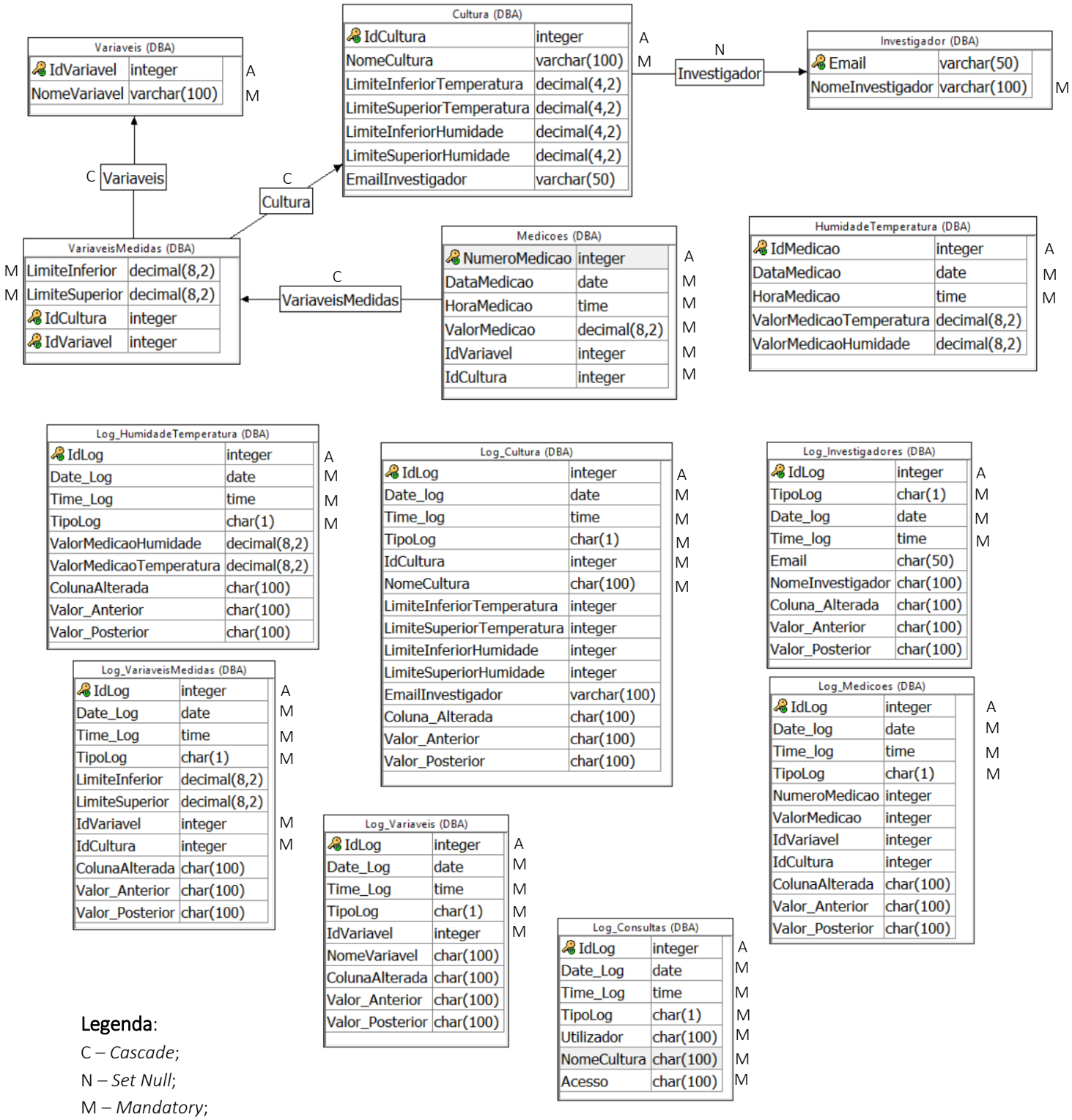


Esquema de Migração



1 Etapa A e B

1.1 Esquema relacional da base de Dados Sybase (origem)



Legenda:

C – *Cascade*;

N – *Set Null*;

M – *Mandatory*;

A – *autoincrement*;

1.1.1 Apreciação Crítica e esquema relacional implementado

Qualidade (Fraca, Razoável, Boa ou Muito Boa): Razoável

Foram encontradas essencialmente 6 categorias de potenciais problemas:

1. Existência de colunas destinadas a acolher conteúdos semelhantes em diversas tabelas com definições de tipos de dados e/ou comprimentos diferentes, o que poderá comprometer a comparabilidade dos respetivos valores;
2. Estando em causa uma associação do tipo "composição" no diagrama de classes original, a chave estrangeira proveniente da tabela "VariaveisMedidas" deveria entrar na composição da chave da tabela "Medições", o que não se verifica. Deste modo, não está garantida a por vezes chamada "relação de morte" entre os objetos relacionados;
3. Com exceção da tabela "LogConsultas", nenhuma das restantes tabelas de log regista o utilizador que faz a operação ("insert", "update" ou "delete").
4. A tabela "LogHumidadeTemperatura" só tem a data e hora do log e não guarda a data e hora da medição original, por isso quem estiver a consultar o log não sabe quando foi efectuada a medição que estão a observar.
5. Na tabela LogConsultas existem 2 atributos "Utilizador" e "Acesso", o que não torna claro o que se pretende em cada um dos campos, nem existe explicação na especificação. Considerou-se que o "Acesso" seria para registar quem consultou as medições e o "Utilizador" quem é responsável pela cultura correspondente à medição, logo quem supostamente efectuou o registo da medição (algo que não é garantido porque na especificação não é feita a salvaguarda do acesso à tabela "Medições" por parte dos investigadores responsáveis pela cultura).
6. Na tabela LogConsultas o atributo "TipoLog" é redundante, uma vez que nessa tabela apenas são guardados comandos do tipo SELECT.

Foram feitas alterações? (Sim/Não): Não

Novo Esquema (assinale e justifique as alterações)

<Apenas preencher caso tenham procedido a alterações>

1.2 Utilizadores

Em baixo, encontram-se as tabelas utilizadas na nossa BD, os utilizadores que têm acesso à mesma e as suas permissões em cada tabela (E / L). Encontram-se também todos os *Stored Procedures* utilizados nesta etapa com as respetivas permissões de execução (X / -).

Os *Stores Procedures* são explicados com mais detalhe nas tabelas de SP mais à frente no relatório.

Tabela	Tipo de Utilizador	
	Investigadores	Administrador
Cultura	-	E, L
HumidadeTemperatura	-	E
Logs (Todas as tabelas Log estão incluídas nesta secção)	-	L
Medições	-	L
Variaveis	-	E, L
VariaveisMedidas	-	E, L
Stored Proc. EXTRAS		
createVariavelMedida **(*1)**	-	X
deleteMedicoes	X	-
insertMedicoes	X	-
updateMedicoes	X	-
consultCulturas	X	-
consultHumidadeTemperatura	X	-
consultVariaveis	X	-
consultVariaveisMedidas	X	-
Stored Proc. Suporte à Manutenção de Utilizadores		
createAdministrador **(*2)**	-	-
createInvestigador	-	X
deleteInvestigador	-	X
Stored Proc. Suporte à Criação Logs		
consultMedicoes	X	-
Stored Proc. Suporte à Migração de Dados		
Export_to_Mysql	-	-

(*1) Procedimento para facilitar a ligação de variáveis com a cultura.

(*2) Procedimento utilizado pelo DBA.

Stored Procedures EXTRAS (que considerámos úteis para uma boa utilização da BD)

Os Stored Procedures descritos em baixo, são SPs extra que considerámos úteis para uma boa utilização da Base de Dados.

Nome Procedimento	Parâmetros Entrada	Parâmetros Saída	Muito breve descrição
createVariavelMedida	cultura VARCHAR(100), variavel VARCHAR(100), limite_inferior DECIMAL(8,2), limite_superior DECIMAL(8,2)	-	
deleteMedicoes	numero_medicao INTEGER	-	
insertMedicoes	valor DECIMAL(8,2), variavelName VARCHAR(100), culturaName VARCHAR(100)	-	
updateMedicoes	valor DECIMAL(8,2), numero_medicao INTEGER	-	
consultCulturas	-	identificador INTEGER, Nome_Culturas VARCHAR(100), Investigador VARCHAR(100)	
consultHumidadeTemperatura	-	Id_mediçao INTEGER, Date DATE, Hora TIME, Valor_Temperatura DECIMAL (8,2), Valor_Humidade DECIMAL(8,2)	
consultVariaveis	-	identificador INTEGER, Nome_Variavel VARCHAR(100)	
consultVariaveisMedidas	-	Nome_Cultura VARCHAR(100), Nome_Variavel VARCHAR(100), limite_inferior DECIMAL(8,2), limite_superior DECIMAL (8,2)	

Stored Procedures Suporte à Manutenção de Utilizadores

Nome Procedimento	Parâmetros Entrada	Parâmetros Saída	Muito breve descrição
createAdministrador	e_mail VARCHAR(50), pswd VARCHAR(50)	-	
createInvestigador	nome VARCHAR(100), e_mail VARCHAR(50), pswd VARCHAR(50)	-	
deleteInvestigador	e_mail VARCHAR(100)	-	

1.2.1 Apreciação Crítica a Gestão de Utilizadores

Qualidade (Fraca, Razoável, Boa ou Muito Boa): Razoável

Análise crítica (clareza, completude, rigor):
A análise da especificação deste capítulo terá de ser dividas em duas áreas: Gestão de Utilizadores e Stored Procedures, uma vez que neste capítulo ambas as componentes foram abordadas.

Stored Procedures:
Na apreciação crítica das SP's definidas na especificação serão apontados alguns casos onde a clareza, completude e rigor podiam ter sido melhor explorados.

- Clareza

consultCulturas - A falta de indicação dos objectivos desta SP tão genérica, levou que considerá-se-mos que o objectivo seria ver uma lista com todas as culturas sem qualquer filtro por user ou outro atributo, pois não é colocado nenhum parâmetro de entrada.

- Completude

deleteMedicoes/insertMedicoes/updateMedicoes - Estas 3 SP's deveriam prever que apenas o investigador responsável da cultura alterasse as medições, mas não são passados os parâmetros de entrada necessários para se fazer tal verificação.

createAdministrador - Nada é especificado sobre a criação de grupos. Assumimos que os grupos foram criados à priori.

createInvestigador/deleteInvestigador - Não foi mencionado no detalhe deste SP se tinha alguma funcionalidade para sincronizar a tabela de Investigadores e os logins na base de dados Sybase. Visto esta falta de informação, assumimos que não foi considerado, implementando apenas alterações aos registos de login na base de dados. Deste modo, os administradores ao registarem um novo Investigador, terão de correr o SP e adicionar manualmente a respetiva entrada na tabela Investigador.

- Rigor

insertMediçoes/consultVariaveisMedidas - Nestas SP's deveriam ser o IdVariavel (integer) e IdCultura (integer) passados como parâmetros de entrada e saída, pois são os atributos que entram na tabela de medições. Na implementação foram essas as variáveis consideradas e não as especificadas.
consultCulturas - O nome dado ao IdCultura, vai variando ao longo da especificação, ora sendo referido como

"Nome_Cultura", "culturaName" ou "identificador". Por uma questão de consistência deveriam manter-se as designações dos atributos.

De uma forma geral os SP's especificados são úteis para uma utilização mais intuitiva da Base de Dados, no entanto, ao serem especificadas algumas SP's para determinadas tabelas também deveriam ser descritas outras SP's para as restantes tabelas, como por exemplo, uma "createCultura" ou uma "createInvestigador".

Gestão de Utilizadores:

Nesta secção iremos abordar as decisões do grupo em relação às permissões para tabelas e acesso a Stored Procedures.

- Clareza

Na tabela HumidadeTemperatura, é atribuída uma permissão de Escrita (E) sobre os Administradores. Isto foi interpretado como o acesso da base de dados Mongo, que irá transportar os dados dos sensores. No entanto, não é referido, apesar de não se encontrar de acordo com o especificado no use cases. Concluímos que irá ligar-se à base de dados Sybase através de uma conta com permissões de administrador.

- Completude

A decisão de ter a base de dados Mongo a ligar-se ao Sybase com conta de administrador poderá levar a problemas na ligação em termos de segurança e fiabilidade. Caso alguém tenha acesso direto ao MongoDB, e obtiver credenciais de login, poderá entrar livremente na base de dados Sybase em cargo de Administrador, por exemplo.
Mais na permissão de Escrita à tabela HumidadeTemperatura para os Administradores, no contexto do problema, nesta tabela apenas são introduzidos dados provenientes dos sensores, não sendo necessário updates e deletes. Isto apenas irá prejudicar no controlo de dados e ações do sistema: qualquer administrador terá permissão para eliminar dados brutos caso assim o entenda.

- Rigor

Como referido anteriormente, na organização da tabela foram colocadas apenas duas opções para tabelas: Escrita (E) e Leitura (L). Isto irá impôr uma grande limitação à escolha de permissões por tabela e por utilizador, podendo ter sido distribuídas permissões mais detalhadas: inserts, updates, etc...

A decisão de ter sido atribuídas apenas permissões de SPs aos Investigadores cria uma certa robustez, no entanto a falta de clareza e rigor na definição das permissões para

tabelas cria uma certa confusão e eventuais problemas no sistema, que poderiam ser contornados, adaptando certas mudanças.

Solução Implementada:

Implementamos de acordo com o especificado, no entanto tivemos de assumir certos detalhes, devido à falta de clareza das mesmas:

- Assumimos que a Escrita (E) referia-se a inserts, updates e também deletes.
- Todos os Stored Procedures de Suporte à Manutenção de Investigadores não tinham nenhum sincronismo com as tabelas de Investigadores.
- Acrescentamos permissões extras de Escrita e Leitura para a tabela Investigador, devido à inexistência desse mesmo sincronismo, mencionado acima.

1.3 Gestão de Logs

1.3.1 Triggers de suporte à criação de logs

Nome Trigger	Tabela	Tipo de Operação (I, U, D)	Evento (A, B)
Log_Create_Cultura	Cultura	I	A
Log_Create_ HumidadeTemperatura	Humidade Temperatura	I	A
Log_Create_Investigador	Investigador	I	A
Log_Create_Medicao	Medicao	I	A
Log_Create_VariaveisMedidas	VariaveisMedidas	I	A
Log_Create_Variavel	Variaveis	I	A
Log_Delete_Cultura	Cultura	D	A
Log_Delete_ HumidadeTemperatura	Humidade Temperatura	D	A
Log_Delete_Investigador	Investigador	D	A
Log_Delete_Medicao	Medicao	D	A
Log_Delete_VariaveisMedidas	VariaveisMedidas	D	A
Log_Delete_Variavel	Variaveis	D	A
Log_Update_Cultura	Cultura	U	A
Log_Update_ HumidadeTemperatura	Humidade Temperatura	U	A
Log_Update_Investigador	Investigador	U	A
Log_Update_Medicao	Medicao	U	A
Log_Update_VariaveisMedidas	VariaveisMedidas	U	A
Log_Update_Variavel	Variaveis	U	A

1.3.1.1 Apreciação Crítica de triggers

Qualidade (Fraca, Razoável, Boa ou Muito Boa): Boa

Os triggers especificados para a criação dos logs são os suficientes para se salvaguardar que todas as operações sobre a base de dados fiquem registadas numa ou noutra tabela de logs.
Todas as tabelas de log têm os atributos "Coluna Alterada", "Valor Anterior" e "Valor Posterior", no entanto, este conjunto de atributos sugere que na especificação apenas é considerado que as alterações a cada entrada duma qualquer tabela se fazem para um atributo de cada vez. Ou seja, apenas se altera um atributo de cada vez que se faz um update, algo que no nosso entender não corresponde à realidade e que um utilizador pode querer fazer update a mais do que um atributo.
Por outro lado nas tabelas de log falta um atributo importante, que registe o utilizador que procedeu à alteração, insert, delete ou update.

Lista de Triggers (para cada trigger assinalar com x em célula correspondente)

	Implementado de Acordo com Especificado	Implementado mas diferente de Especificado	Não Implementado	Não Especificado (criado de novo)
Log_Create_Cultura	X			
Log_Create_HumidadeTemperatura	X			
Log_Create_Investigador	X			
Log_Create_Medicao	X			
Log_Create_VariaveisMedidas	X			
Log_Create_Variavel	X			
Log_Delete_Cultura	X			
Log_Delete_HumidadeTemperatura	X			
Log_Delete_Investigador	X			
Log_Delete_Medicao	X			
Log_Delete_VariaveisMedidas	X			
Log_Delete_Variavel	X			

Log_Update_Cultura		X		
Log_Update_ HumidadeTemperatura		X		
Log_Update_Investigador		X		
Log_Update_Medicao		X		
Log_Update_VariaveisMedidas		X		
Log_Update_Variavel		X		

1.3.1.2 Triggers Implementados

```
1. Nome Trigger: Log_Create_Cultura
// Registo na tabela log respetiva o registo criado
CREATE TRIGGER "Log_Create_Cultura" AFTER INSERT
ORDER 1 ON "DBA"."CULTURA"
REFERENCING NEW AS new_ins
FOR EACH ROW
BEGIN
  INSERT INTO Log_Cultura (DATELOG,
    TIME_LOG,
    TIPOLOG,
    IDCULTURA,
    NOMECULTURA,
    limiteInferiorTemperatura,
    limiteSuperiorTemperatura,
    limiteInferiorHumidade,
    limiteSuperiorHumidade,
    EMAILINVESTIGADOR)
  VALUES (date(now()),
    datetime(now()),
    'I',
    new_ins.IDCULTURA,
    new_ins.NOMECULTURA,
    new_ins.limiteInferiorTemperatura,
    new_ins.limiteSuperiorTemperatura,
    new_ins.limiteInferiorHumidade,
    new_ins.limiteSuperiorHumidade,
    new_ins.EMAILINVESTIGADOR;
END;


2. Nome Trigger: Log_Create_HumidadeTemperatura
// Registo na tabela log respetiva o registo criado
CREATE   TRIGGER   "Log_Create_HumidadeTemperatura"   AFTER
INSERT
ORDER 1 ON "DBA"."HUMIDADETEMPERATURA"
REFERENCING NEW AS new_ins
FOR EACH ROW
BEGIN
  INSERT INTO Log_HumidadeTemperatura (DATELOG,
    TIME_LOG,
    TIPOLOG,
    VALORMEDICAOHUMIDADE,
    VALORMEDICAOTEMPERATURA)
  VALUES (date(now()),
    datetime(now()),
    'I',
    new_ins.VALORMEDICAOHUMIDADE,
```

```
    new_ins.VALORMEDICAOTEMPERATURA;
END;
```

3. Nome Trigger: *Log_Create_Investigador*

```
// Registo na tabela log respetiva o registo criado
CREATE TRIGGER "Log_Create_Investigador" AFTER INSERT
ORDER 1 ON "DBA"."INVESTIGADOR"
REFERENCING NEW AS new_ins
FOR EACH ROW
BEGIN
  INSERT INTO LOG_INVESTIGADORES (DATELOG,
    TIME_LOG,
    TIPOLOG,
    EMAIL,
    NOMEINVESTIGADOR)
  VALUES (date(now()),
    datetime(now()),
    'I',
    new_ins.EMAIL,
    new_ins.NOMEINVESTIGADOR);
END;
```

4. Nome Trigger: *Log_Create_Medicao*

```
// Registo na tabela log respetiva o registo criado
CREATE TRIGGER "Log_Create_Medicao" AFTER INSERT
ORDER 1 ON "DBA"."MEDICOES"
REFERENCING NEW AS new_ins
FOR EACH ROW
BEGIN
  INSERT INTO LOG_MEDICOES (DATELOG,
    TIME_LOG,
    TIPOLOG,
    NUMEROMEDICAO,
    VALORMEDICAO,
    IDVARIAVEL,
    IDCULTURA)
  VALUES (date(now()),
    datetime(now()),
    'I',
    new_ins.NUMEROMEDICAO,
    new_ins.VALORMEDICAO,
    new_ins.IDVARIAVEL,
    new_ins.IDCULTURA);
END;
```

```
5. Nome Trigger: Log_Create_VariaveisMedidas
// Registo na tabela log respetiva o registo criado
CREATE TRIGGER "Log_Create_VariaveisMedidas" AFTER INSERT
ORDER 1 ON "DBA"."VARIAVEISMEDIDAS"
REFERENCING NEW AS new_ins
FOR EACH ROW
BEGIN
  INSERT INTO LOG_VARIAVEISMEDIDAS (DATELOG,
    TIME_LOG,
    TIPOLOG,
    LIMITEINFERIOR,
    LIMITESUPERIOR,
    IDVARIAVEL,
    IDCULTURA)
  VALUES (date(now()),
    datetime(now()),
    'I',
    new_ins.LIMITEINFERIOR,
    new_ins.LIMITESUPERIOR,
    new_ins.IDVARIAVEL,
    new_ins.IDCULTURA);
END;


6. Nome Trigger: Log_Create_Variavel
// Registo na tabela log respetiva o registo criado
CREATE TRIGGER "Log_Create_Variavel" AFTER INSERT
ORDER 1 ON "DBA"."VARIAVEIS"
REFERENCING NEW AS new_ins
FOR EACH ROW
BEGIN
  INSERT INTO LOG_VARIAVEIS (DATELOG,
    TIME_LOG,
    TIPOLOG,
    IDVARIAVEL,
    NOMEVARIAVEL)
  VALUES (date(now()),
    datetime(now()),
    'I',
    new_ins.IDVARIAVEL,
    new_ins.NOMEVARIAVEL);
END;



7. Nome Trigger: Log_Delete_Cultura
// Registo na tabela log respetiva o registo eliminado
CREATE TRIGGER "Log_Delete_Cultura" AFTER DELETE
ORDER 1 ON "DBA"."CULTURA"
REFERENCING OLD AS old_del
FOR EACH ROW
```

```
BEGIN
  INSERT INTO Log_Cultura (DATELOG,
    TIME_LOG,
    TIPOLOG,
    IDCULTURA,
    NOMECULTURA,
    limiteInferiorTemperatura,
    limiteSuperiorTemperatura,
    limiteInferiorHumidade,
    limiteSuperiorHumidade,
    EMAILINVESTIGADOR)
  VALUES (date(now()),
    datetime(now()),
    'D',
    old_del.IDCULTURA,
    old_del.NOMECULTURA,
    old_del.limiteInferiorTemperatura,
    old_del.limiteSuperiorTemperatura,
    old_del.limiteInferiorHumidade,
    old_del.limiteSuperiorHumidade,
    old_del.EMAILINVESTIGADOR;
END;


8. Nome Trigger: Log_Delete_HumidadeTemperatura
// Registo na tabela log respetiva o registo eliminado
CREATE   TRIGGER   "Log_Delete_HumidadeTemperatura"   AFTER
DELETE
ORDER 1 ON "DBA"."HUMIDADETEMPERATURA"
REFERENCING OLD AS old_del
FOR EACH ROW
BEGIN
  INSERT INTO Log_HumidadeTemperatura (DATELOG,
    TIME_LOG,
    TIPOLOG,
    VALORMEDICAOHUMIDADE,
    VALORMEDICAOTEMPERATURA)
  VALUES (date(now()),
    datetime(now()),
    'D',
    old_del.VALORMEDICAOHUMIDADE,
    old_del.VALORMEDICAOTEMPERATURA;
END;


9. Nome Trigger: Log_Delete_HumidadeTemperatura
// Registo na tabela log respetiva o registo eliminado
CREATE TRIGGER "Log_Delete_Investigador" AFTER DELETE
ORDER 1 ON "DBA"."INVESTIGADOR"
REFERENCING OLD AS old_del
FOR EACH ROW
```

```
BEGIN
  INSERT INTO LOG_INVESTIGADORES (DATELOG,
    TIME_LOG,
    TIPOLOG,
    EMAIL,
    NOMEINVESTIGADOR)
  VALUES (date(now()),
    datetime(now()),
    'D',
    old_del.EMAIL,
    old_del.NOMEINVESTIGADOR);
END;
```

10. Nome Trigger: *Log_Delete_Medicao*

```
// Registo na tabela log respetiva o registo eliminado
CREATE TRIGGER "Log_Delete_Medicao" AFTER DELETE
ORDER 1 ON "DBA"."MEDICOES"
REFERENCING OLD AS old_del
FOR EACH ROW
BEGIN
  INSERT INTO LOG_MEDICOES (DATELOG,
    TIME_LOG,
    TIPOLOG,
    NUMEROMEDICAO,
    VALORMEDICAO,
    IDVARIAVEL,
    IDCULTURA)
  VALUES (date(now()),
    datetime(now()),
    'D',
    old_del.NUMEROMEDICAO,
    old_del.VALORMEDICAO,
    old_del.IDVARIAVEL,
    old_del.IDCULTURA);
END;
```

11. Nome Trigger: *Log_Delete_VariaveisMedidas*

```
// Registo na tabela log respetiva o registo eliminado
CREATE TRIGGER "Log_Delete_VariaveisMedidas" AFTER DELETE
ORDER 1 ON "DBA"."VARIAVEISMEDIDAS"
REFERENCING OLD AS old_del
FOR EACH ROW
BEGIN
  INSERT INTO LOG_VARIAVEISMEDIDAS (DATELOG,
    TIME_LOG,
    TIPOLOG,
    LIMITEINFERIOR,
    LIMITESUPERIOR,
    IDVARIAVEL,
```

```
    IDCULTURA)
  VALUES (date(now()),
    datetime(now()),
    'D',
    old_del.LIMITEINFERIOR,
    old_del.LIMITESUPERIOR,
    old_del.IDVARIAVEL,
    old_del.IDCULTURA);
END;
```

12. Nome Trigger: *Log_Delete_Variavel*

```
// Registo na tabela log respetiva o registo eliminado
CREATE TRIGGER "Log_Delete_Variavel" AFTER DELETE
ORDER 1 ON "DBA"."VARIAVEIS"
REFERENCING OLD AS old_del
FOR EACH ROW
BEGIN
  INSERT INTO LOG_VARIAVEIS (DATELOG,
    TIME_LOG,
    TIPOLOG,
    IDVARIAVEL,
    NOMEVARIAVEL)
  VALUES (date(now()),
    datetime(now()),
    'D',
    old_del.IDVARIAVEL,
    old_del.NOMEVARIAVEL);
END;
```

13. Nome Trigger: *Log_Update_Cultura*

```
// Registo na tabela log respetiva o registo alterado
CREATE TRIGGER "Log_Update_Cultura" AFTER UPDATE
ORDER 1 ON "DBA"."CULTURA"
REFERENCING new as new_upd old as old_upd
FOR EACH ROW
BEGIN
  DECLARE @coluna_alt CHAR(200);
  DECLARE @new_alt CHAR(200);
  DECLARE @old_alt CHAR(200);

  IF  new_upd.NOMECULTURA <> old_upd.NOMECULTURA THEN
  SELECT (@coluna_alt + 'NomeCultura - ')
    INTO @coluna_alt;
  SELECT (@new_alt + new_upd.NOMECULTURA + ' - ')
    INTO @new_alt;
  SELECT (@old_alt + old_upd.NOMECULTURA + ' - ')
    INTO @old_alt;
  ENDIF;
```

```
IF new_upd.limiteInferiorTemperatura <>
  old_upd.limiteInferiorTemperatura THEN
  SELECT (@coluna_alt + 'limiteInferiorTemperatura - ')
  INTO @coluna_alt;
  SELECT (@new_alt +
  cast(new_upd.limiteInferiorTemperatura as varchar) +
  ' - ') INTO @new_alt;
  SELECT (@old_alt +
  cast(old_upd.limiteInferiorTemperatura as varchar) +
  ' - ') INTO @old_alt;
ENDIF;

IF new_upd.LIMITESUPERIORTEMPERATURA <>
  old_upd.LIMITESUPERIORTEMPERATURA THEN
  SELECT (@coluna_alt + 'limiteSuperiorTemperatura - ')
  INTO @coluna_alt;
  SELECT (@new_alt +
  cast(new_upd.limiteSuperiorTemperatura as varchar) +
  ' - ') INTO @new_alt;
  SELECT (@old_alt +
  cast(old_upd.limiteSuperiorTemperatura as varchar) +
  ' - ') INTO @old_alt;
ENDIF;

IF new_upd.limiteInferiorHumidade <>
  old_upd.limiteInferiorHumidade THEN
  SELECT (@coluna_alt + 'limiteInferiorHumidade - ')
  INTO @coluna_alt;
  SELECT (@new_alt +
  cast(new_upd.limiteInferiorHumidade as varchar) +
  ' - ') INTO @new_alt;
  SELECT (@old_alt +
  cast(old_upd.limiteInferiorHumidade as varchar) +
  ' - ') INTO @old_alt;
  ENDIF;

IF new_upd.limiteSuperiorHumidade <>
  old_upd.limiteSuperiorHumidade THEN
  SELECT (@coluna_alt + 'limiteSuperiorHumidade - ')
  INTO @coluna_alt;
  SELECT (@new_alt +
  cast(new_upd.limiteSuperiorHumidade as varchar) +
  ' - ') INTO @new_alt;
  SELECT (@old_alt +
  cast(old_upd.limiteSuperiorHumidade as varchar) +
  ' - ') INTO @old_alt;
  ENDIF;

IF new_upd.EMAILINVESTIGADOR <>
  old_upd.EMAILINVESTIGADOR THEN
  SELECT (@coluna_alt + 'EmailInvestigador - ')
```

```
    INTO @coluna_alt;
    SELECT (@new_alt + new_upd.EMAILINVESTIGADOR + ' - ')
    INTO @new_alt;
    SELECT (@old_alt + old_upd.EMAILINVESTIGADOR + ' - ')
    INTO @old_alt;
  ENDIF;

  INSERT INTO Log_Cultura (DATELOG,
    TIME_LOG,
    TIPOLOG,
    IDCULTURA,
    NOMECULTURA,
    limiteInferiorTemperatura,
    limiteSuperiorTemperatura,
    limiteInferiorHumidade,
    limiteSuperiorHumidade,
    EMAILINVESTIGADOR,
    COLUNAALTERADA,
    VALOR_ANTERIOR,
    VALOR_POSTERIOR)
  VALUES (date(now()),
    datetime(now()),
    'U',
    new_upd.IDCULTURA,
    new_upd.NOMECULTURA,
    new_upd.limiteInferiorTemperatura,
    new_upd.limiteSuperiorTemperatura,
    new_upd.limiteInferiorHumidade,
    new_upd.limiteSuperiorHumidade,
    new_upd.EMAILINVESTIGADOR,
    @coluna_alt,
    @old_alt,
    @new_alt);
END;


14. Nome Trigger: Log_Update_HumidadeTemperatura
// Registo na tabela log respetiva o registo alterado
CREATE   TRIGGER   "Log_Update_HumidadeTemperatura"   AFTER
UPDATE
ORDER 1 ON "DBA"."HUMIDADETEMPERATURA"
REFERENCING new as new_upd old as old_upd
FOR EACH ROW
BEGIN
  DECLARE @coluna_alt CHAR(200);
  DECLARE @new_alt CHAR(200);
  DECLARE @old_alt CHAR(200);

  IF new_upd.VALORMEDICAOHUMIDADE <>
    old_upd.VALORMEDICAOHUMIDADE THEN
    SELECT (@coluna_alt + 'ValorMedicaoHumidade - ')
```

```
    INTO @coluna_alt;
    SELECT (@new_alt + cast(new_upd.VALORMEDICAOHUMIDADE
    as varchar) + ' - ') INTO @new_alt;
    SELECT (@old_alt + cast(old_upd.VALORMEDICAOHUMIDADE
    as varchar) + ' - ') INTO @old_alt;
  ENDIF;

  IF new_upd.VALORMEDICAOTEMPERATURA <>
    old_upd.VALORMEDICAOTEMPERATURA THEN
    SELECT (@coluna_alt + 'ValorMedicaoTemperatura - ')
    INTO @coluna_alt;
    SELECT (@new_alt + cast(new_upd.VALORMEDICAOTEMPERATURA
    as varchar) + ' - ') INTO @new_alt;
    SELECT (@old_alt + cast(old_upd.VALORMEDICAOTEMPERATURA
    as varchar) + ' - ') INTO @old_alt;
  ENDIF;

  INSERT INTO Log_HumidadeTemperatura (DATELOG,
    TIME_LOG,
    TIPOLOG,
    VALORMEDICAOHUMIDADE,
    VALORMEDICAOTEMPERATURA,
    COLUNAALTERADA,
    VALOR_ANTERIOR,
    VALOR_POSTERIOR)
  VALUES (date(now()),
    datetime(now()),
    'U',
    new_upd.VALORMEDICAOHUMIDADE,
    new_upd.VALORMEDICAOTEMPERATURA,
    @coluna_alt,
    @old_alt,
    @new_alt);
END;


15. Nome Trigger: Log_Update_Investigador
// Registo na tabela log respetiva o registo alterado
CREATE TRIGGER "Log_Update_Investigador" AFTER UPDATE
ORDER 1 ON "DBA"."INVESTIGADOR"
REFERENCING new as new_upd old as old_upd
FOR EACH ROW
BEGIN
  DECLARE @coluna_alt CHAR(200);
  DECLARE @new_alt CHAR(200);
  DECLARE @old_alt CHAR(200);

  IF new_upd.EMAIL <> old_upd.EMAIL THEN
    SELECT (@coluna_alt + 'Email - ') INTO @coluna_alt;
    SELECT (@new_alt + cast(new_upd.EMAIL as varchar) +
    ' - ') INTO @new_alt;
```

```
    SELECT (@old_alt + cast(old_upd.EMAIL as varchar) +
    ' - ') INTO @old_alt;
  ENDIF;

  IF new_upd.NOMEINVESTIGADOR <>
    old_upd.NOMEINVESTIGADOR THEN
    SELECT (@coluna_alt + 'NomeInvestigador - ')
    INTO @coluna_alt;
    SELECT (@new_alt +
    cast(new_upd.NOMEINVESTIGADOR as varchar) +
    ' - ') INTO @new_alt;
    SELECT (@old_alt +
    cast(old_upd.NOMEINVESTIGADOR as varchar) + ' - ')
    INTO @old_alt;
  ENDIF;

  INSERT INTO LOG_INVESTIGADORES (DATELOG,
    TIME_LOG,
    TIPOLOG,
    EMAIL,
    NOMEINVESTIGADOR,
    COLUNAALTERADA,
    VALOR_ANTERIOR,
    VALOR_POSTERIOR)
  VALUES (date(now()),
    datetime(now()),
    'U',
    new_upd.EMAIL,
    new_upd.NOMEINVESTIGADOR,
    @coluna_alt,
    @old_alt,
    @new_alt);
END;


16. Nome Trigger: Log_Update_Medicao
// Registo na tabela log respetiva o registo alterado
CREATE TRIGGER "Log_Update_Medicao" AFTER UPDATE
ORDER 1 ON "DBA"."MEDICOES"
REFERENCING new as new_upd old as old_upd
FOR EACH ROW
BEGIN
  DECLARE @coluna_alt CHAR(200);
  DECLARE @new_alt CHAR(200);
  DECLARE @old_alt CHAR(200);

  IF new_upd.NUMEROMEDICAO <> old_upd.NUMEROMEDICAO THEN
    SELECT (@coluna_alt + 'NumeroMedicao - ')
    INTO @coluna_alt;
    SELECT (@new_alt +
    cast(new_upd.NUMEROMEDICAO as varchar) + ' - ')
```

```
    INTO @new_alt;
    SELECT (@old_alt +
    cast(old_upd.NUMEROMEDICAO as varchar) + ' - ')
    INTO @old_alt;
  ENDIF;

  IF new_upd.VALORMEDICAO <> old_upd.VALORMEDICAO THEN
    SELECT (@coluna_alt + 'ValorMedicao - ')
    INTO @coluna_alt;
    SELECT (@new_alt +
    cast(new_upd.VALORMEDICAO as varchar) + ' - ')
    INTO @new_alt;
    SELECT (@old_alt +
    cast(old_upd.VALORMEDICAO as varchar) + ' - ')
    INTO @old_alt;
  ENDIF;

  IF new_upd.IDVARIAVEL <> old_upd.IDVARIAVEL THEN
    SELECT (@coluna_alt + 'IdVariavel - ')
    INTO @coluna_alt;
    SELECT (@new_alt +
    cast(new_upd.IDVARIAVEL as varchar) + ' - ')
    INTO @new_alt;
    SELECT (@old_alt +
    cast(old_upd.IDVARIAVEL as varchar) + ' - ')
    INTO @old_alt;
  ENDIF;

  IF new_upd.IDCULTURA <> old_upd.IDCULTURA THEN
    SELECT (@coluna_alt + 'IdCultura - ') INTO @coluna_alt;
    SELECT (@new_alt + cast(new_upd.IDCULTURA as varchar) +
    ' - ') INTO @new_alt;
    SELECT (@old_alt + cast(old_upd.IDCULTURA as varchar) +
    ' - ') INTO @old_alt;
  ENDIF;

  INSERT INTO LOG_MEDICOES (DATELOG,
    TIME_LOG,
    TIPOLOG,
    NUMEROMEDICAO,
    VALORMEDICAO,
    IDVARIAVEL,
    IDCULTURA,
    COLUNAALTERADA,
    VALOR_ANTERIOR,
    VALOR_POSTERIOR)
  VALUES (date(now()),
    datetime(now()),
    'U',
    new_upd.NUMEROMEDICAO,
    new_upd.VALORMEDICAO,
```

```
    new_upd.IDVARIAVEL,
    new_upd.IDCULTURA,
    @coluna_alt,
    @old_alt,
    @new_alt);
END;
```

17. Nome Trigger: *Log_Update_VariaveisMedidas*

```
// Registo na tabela log respetiva o registo alterado
CREATE TRIGGER "Log_Update_VariaveisMedidas" AFTER UPDATE
ORDER 1 ON "DBA"."VARIAVEISMEDIDAS"
REFERENCING new as new_upd old as old_upd
FOR EACH ROW
BEGIN
  DECLARE @coluna_alt CHAR(200);
  DECLARE @new_alt CHAR(200);
  DECLARE @old_alt CHAR(200);

  IF new_upd.LIMITEINFERIOR <> old_upd.LIMITEINFERIOR THEN
    SELECT (@coluna_alt + 'LimiteInferior - ')
    INTO @coluna_alt;
    SELECT (@new_alt +
    cast(new_upd.LIMITEINFERIOR as varchar) + ' - ')
    INTO @new_alt;
    SELECT (@old_alt +
    cast(old_upd.LIMITEINFERIOR as varchar) + ' - ')
    INTO @old_alt;
  ENDIF;

  IF new_upd.LIMITESUPERIOR <> old_upd.LIMITESUPERIOR THEN
    SELECT (@coluna_alt + 'LimiteSuperior - ')
    INTO @coluna_alt;
    SELECT (@new_alt +
    cast(new_upd.LIMITESUPERIOR as varchar) + ' - ')
    INTO @new_alt;
    SELECT (@old_alt +
    cast(old_upd.LIMITESUPERIOR as varchar) + ' - ')
    INTO @old_alt;
  ENDIF;

  IF new_upd.IDVARIAVEL <> old_upd.IDVARIAVEL THEN
    SELECT (@coluna_alt + 'IdVariavel - ')
    INTO @coluna_alt;
    SELECT (@new_alt +
    cast(new_upd.IDVARIAVEL as varchar) + ' - ')
    INTO @new_alt;
    SELECT (@old_alt +
    cast(old_upd.IDVARIAVEL as varchar) + ' - ')
    INTO @old_alt;
  ENDIF;
```

```
  IF new_upd.IDCULTURA <> old_upd.IDCULTURA THEN
    SELECT (@coluna_alt + 'IdCultura - ') INTO @coluna_alt;
    SELECT (@new_alt +
    cast(new_upd.IDCULTURA as varchar) + ' - ')
    INTO @new_alt;
    SELECT (@old_alt +
    cast(old_upd.IDCULTURA as varchar) + ' - ')
    INTO @old_alt;
  ENDIF;

  INSERT INTO LOG_VARIAVEISMEDIDAS (DATELOG,
    TIME_LOG,
    TIPOLOG,
    LIMITEINFERIOR,
    LIMITESUPERIOR,
    IDVARIAVEL,
    IDCULTURA,
    COLUNAALTERADA,
    VALOR_ANTERIOR,
    VALOR_POSTERIOR)
  VALUES (date(now()),
    datetime(now()),
    'U',
    new_upd.LIMITEINFERIOR,
    new_upd.LIMITESUPERIOR,
    new_upd.IDVARIAVEL,
    new_upd.IDCULTURA,
    @coluna_alt,
    @old_alt,
    @new_alt);
END;
```

18. Nome Trigger: *Log_Update_Variavel*

```
// Registo na tabela log respetiva o registo alterado
CREATE TRIGGER "Log_Update_Variavel" AFTER UPDATE
ORDER 1 ON "DBA"."VARIAVEIS"
REFERENCING new as new_upd old as old_upd
FOR EACH ROW
BEGIN
  DECLARE @coluna_alt CHAR(200);
  DECLARE @new_alt CHAR(200);
  DECLARE @old_alt CHAR(200);

  IF new_upd.IDVARIAVEL <> old_upd.IDVARIAVEL THEN
    SELECT (@coluna_alt + 'IdVariavel - ')
    INTO @coluna_alt;
    SELECT (@new_alt +
    cast(new_upd.IDVARIAVEL as varchar) + ' - ')
    INTO @new_alt;
    SELECT (@old_alt +
```

```
    cast(old_upd.IDVARIAVEL as varchar) + ' - ')
    INTO @old_alt;
  ENDIF;

  IF new_upd.NOMEVARIAVEL <> old_upd.NOMEVARIAVEL THEN
    SELECT (@coluna_alt + 'NomeVariavel - ')
    INTO @coluna_alt;
    SELECT (@new_alt +
    cast(new_upd.NOMEVARIAVEL as varchar) + ' - ')
    INTO @new_alt;
    SELECT (@old_alt +
    cast(old_upd.NOMEVARIAVEL as varchar) + ' - ')
    INTO @old_alt;
  ENDIF;

  INSERT INTO LOG_VARIAVEIS (DATELOG,
    TIME_LOG,
    TIPOLOG,
    IDVARIAVEL,
    NOMEVARIAVEL,
    COLUNAALTERADA,
    VALOR_ANTERIOR,
    VALOR_POSTERIOR)
  VALUES (date(now()),
    datetime(now()),
    'U',
    new_upd.IDVARIAVEL,
    new_upd.NOMEVARIAVEL,
    @coluna_alt,
    @old_alt,
    @new_alt);
END;
```

1.3.2 Stored Procedures de suporte à criação de logs (**se relevante**)

Nome Procedimento	Parâmetros Entrada	Parâmetros Saída	Muito breve descrição
cunsultMedicoes	cultureName VARCHAR(100)	NumeroMedicao INTEGER, NomeCultura VARCHAR(100), Valor DECIMAL(8,2), LimiteInferior DECIMAL(8,2), LimiteSuperior DECIMAL(8,2), Investigador VARCHAR(100)	

Este é o único SP que se encontra nesta secção visto que, segundo a especificação do enunciado ("registo de operações de consulta sobre a tabela Medições") é a único tipo de consulta que tem de ficar registada na tabela de Logs.

1.3.2.1 Apreciação Crítica de Stored Procedures

Qualidade (Fraca, Razoável, Boa ou Muito Boa): Razoável

O único SP especificado nesta fase é responsável pela consulta da tabela de Medições e consequente registo dessa acção na tabela de Log_Medições.
O objecto desta SP seria "apanhar" as consultas feitas à tabela de Medições por parte dos utilizadores, e esse objectivo é cumprido. No entanto, uma parte muito importante não foi salvaguardada, que consistia na protecção dos dados contra utilizadores que não fossem os responsáveis por determinada consulta.

Lista de SP (para cada SP assinalar com x em célula correspondente)

	Implementado de Acordo com Especificado	Implementado mas diferente de Especificado	Não Implementado	Não Especificado (criado de novo)
cunsultMedicoes	X			
createVariavelMedida	X			
deleteMedicoes	X			
insertMedicoes	X			
updateMedicoes	X			
consultCulturas	X			
consultHumidadeTemperatura	X			
consultVariaveis	X			
consultVariaveisMedidas	X			
createAdministrador	X			
createInvestigador	X			
deleteInvestigador	X			

1.3.2.2 Stored Procedures Implementados

```
1. Nome SP: consultMedicoes
//  Procedimento  que  permite  efetuar  consultas  à  tabela
Medicoes e registar o que foi consultado na tabela log
CREATE PROCEDURE "DBA"."consultMedicoes"(
  IN id_cult INTEGER, OUT id_Medicao INTEGER,
  OUT id_Cultura VARCHAR(100), OUT Valor DECIMAL(8,2),
  OUT LimiteInferior DECIMAL(8,2),
  OUT LimiteSuperior DECIMAL(8,2),
  OUT Investigador VARCHAR(100))
BEGIN
  INSERT INTO LOG_CONSULTAS (DATELOG,
    TIME_LOG,
    TIPOLOG,
    UTILIZADOR,
    NOMECULTURA,
    ACESSO)
  VALUES (date(now()),
    datetime(now()),
    'S',
    (select CULTURA.EMAILINVESTIGADOR
       from CULTURA where cultura.IDCULTURA=id_cult),
         id_cult, user_name());

  SELECT MEDICOES.NUMEROMEDICAO, MEDICOES.IDCULTURA,
    MEDICOES.VALORMEDICAO, VARIAVEISMEDIDAS.LIMITEINFERIOR,
    VARIAVEISMEDIDAS.LIMITESUPERIOR,
    CULTURA.EMAILINVESTIGADOR
  FROM MEDICOES, VARIAVEISMEDIDAS, CULTURA
  WHERE id_cult=MEDICOES.IDCULTURA AND
    MEDICOES.IDVARIAVEL=VARIAVEISMEDIDAS.IDVARIAVEL AND
    CULTURA.IDCULTURA=VARIAVEISMEDIDAS.IDCULTURA;
END;


2. Nome SP: consultCulturas
//  Procedimento  que  permite  efetuar  consultas  à  tabela
Culturas
CREATE PROCEDURE "DBA"."consultCulturas"(
  OUT identificador INTEGER,
  OUT Nome_Culturas VARCHAR(100),
  OUT Investigador VARCHAR(100) )
BEGIN
     SELECT IDCULTURA, NOMECULTURA, EMAILINVESTIGADOR
    FROM CULTURA;
END;
```

```
3. Nome SP: consultHumidadeTemperatura
// Procedimento que permite efetuar consultas à tabela
HumidadeTemperatura
CREATE PROCEDURE "DBA"."consultHumidadeTemperatura"(
  OUT Id_medicao INTEGER,
  OUT Data_medicao DATE,
  OUT Hora TIME,
  OUT Valor_Temperatura DECIMAL (8,2),
  OUT Valor_Humidade DECIMAL(8,2))
BEGIN
  SELECT IDMEDICAO, DATAMEDICAO, HORAMEDICAO,
    VALORMEDICAOTEMPERATURA, VALORMEDICAOHUMIDADE
  FROM HUMIDADETEMPERATURA
END;


4. Nome SP: consultVariaveis
// Procedimento que permite efetuar consultas à tabela
Variaveis
CREATE PROCEDURE "DBA"."consultVariaveis"(
  OUT identificador INTEGER,
  OUT Nome_Variavel VARCHAR(100))
BEGIN
  SELECT IDVARIAVEL, NOMEVARIAVEL
  FROM VARIAVEIS
END;


5. Nome SP: consultVariaveisMedidas
// Procedimento que permite efetuar consultas à tabela
VariaveisMedidas
CREATE PROCEDURE "DBA"."consultVariaveisMedidas"(
  OUT id_Cultura VARCHAR(100),
  OUT id_Variavel VARCHAR(100),
  OUT limite_inferior DECIMAL(8,2),
  OUT limite_superior DECIMAL (8,2))
BEGIN
  SELECT IDCULTURA, IDVARIAVEL, LIMITEINFERIOR,
    LIMITESUPERIOR
  FROM VARIAVEISMEDIDAS
END;


6. Nome SP: createVariavelMedida
// Procedimento que permite criar registos na tabela
VariaveisMedidas
CREATE PROCEDURE "DBA"."createVariavelMedida"(
  IN id_cultura VARCHAR(100),
  IN id_variavel VARCHAR(100),
  IN limite_inferior DECIMAL(8,2),
  IN limite_superior DECIMAL(8,2))
```

```
BEGIN
  INSERT INTO VARIAVEISMEDIDAS (LIMITEINFERIOR,
    LIMITESUPERIOR,
    IDVARIAVEL,
    IDCULTURA)
  VALUES (limite_inferior,
    limite_superior,
    id_variavel,
    id_cultura);
END;
```

7. Nome SP: deleteMedicoes

```
// Procedimento para eliminar registos na tabela Medicoes
CREATE PROCEDURE "DBA"."deleteMedicoes"(
  IN numero_medicao INTEGER )
BEGIN
  DELETE FROM MEDICOES
  WHERE MEDICOES.NUMEROMEDICAO = numero_medicao;
END;
```

8. Nome SP: *insertMedicoes*

```
// Procedimento para inserir registos na tabela Medicoes
CREATE PROCEDURE "DBA"."insertMedicoes"(
  IN valor DECIMAL(8,2),
  IN id_variavel INTEGER ,
  IN id_cultura INTEGER  )
BEGIN
  INSERT INTO MEDICOES (DATAMEDICAO,
    HORAMEDICAO,
    VALORMEDICAO,
    IDVARIAVEL,
    IDCULTURA)
  VALUES (date(now()),
    datetime(now()),
    valor,
    id_variavel,
    id_cultura);
END;
```

9. Nome SP: *updateMedicoes*

```
// Procedimento para atualizar registos na tabela Medicoes
CREATE PROCEDURE "DBA"."updateMedicoes"(
  IN valor DECIMAL(8,2),
  IN numero_medicao INTEGER )
BEGIN
  UPDATE MEDICOES
  SET VALORMEDICAO = valor
  WHERE NUMEROMEDICAO = numero_medicao;
END;
```

10. Nome SP: *createInvestigador*

```
// Procedimento para criar um investigador
CREATE PROCEDURE createInvestigador (
  USERNAME VARCHAR(50) DEFAULT '',
  PASS VARCHAR(50) DEFAULT '')
BEGIN
  IF EXISTS (SELECT * FROM dbo.sysusers
    where dbo.sysusers.name = USERNAME) THEN
    EXECUTE IMMEDIATE 'DROP USER '  || USERNAME
  END IF;
  EXECUTE IMMEDIATE 'CREATE USER '  ||
    USERNAME || ' IDENTIFIED BY ' ||  PASS;

  IF EXISTS (SELECT * FROM dbo.sysusers
    where dbo.sysusers.name = 'Investigadores' ) THEN
    EXECUTE IMMEDIATE 'GRANT MEMBERSHIP IN GROUP
    Investigadores TO ' || USERNAME
  END IF;
END;
```

11. Nome SP: *createAdministrador*

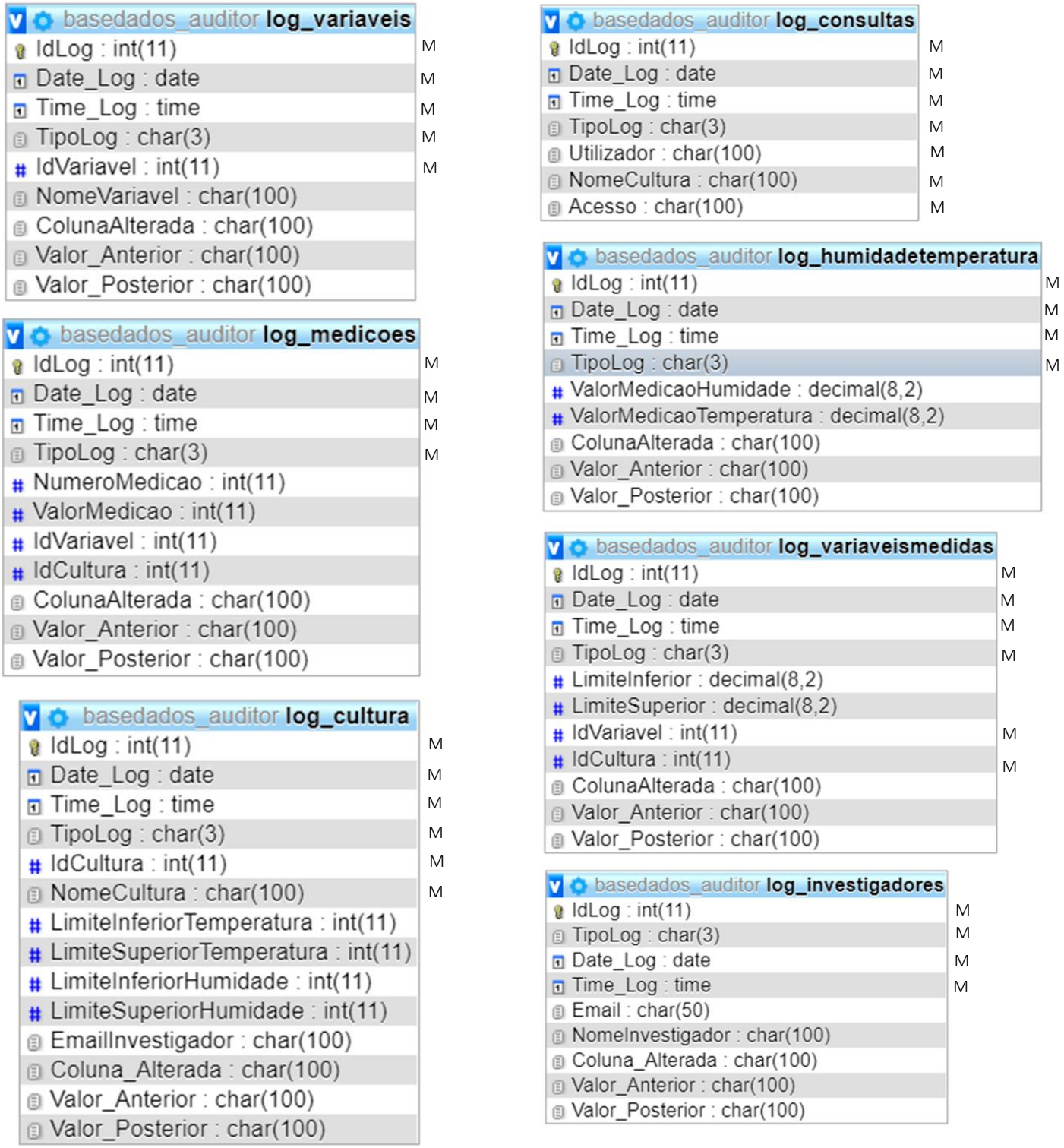
```
// Procedimento para criar um administrador
CREATE PROCEDURE createAdministrador (
  USERNAME VARCHAR(50) DEFAULT '',
  PASS VARCHAR(50) DEFAULT '')
BEGIN
  IF EXISTS (SELECT * FROM dbo.sysusers
    where dbo.sysusers.name = USERNAME) THEN
    EXECUTE IMMEDIATE 'DROP USER '  || USERNAME
  END IF;
  EXECUTE IMMEDIATE 'CREATE USER '  ||
    USERNAME || ' IDENTIFIED BY ' ||  PASS;
  IF EXISTS (SELECT * FROM dbo.sysusers
    where dbo.sysusers.name = 'Administradores' ) THEN
    EXECUTE IMMEDIATE 'GRANT MEMBERSHIP IN GROUP
    Administradores TO ' || USERNAME
  END IF;
END;
```

12. Nome SP: *deleteInvestigador*

```
// Procedimento para eliminar um investigador
CREATE PROCEDURE deleteInvestigador (
  USERNAME VARCHAR(50) DEFAULT '')
BEGIN
  IF EXISTS (SELECT * FROM dbo.sysusers
    where dbo.sysusers.name = USERNAME) THEN
    EXECUTE IMMEDIATE 'DROP USER '  || USERNAME
  END IF;
END;
```

1.4 Migração entre Bases de Dados

1.4.1 Esquema relacional da base de Dados Mysql (destino)



1.4.1.1 Apreciação Crítica e esquema relacional implementado

Qualidade (Fraca, Razoável, Boa ou Muito Boa): Razoável

Foram encontradas essencialmente 3 categorias de potenciais problemas:

1. Existência de colunas destinadas a acolher conteúdos semelhantes em diversas tabelas com definições de tipos de dados e/ou comprimentos diferentes, o que poderá comprometer a comparabilidade dos respetivos valores;
2. Estando em causa uma associação do tipo "composição" no diagrama de classes original, a chave estrangeira proveniente da tabela "VariaveisMedidas" deveria entrar na composição da chave da tabela "Medições", o que não se verifica. Deste modo, não está garantida a por vezes chamada "relação de morte" entre os objetos relacionados;
3. Com exceção da tabela "LogConsultas", nenhuma das restantes tabelas de log regista o utilizador que faz a operação ("insert", "update" ou "delete").

Foram feitas alterações? (Sim/Não): Não

Novo Esquema (assinale e justifique as alterações)

<Apenas preencher caso tenham procedido a alterações>

1.4.2 Forma de Migração

Especificação:

1. Fazer o download e instalar o software "XAMPP", disponível no endereço:

   https://www.apachefriends.org/pt_br/index.html

2. Ligar o XAMPP e acionar o botão "*Start*" nos módulos "*Apache*" e "MYSQL", por esta ordem (ver figura 1).

3. No browser de Internet, ir para o endereço "localhost/phpmyadmin/".

4. Criar a base de dados no MYSQL, usando a interface *phpMyAdmin* (ver a figura 2)
   (i) Utilizar o nome "basedados_auditor" para a base de dados. Caso seja usado outro nome terá que ser modificado posteriormente no ficheiro *bat,* utilizado para fazer a importação.

5. Preencher a base de dados utilizando o esquema relacional disponível no ponto 1.4.1.

6. No *Sybase*, criar o evento "Export_to_Mysql" (observando as figuras 3.1, 3.2, 3.3, 3.4, 3.5, 3.6), que irá chamar o *Stored Procedure* "Export_to_Mysql", de modo a exportar as tabelas de *Log* do *Sybase* para ficheiros ".txt" no Ambiente de trabalho.

7. No Ambiente de trabalho, criar o ficheiro "scrip.txt" e inserir o código do ponto (i).
   Nota: Ter em atenção a diretoria dos ficheiros a ser importados para o *MySQL*, ou seja, será necessário substituir o parâmetro "PC_USER" para o nome do seu utilizador Windows.

   (i)

```
C:/xampp/mysql/bin/mysql.exe -u root -h localhost basedados_auditor -e "LOAD DATA INFILE
'C:/Users/PC_USER /Desktop/Log_Cultura.txt' IGNORE INTO TABLE Log_Cultura fields terminated by ','"

C:/xampp/mysql/bin/mysql.exe -u root -h localhost basedados_auditor -e "LOAD DATA INFILE
'C:/Users/PC_USER/Desktop/Log_HumidadeTemperatura.txt' IGNORE INTO TABLE Log_HumidadeTemperatura fields terminated by ','"

C:/xampp/mysql/bin/mysql.exe -u root -h localhost basedados_auditor -e "LOAD DATA INFILE
'C:/Users/PC_USER/Desktop/Log_Investigadores.txt' IGNORE INTO TABLE Log_Investigadores fields terminated by ','"

C:/xampp/mysql/bin/mysql.exe -u root -h localhost basedados_auditor -e "LOAD DATA INFILE
 'C:/Users/PC_USER/Desktop/Log_Medicoes.txt' IGNORE INTO TABLE Log_Medicoes fields terminated by ','"

C:/xampp/mysql/bin/mysql.exe -u root -h localhost basedados_auditor -e "LOAD DATA INFILE
 'C:/Users/PC_USER/Desktop/Log_Variaveis.txt' IGNORE INTO TABLE Log_Variaveis fields terminated by ','"

C:/xampp/mysql/bin/mysql.exe -u root -h localhost basedados_auditor -e "LOAD DATA INFILE
'C:/Users/PC_USER/Desktop/Log_VariaveisMedidas.txt' IGNORE INTO TABLE Log_VariaveisMedidas fields terminated by ','"

C:/xampp/mysql/bin/mysql.exe -u root -h localhost basedados_auditor -e "LOAD DATA INFILE
'C:/Users/PC_USER/Desktop/Log_Consultas.txt' IGNORE INTO TABLE Log_Consultas fields terminated by ','"
```

8. Guardar o ficheiro “script.txt” como “script.bat” (No campo “*Save as*” tem que estar o valor “*All Files* *.*”).

9. Na barra de pesquisa do Windows, pesquisar “Programador de Tarefas” e correr este programa.

10. Criar um novo evento utilizando as figuras 4.1, 4.2, 4.3, 4.4, 4.5, 4.6.
    Nota: Ao clicar “OK” para concluir a criação do evento, terá que escrever a sua password do Windows.

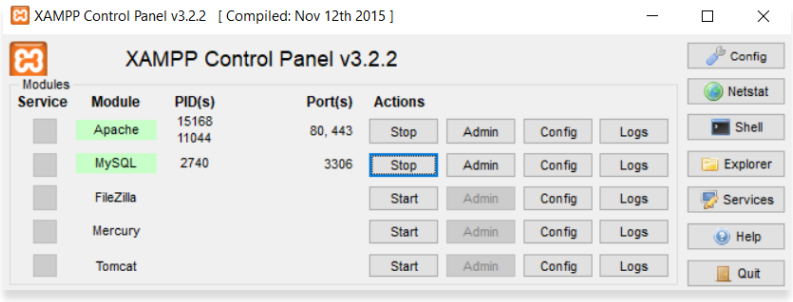


Figura 1

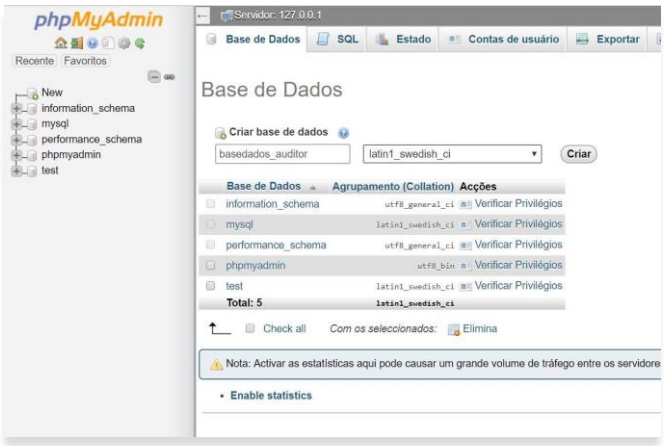


Figura 2

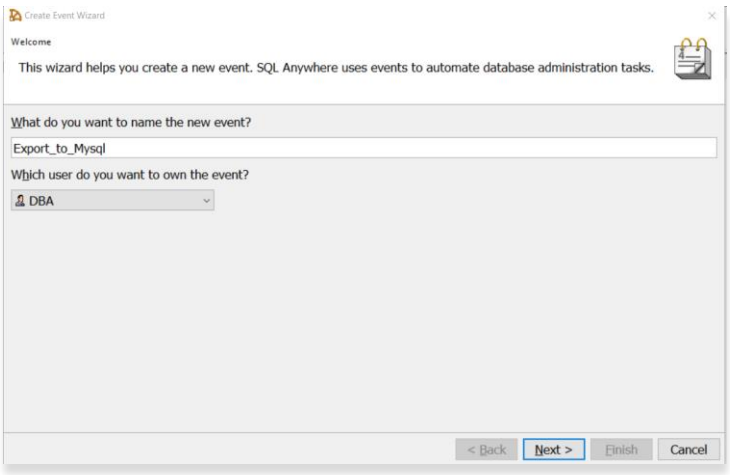


Figura 3.1

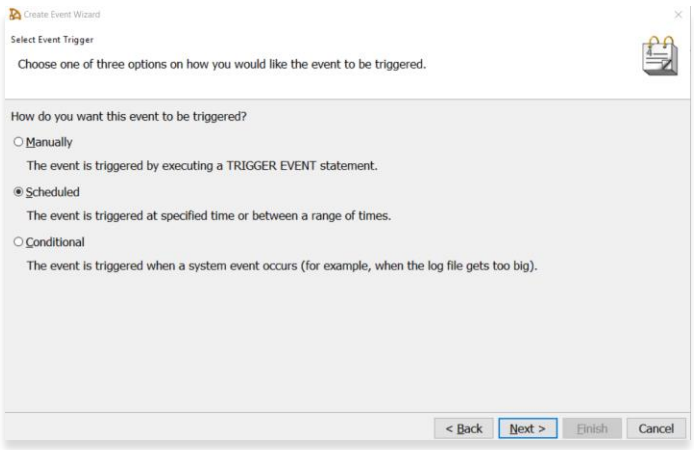


Figura 3.2

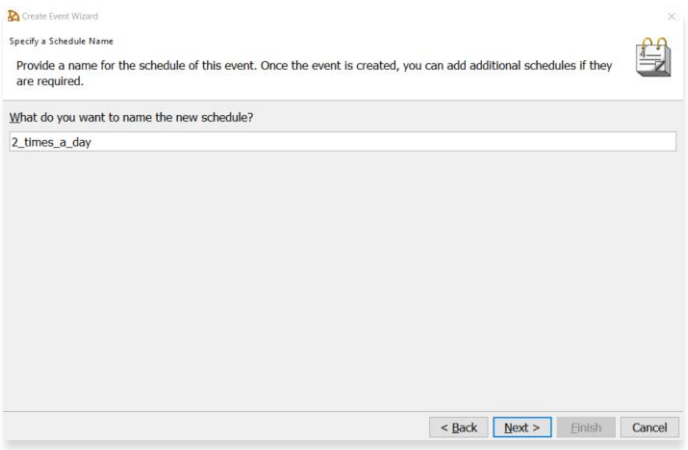


Figura 3.3

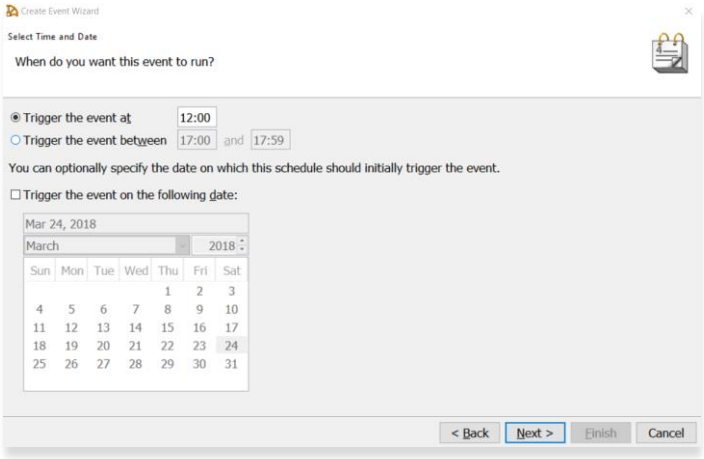


Figura 3.4

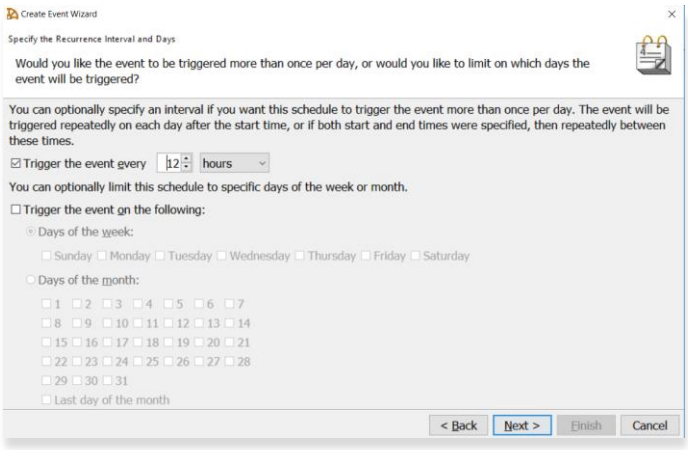


Figura 3.5

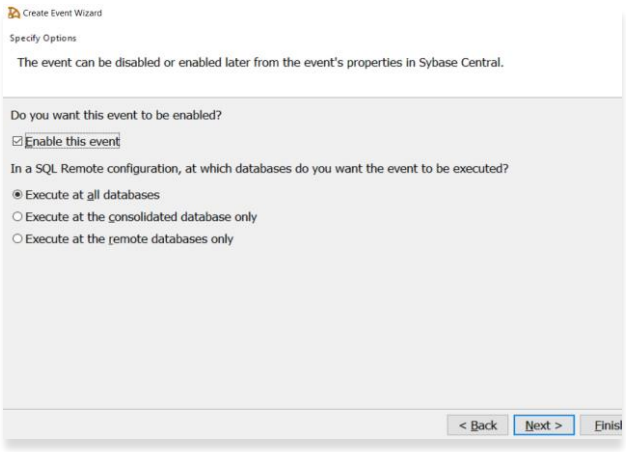


Figura 3.6

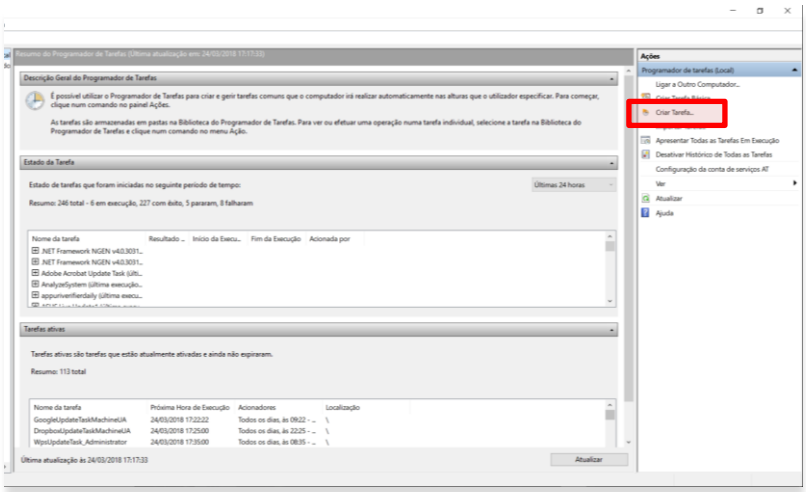


Figura 4.1

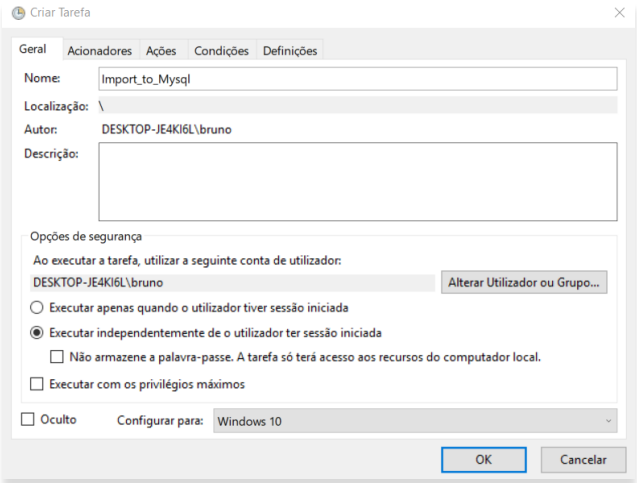


Figura 4.2

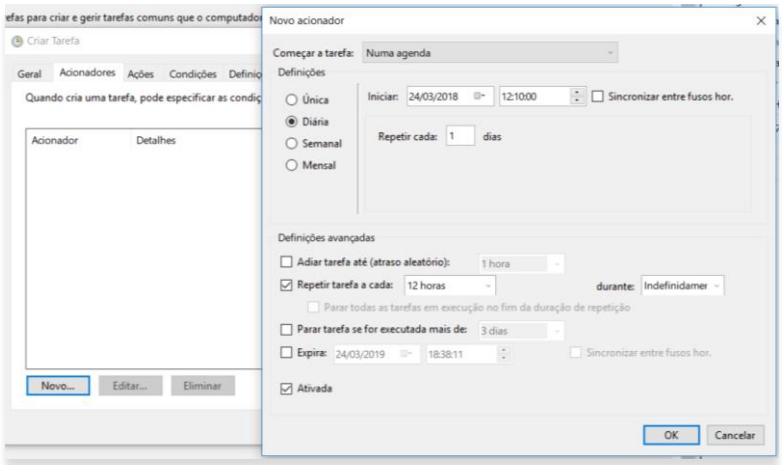


Figura 4.3

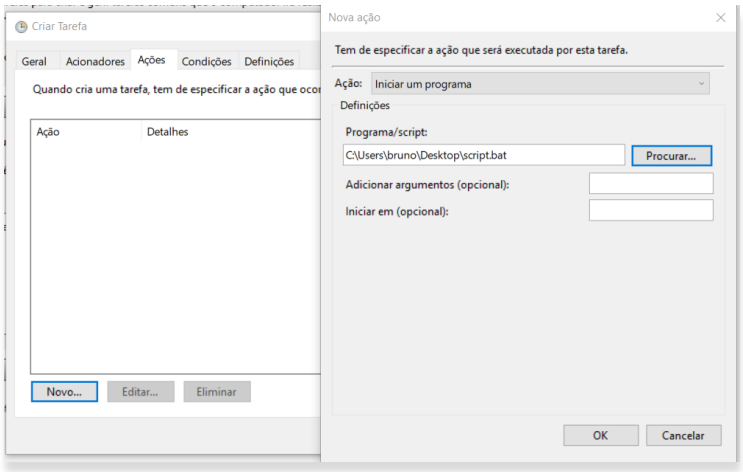


Figura 4.4

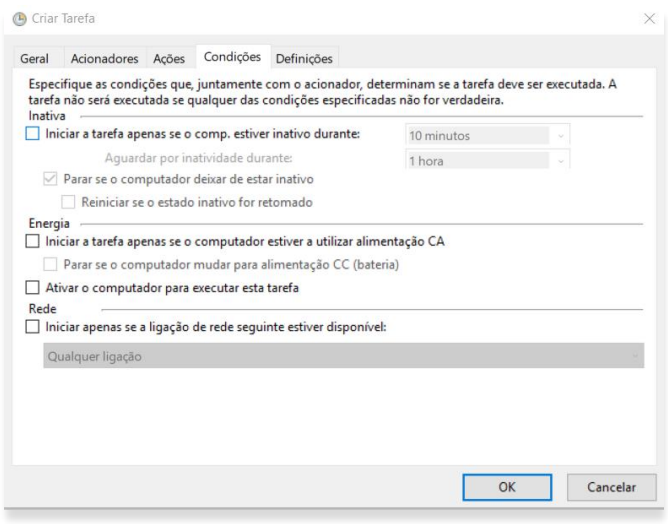


Figura 4.5

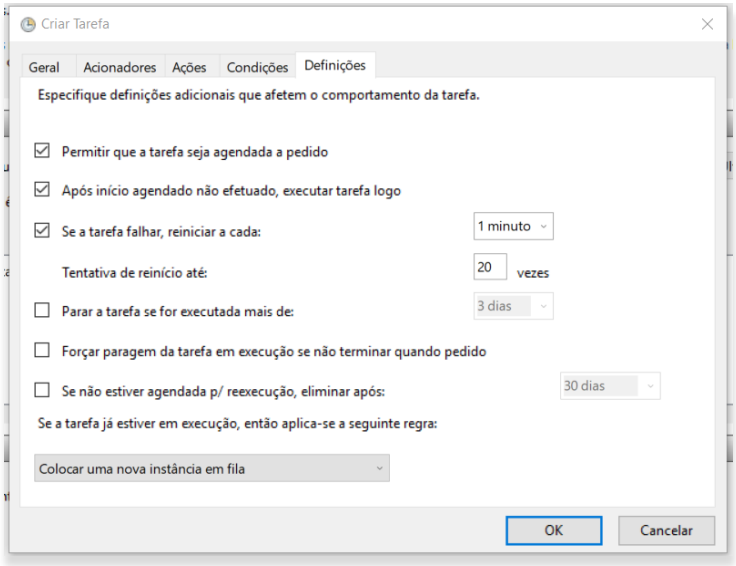


Figura 4.6

Utilizamos o *XAMPP* por ser um software grátis que contém a interface *phpMyAdmin*, que é bastante "user-friendly", permitindo uma fácil navegação e uso da base de dados MySQL.

A base de dados no MySQL foi criada à imagem da base de dados do Sybase, contendo apenas as tabelas de Log, permitindo a importação de dados sem equívocos.

Escolhemos ficheiros ".txt" para fazer a exportação e importação dos dados das tabelas de Log pois permite uma **fácil implementação**, bem como uma **maior rapidez na escrita de dados**.

Optamos por utilizador o Programador de Tarefas por ser um software grátis, de fácil uso e já pré-instalado no Windows.

1.4.2.1 Apreciação Crítica à especificação da forma de migração

Qualidade (Fraca, Razoável, Boa ou Muito Boa): Boa

Análise crítica (clareza, completude, rigor):

1. A especificação contém efetivamente uma "prescrição" do modo de efetuar a migração, bem mais detalhada do que seria exigível, ou até conveniente, num documento de especificação técnica. Tem o mérito de ser completa e clara, mas como exemplifica uma implementação concreta,fica encerrada nesse exemplo e não atende às distinções potencialmente resultantes da implementação numa arquitetura tecnológica alternativa, por exemplo, com máquinas distintas para o Sybase e o MySQL, com efeitos na configuração não apenas dos "path" para o "mysql" e os ficheiros, mas também no formato do comando "LOAD" (LOAD DATA INFILE vs. LOAD DATA LOCAL INFILE);
2. Poderá questionar-se a (in)eficiência resultante da inexistência de mecanismos de controlo alternativos que permitissem evitar a tentativa de inserir todos os registos todas as vezes.

1.4.3 Gestão de Utilizadores

Tabela	Tipo de Utilizador
	Auditor
Log_Consultas	L
Log_Cultura	L
Log_HumidadeTemperatura	L
Log_Investigadores	L
Log_Medicoes	L
Log_Variaveis	L
Log_VariaveisMedidas	L

1.4.3.1 Apreciação Crítica à especificação da Gestão de Utilizadores

Qualidade (Fraca, Razoável, Boa ou Muito Boa): Muito boa

Análise crítica (clareza, completude, rigor):
Nesta secção não impõe bastante dificuldade, devido à utilização de apenas um utilizador: Auditor. No entanto este apenas tem acesso de leitura a todas as tabelas de Logs presentes na base de dados, sendo o mais apropriado.

Solução Implementada:
Foi implementado de acordo com o especificado: atribuir permissões de leitura a todas as tabelas ao Auditor.

1.4.4 Triggers de suporte à migração de dados (**se relevante**)

Não foram aplicados quaisquer triggers à migração de dados

1.4.4.1 Apreciação Crítica de triggers

Sem efeito

1.4.4.2 Triggers Implementados

Sem efeito

1.4.5 Stored Procedures de suporte à migração de dados

Nome Procedimento	Parâmetros Entrada	Parâmetros Saída	BD (Sybase ou MySQL)	Muito breve descrição
Export_to_Mysql	-	-	Sybase	Exportação das tabelas de log para ficheiros ".txt"

Este é o *Stored Procedure* que será chamado pelo evento da exportação automática das tabelas de Log. Quando o evento (especificado no ponto 1.4.6) chama este SP, irá ocorrer a exportação das tabelas Log para o ambiente de trabalho como ficheiros ".txt".

1.4.5.1 Apreciação Crítica de Stored Procedures

Qualidade (Fraca, Razoável, Boa ou Muito Boa): Muito Boa

A SP "Export_to_MySQL" foi implementada conforme especificado.

Lista de SP (para cada SP assinalar com x em célula correspondente)

	Implementado de Acordo com Especificado	Implementado mas diferente de Especificado	Não Implementado	Não Especificado (criado de novo)
Export_to_Mysql	X			

1.4.5.2 Stored Procedures Implementados

```
1. Nome SP: Export_to_Mysql
// Procedure que carrega os dados das tabelas em ficheiros
txt
ALTER PROCEDURE "DBA"."Export_to_MySQL" (IN arg_id INTEGER
DEFAULT NULL)
BEGIN
  IF (arg_id IS NULL) THEN
     UNLOAD TABLE LOG_HUMIDADETEMPERATURA TO
'Z:\\VM_SHAREDFOLDER\\FILEStoEXPORTandIMPORT\\Log_HumidadeT
emperatura.txt';
     UNLOAD TABLE LOG_VARIAVEISMEDIDAS TO
'Z:\\VM_SHAREDFOLDER\\FILEStoEXPORTandIMPORT\\Log_Variaveis
Medidas.txt';
     UNLOAD TABLE LOG_VARIAVEIS TO
'Z:\\VM_SHAREDFOLDER\\FILEStoEXPORTandIMPORT\\Log_Variaveis
.txt';
     UNLOAD TABLE LOG_CULTURA TO
'Z:\\VM_SHAREDFOLDER\\FILEStoEXPORTandIMPORT\\Log_Cultura.t
xt';
     UNLOAD TABLE LOG_INVESTIGADORES TO
'Z:\\VM_SHAREDFOLDER\\FILEStoEXPORTandIMPORT\\Log_Investiga
dores.txt';
     UNLOAD TABLE LOG_MEDICOES TO
'Z:\\VM_SHAREDFOLDER\\FILEStoEXPORTandIMPORT\\Log_Medicoes.
txt';
     UNLOAD TABLE LOG_CONSULTAS TO
'Z:\\VM_SHAREDFOLDER\\FILEStoEXPORTandIMPORT\\Log_Consultas
.txt';
  ELSE
     UNLOAD SELECT TOP arg_id * FROM
LOG_HUMIDADETEMPERATURA TO
'Z:\\VM_SHAREDFOLDER\\FILEStoEXPORTandIMPORT3\\Log_Humidade
Temperatura.txt';
     UNLOAD SELECT TOP arg_id * FROM LOG_VARIAVEISMEDIDAS
TO
'Z:\\VM_SHAREDFOLDER\\FILEStoEXPORTandIMPORT3\\Log_Variavei
sMedidas.txt';
     UNLOAD SELECT TOP arg_id * FROM LOG_VARIAVEIS TO
'Z:\\VM_SHAREDFOLDER\\FILEStoEXPORTandIMPORT3\\Log_Variavei
s.txt';
     UNLOAD SELECT TOP arg_id * FROM LOG_CULTURA TO
'Z:\\VM_SHAREDFOLDER\\FILEStoEXPORTandIMPORT3\\Log_Cultura.
txt';
     UNLOAD SELECT TOP arg_id * FROM LOG_INVESTIGADORES TO
'Z:\\VM_SHAREDFOLDER\\FILEStoEXPORTandIMPORT3\\Log_Investig
adores.txt';
     UNLOAD SELECT TOP arg_id * FROM LOG_MEDICOES TO
```

```
'Z:\\VM_SHAREDFOLDER\\FILEStoEXPORTandIMPORT3\\Log_Medicoes
.txt';
     UNLOAD SELECT TOP arg_id * FROM LOG_CONSULTAS TO
'Z:\\VM_SHAREDFOLDER\\FILEStoEXPORTandIMPORT3\\Log_Consulta
s.txt';
  END IF;
END
```

1.4.6 Eventos de suporte à migração de dados

Nome Evento	Local Execução (Sybase ou MySQL, ou SO)	Muito breve descrição
Export_to_Mysql	Sybase	Este evento invoca o SP "Export_to_Mysql"
Import_to_Mysql	SO	Este evento invoca o ficheiro ".bat", que possui o código necessário para a importação

Estes eventos foram programados para ocorrer duas vezes ao dia (a cada 12h), pois considerámos que este período de tempo tem um valor adequado quando aplicado à vida real.

As horas definidas para a ocorrência do evento "Export_to_Mysql" são às 12:00 e 00:00.

As horas definidas para a ocorrência do evento "Import_to_Mysql" são às 12:10 e 00:10.

1.4.6.1 Apreciação Crítica de Eventos

Qualidade (Fraca, Razoável, Boa ou Muito Boa): Boa

O evento "Export_to_MySQL" foi implementado no Sybase conforme especificado.

O evento "Import_to_MySQL" consiste na definição de uma tarefa no "scheduler" do sistema operativo e foi implementado mediante pequenas adaptações do exemplo dado na especificação, e embora a mesma não atendesse às distinções potencialmente resultantes da implementação numa arquitetura tecnológica alternativa, por exemplo, com máquinas distintas para o Sybase e o MySQL, com efeitos na configuração não apenas dos "path" para o "mysql" e os ficheiros, mas também no formato do comando "LOAD" (LOAD DATA INFILE vs. LOAD DATA LOCAL INFILE), foi possível implementar a batch file especificada (script), apenas com as alterações atinentes à resolução dos problemas acima referidos.

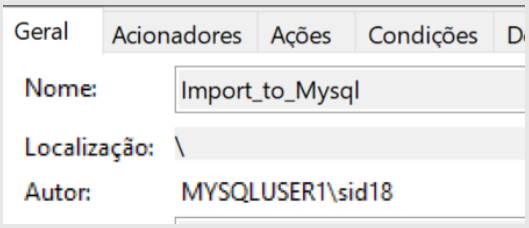
Lista de Eventos (para cada evento assinalar com x em célula correspondente)

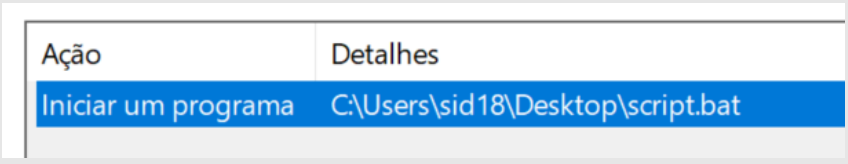
	Implementado de Acordo com Especificado	Implementado mas diferente de Especificado	Não Implementado	Não Especificado (criado de novo)
Export_to_Mysql	X			
Import_to_Mysql		X		

1.4.6.2 Eventos Implementados

```
1. Nome Evento: Export_to_Mysql
// Evento que trata da exportação para o MySql
CREATE EVENT "DBA"."Export_to_Mysql"
SCHEDULE schedule_2_times_a_day
     START TIME '12:00:00'
     EVERY 12 HOURS
HANDLER BEGIN
     ----------------------------
     CALL dba.Export_to_MySQL;
     ----------------------------
     MESSAGE STRING (
     'Evento ' ,
     EVENT_PARAMETER ( 'EventName' ),
        ' despoletado em ',
     CURRENT TIMESTAMP,
     ' devido ao schedule ',
     EVENT_PARAMETER ( 'ScheduleName' )) TO CONSOLE;
END;


2. Nome Evento: Import_to_Mysql
// Código .bat que importa os dados dos txt para o MySql
// Este evento foi configurado manualmente através do
programador de tarefas do Windows
```






```
@ECHO ON

set startTime=%time%

"C:/Program Files/MySQL/MySQLServer/bin/mysql.exe" -u DBA -
psqll -h 127.0.0.1 mysqlfile -e "LOAD DATA LOCAL INFILE
'Z:/VM_SHAREDFOLDER/FILEStoEXPORTandIMPORT3/Log_Cultura.txt
' IGNORE INTO TABLE Log_Cultura fields terminated by ','"

"C:/Program Files/MySQL/MySQLServer/bin/mysql.exe" -u DBA -
```

```
psql1 -h 127.0.0.1 mysqlfile -e "LOAD DATA LOCAL INFILE
'Z:/VM_SHAREDFOLDER/FILEStoEXPORTandIMPORT3/Log_HumidadeTem
peratura.txt' IGNORE INTO TABLE Log_HumidadeTemperatura
fields terminated by ','"

"C:/Program Files/MySQL/MySQLServer/bin/mysql.exe" -u DBA -
psql1 -h 127.0.0.1 mysqlfile -e "LOAD DATA LOCAL INFILE
'Z:/VM_SHAREDFOLDER/FILEStoEXPORTandIMPORT3/Log_Investigado
res.txt' IGNORE INTO TABLE Log_Investigadores fields
terminated by ','"

"C:/Program Files/MySQL/MySQLServer/bin/mysql.exe" -u DBA -
psql1 -h 127.0.0.1 mysqlfile -e "LOAD DATA LOCAL INFILE
'Z:/VM_SHAREDFOLDER/FILEStoEXPORTandIMPORT3/Log_Medicoes.tx
t' IGNORE INTO TABLE Log_Medicoes fields terminated by ','"

"C:/Program Files/MySQL/MySQLServer/bin/mysql.exe" -u DBA -
psql1 -h 127.0.0.1 mysqlfile -e "LOAD DATA LOCAL INFILE
'Z:/VM_SHAREDFOLDER/FILEStoEXPORTandIMPORT3/Log_Variaveis.t
xt' IGNORE INTO TABLE Log_Variaveis fields terminated by
','"

"C:/Program Files/MySQL/MySQLServer/bin/mysql.exe" -u DBA -
psql1 -h 127.0.0.1 mysqlfile -e "LOAD DATA LOCAL INFILE
'Z:/VM_SHAREDFOLDER/FILEStoEXPORTandIMPORT3/Log_VariaveisMe
didas.txt' IGNORE INTO TABLE Log_VariaveisMedidas fields
terminated by ','"

"C:/Program Files/MySQL/MySQLServer/bin/mysql.exe" -u DBA -
psql1 -h 127.0.0.1 mysqlfile -e "LOAD DATA LOCAL INFILE
'Z:/VM_SHAREDFOLDER/FILEStoEXPORTandIMPORT3/Log_Consultas.t
xt' IGNORE INTO TABLE Log_Consultas fields terminated by
','"

echo Start Time: %startTime%
echo Finish Time: %time%

@ECHO OFF
```

1.5 Avaliação Global de especificações da Etapa A

As especificações recebidas eram, quase sempre, **claras, com algumas exceções**. Foram anotados problemas a este nível no que se refere a um detalhe do modelo relacional (utilizador vs. acesso na tabela Log_Consultas - vide parágrafo 1.1.1), bem como na "stored procedure" (SP) "consultCulturas", onde se releva a falta de indicação dos objectivos desta SP tão genérica (vide parágrafo 1.2.1);

Quanto à **coerência**, as especificações recebidas têm coerência interna, mas nem sempre são coerentes com o problema em vista de resolução, tendo sido anotado que presumem que apenas uma coluna poderá ser alterada em cada operação de "update" (vide parágrafo 1.3.1.1) e que não contemplam a necessidade de registo dos autores das operações de "insert", "update" e "delete" (vide parágrafo 1.2.1).

Ao nível da **completude**, foram anotadas insuficiências nas permissões de acesso à base de dados, especialmente para o caso do Sybase, para o qual foram definidas as permissões mais básicas e intuitivas, mas afigura-se que poderá ter faltado alguma atenção para possibilidades extra que poderiam evitar eventuais problemas, como por exemplo a implementação de restrições correspondentes a uma maior segregação de funções, de modo a melhor limitar e controlar as operações sobre os dados, como podemos observar no caso dos administradores com acesso direto sobre os dados provenientes dos sensores, ou os Investigadores não terem restrições de acesso às tabelas adequadas (como anotado no parágrafo 1.2.1). Os SPs de suporte aos utilizadores também não foram definidos de uma forma rigorosa. Faltou bastante informação sobre os mesmos, sendo apenas dados os parâmetros de entrada e saída (também apontado no parágrafo 1.2.1).

Todavia, no que diz respeito ao **rigor**, as mesmas especificações continham alguns presumíveis defeitos, potencialmente geradoras de problemas de funcionamento, designadamente no que diz respeito ao modelo relacional proposto (vide parágrafo 1.1.1), às SPs propostas insertMediçoes, consultVariaveisMedidas e consultCulturas (vide parágrafo 1.2.1).

Globalmente, consideramos que a qualidade da especificação recebida é boa.

Avaliação Global da Qualidade das Especificações recebidas

Avaliação (A,B,C,D,E): D

Utilize a seguinte escala:
A: - 1 – 5 valores B: 6 – 9 valores C: 10 – 13 Valores D: 14 – 17 valores E: 18 – 20 valores

Três principais deficiências de especificação que tiveram impacto mais negativo na qualidade da implementação

O modelo relacional proposto na especificação limita a capacidade de auditoria de dados e presume apenas um atributo alterado e cada operação de atualização.

Na especificação não foi dada importância à protecção de dados e por esse motivo a mesma não foi implementada, reduzindo a qualidade da implementação.

Pouca justificação/explicação sobre funções pretendidas para algumas stored procedures, levando a que fossem assumidos alguns pressupostos que podem ser dissonantes do objetivo final, pretendido na especificação.

Resumo de Avaliações de Qualidade Anteriores (para cada linha assinalar com x em célula correspondente)

	Fraco	Razoável	Bom	Muito Bom
BD Sybase		X		
Triggers Log			X	
SP Log		X		
Utilizadores Log		X		
BD Mysql		X		
Forma Migração			X	
Triggers Migração	n/a			
SP Migração				X
Eventos Migração			X	
Utilizadores Migração				X

2 Etapa C (Especificação e Implementação do Próprio Grupo)

2.1 Especificação do Esquema relacional da base de Dados Sybase

Para o esquema relacional da base de dados *Sybase* optou-se pela representação da Figura 5 por ser mais simples a perceção das chaves primárias e estrangeiras de cada tabela.

De notar que as tabelas de log foram criadas no lado do *Sybase* (à exceção da tabela HumidadeTemperatura, que já funciona como tal) e serão depois replicadas do lado do *MySql*, porque caso contrário, sempre que houvesse uma alteração às tabelas do *Sybase* tínhamos de enviar imediatamente a informação para o *MySql*, correndo o risco de a ligação ODBC não estar ativa e a informação ser perdida. Desta forma garantimos que todas as alterações e acessos às tabelas são registadas.

Todas as tabelas de log têm um campo Id *auto-increment* para que cada operação sobre a tabela respetiva seja guardada de forma sequencial. Essa sequência também permitirá tornar a migração mais eficiente pois quando for solicitada a migração da informação para a base de dados MySql, apenas as entradas com Id superior ao Id anteriormente transferido serão migradas.

Alterações às tabelas fornecidas:

1. Foi acrescentada a coluna "idInvestigador" à tabela "Investigador" para que o email não seja a chave primária da mesma.

2. Foi acrescentado em todas as tabelas um atributo denominado "deleted", do tipo booleano, que faz com que:
   - Passe a haver dois tipos de operações "*Delete"*: um "*Soft Delete"* em que apenas é alterado o atributo *Deleted* e o registo permanece na base de dados (correspondendo na realidade a um "*Update*"), e um "*Hard Delete"* que elimina definitivamente uma linha da base de dados e não pode ser facilmente revertido (haveria ainda, em princípio, a possibilidade de recorrer aos "logs").
   - Apenas o Administrador pode fazer *Hard Delete*.
   - O investigador só poderá fazer um *Soft Delete* e este terá de ser feito através de uma "*Stored Procedure"*.
   - A "*Stored Procedure"* para execução de comandos "*Select"* e as "*Views"* têm que ter em conta a necessidade de subtrair os registos "*deleted"* aos resultados retornados.
   - Seja possível reverter um "Soft *Delete"* enganado, através do utilizador Administrador.

3. Nas tabelas "HumidadeTemperatura" e "Medicoes", os campos "data da medição" e "hora da medição" foram unidos num campo único de Data e Hora (timestamp). Dessa forma é possível uniformizar os campos de data e hora com as restantes tabelas de "logs", visto a tabela "HumidadeTemperatura" funcionar também como "log". No caso da tabela "Medicoes", apesar de esta ter uma tabela de "log" associada, usando um timestamp uniformiza todos os atributos de data e hora na base de dados.

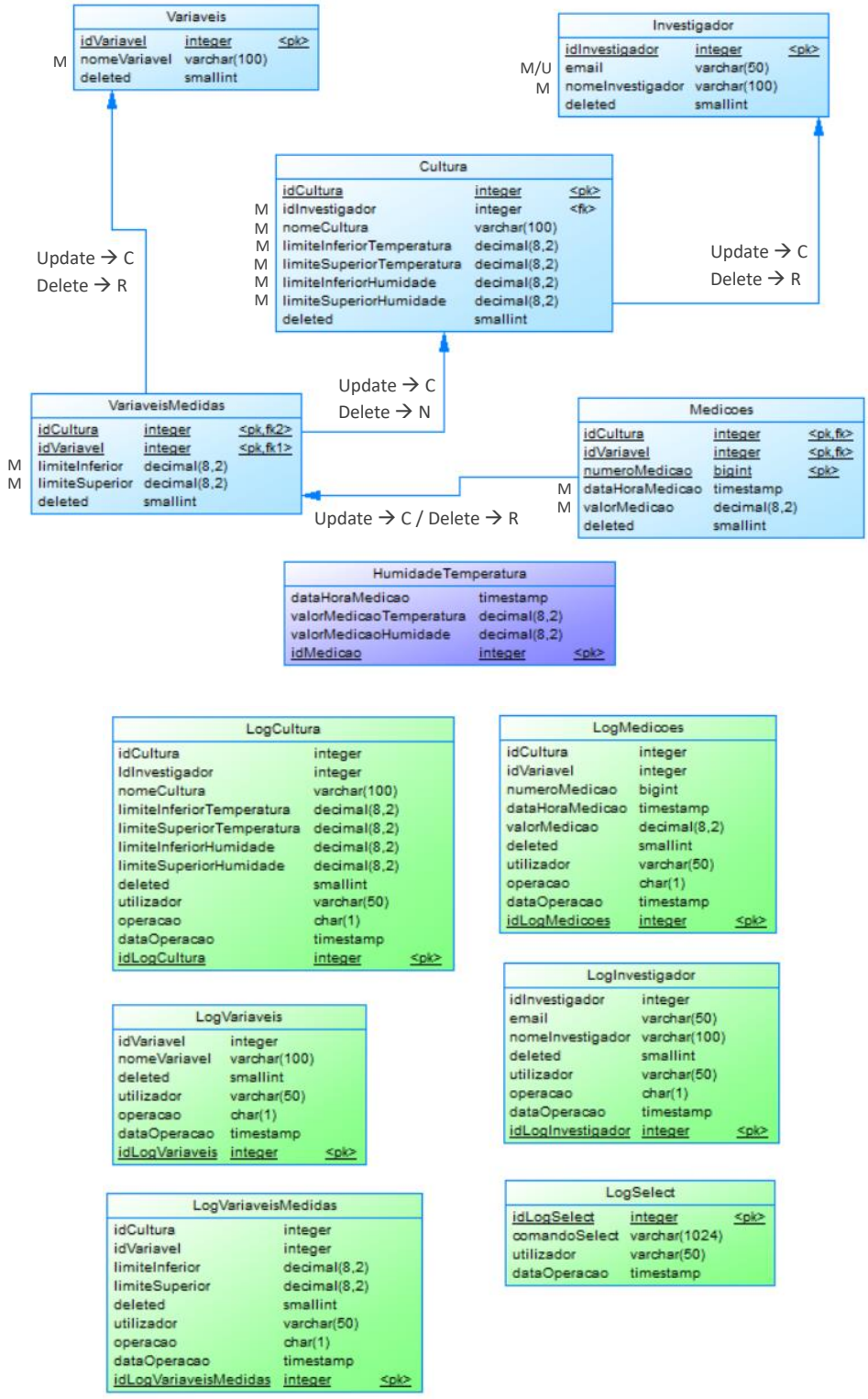


Figura 5 - Diagrama físico da BD Sybase

Legenda:

- U – Unique
- M – Mandatory
- C – Cascade
- R – Restrict
- N – Set Null

2.2 Especificação de Utilizadores

Na implementação da base de dados Sybase, irão existir 4 tipos de utilizadores:

- Um administrador principal;
- Um grupo de administradores;
- Um grupo de investigadores;
- Um utilizador MongoDB;

Administrador principal

Este utilizador somente pode pertencer a uma pessoa na organização, correspondendo a um "SuperAdministrador". Não confundir com o utilizador DBA, pois esse utilizador pertence à equipa de desenvolvimento e não possui quaisquer restrições. Este utilizador irá fazer parte do grupo dos Administradores, herdando todas as outras permissões e autoridades do grupo. No entanto, será responsável pela criação de outros administradores, sendo esta a sua única função adicional ao resto dos administradores.

Grupo de Administradores

Os administradores, após o administrador principal, são os utilizadores que terão maior controlo sobre operações na base de dados, podendo ver e alterar qualquer tabela, à exceção das de logs. Todos os administradores, incluindo o "SuperAdministrador", não terão qualquer permissão para criar ou alterar a estrutura de tabelas, nem alterar a estrutura da base de dados, sendo estas funções da responsabilidade do utilizador DBA. As suas permissões limitam-se apenas a controlo de dados.

Notas:

(a) Os *inserts*, *deletes* e *updates* na tabela dos Investigadores, terão de ser substituídos por *stored procedures* para a criação de logins de acesso à base de dados. O nome dos utilizadores na base de dados será o email definido no atributo da tabela Investigadores.

Grupo de Investigadores

Os Investigadores serão o grupo mais complexo da base de dados. Terão acessos variados por tabela, como refere no use cases, mas também outras permissões que consideramos importantes ter. O principal objetivo das restrições implementadas foi garantir que os investigadores, apesar de poderem consultar dados variados, somente teriam acesso à informação das suas culturas.

Notas:

(b) Os s*elects* (feitos a partir de Store Procedures) executados pelos investigadores às tabelas Investigador, Cultura, Variaveis, VariaveisMedidas e Medicoes serão realizados através de

Views, para garantirmos que cada investigador apenas consulta informação pertencente ao investigador em questão.

(c) Existirá um *trigger* que antecede às operações *Insert*, *Delete* e *Update* da tabela Medicoes com o objetivo de, sempre que seja feita uma alteração por parte do investigador, verificar se está a referir uma cultura pertencente ao mesmo.

Utilizador MongoDB

Este utilizador será utilizado pela base de dados MongoDB, para enviar dados brutos dos sensores para a tabela HumidadeTemperatura, tendo apenas uma permissão de *insert* nessa mesma tabela.

Permissões	Administrador Principal	Administradores	Investigadores	MongoDB
Tabelas				
Investigador	U, I, SD, HD	U, I, SD, HD (a)	-	-
Cultura	U, I, SD, HD	U, I, SD, HD	-	-
Variaveis	U, I, SD, HD	U, I, SD, HD	-	-
VariaveisMedidas	U, I, SD, HD	U, I, SD, HD	-	-
Medicoes	U, I, SD, HD	U, I, SD, HD	U, I, SD (c)	-
HumidadeTemperatura	-	-	-	I
Tabelas Logs	-	-	-	-
Views	S	S	S (b)	-
Store Procedures				
SoftDelete Investigador	C	C	-	-
SoftDelete Cultura	C	C	-	-
SoftDelete Variaveis	C	C	-	-
SoftDelete VariaveisMedidas	C	C	-	-
SoftDelete Medicoes	C	C	C	-
Registo Log Selects	C	C	C	-
Create User Investigador	C	C	-	-
Alter User Investigador	C	C	-	-
Drop User Investigador	C	C	-	-

Tabela 1 - Lista de permissões por grupos de utilizadores

Legenda:

- S – Select
- U – Update
- I – Insert
- SD – Soft Delete (através de um Store Procedure)
- HD – Hard Delete
- C – Call Store Procedure

2.3 Especificação de Gestão de Logs

Relativamente à informação a guardar, optámos por apenas guardar a informação nova inserida, juntamente com os dados que permaneceram inalterados, não havendo lugar a campos de Old e New e evitando redundância de informação.

Inserindo toda a informação da respetiva entrada na tabela, salvaguardamos os casos em que é feita uma pesquisa por parte do Auditor dentro de um período específico que não englobasse toda a informação necessária. Um exemplo seria não saber a que Cultura pertencia um registo update que está a observar. Outra vantagem, é o facto de não ser preciso consultar mais tabelas para completar informação.

Esta abordagem tem ainda a vantagem de não ser necessário criar, em cada tabela de log, o dobro (aproximadamente) dos atributos pois seria necessário um campo Old e um New para cada atributo que se pudesse editar.

Esta especificação tem a desvantagem de tornar mais difícil ao Auditor perceber que dados foram alterados. Todavia, de forma a facilitar a pesquisa pela base de dados MySql, por parte do Auditor, serão criadas Views em que é possível visualizar na mesma linha os valores antigos e novos, para que seja facilitada a visualização das alterações efetuadas em cada UPDATE.

2.3.1 Triggers de suporte à gestão de logs

A Tabela 2 apresenta os triggers criados para o preenchimento das tabelas de log, sendo que cada operação (DELETE, INSERT e UPDATE) sobre cada tabela leva à inserção de uma entrada na respetiva tabela de log (exceto a tabela HumidadeTemperatura que já é uma tabela log).

Nome Trigger	Tabela	Tipo de Operação	After / Before	Notas
tr_del_Cultura	Cultura	Delete	After	
tr_ins_Cultura	Cultura	Insert	After	
tr_upd_Cultura	Cultura	Update	After	
tr_del_Investigador	Investigador	Delete	After	
tr_ins_Investigador	Investigador	Insert	After	
tr_upd_Investigador	Investigador	Update	After	
tr_del_Medicoes	Medicoes	Delete	After	
tr_ins_ Medicoes	Medicoes	Insert	After	
tr_upd_ Medicoes	Medicoes	Update	After	
tr_del_Variaveis	Variaveis	Delete	After	
tr_ins_ Variaveis	Variaveis	Insert	After	
tr_upd_ Variaveis	Variaveis	Update	After	
tr_del_VariaveisMedidas	VariaveisMedidas	Delete	After	
tr_ins_ VariaveisMedidas	VariaveisMedidas	Insert	After	
tr_upd_ VariaveisMedidas	VariaveisMedidas	Update	After	

Tabela 2 - Listagem de triggers para registo de logs

Os Comandos “SELECT” efetuados pelos utilizadores não serão tratados por “Triggers”, mas antes por “Stored Procedures” construídas para o efeito.

Os “Triggers” são ativados depois (AFTER) da operação ter ocorrido, para no caso da operação falhar, não sejam lançados os “Triggers”, nem sejam executadas as ações nestes contidas.

2.3.2 Stored Procedures de suporte à gestão de logs

Por forma a registar todas as operações de select na base de dados, optou-se por criar um Stored Procedure (visto não existir um trigger previsto para essa operação). A forma de construção do SP prevê que este funcione independentemente do comando select efetuado e das tabelas chamadas (não é necessário este fazer controlo de permissões pois a gestão de utilizadores encarrega-se disso).

Nome SP	**Parâmetro Entrada**	**Parâmetro Saída**	**BD**	**Nota**
sp_selectLogs	String do Comando SQL	Resultado da Consulta do Comando SQL	Sybase	

O sp_selectLogs tem por objetivo servir de porta de entrada para todas as consultas à base de dados por parte dos utilizadores. Este, para além de permitir a consulta das tabelas e views, deverá também gravar o comando select efetuado e convertê-lo de forma a poder ser utilizado pelo auditor no lado do MySQL, retornando os mesmos dados observados na altura por quem chamou a SP.

Este esquema de registo apesar da complexidade de implementação, possui enormes vantagens, quer em termos de memória, quer em termos de informação a reter, ou seja:

1. Como o auditor recebe o comando SQL efetuado pelo utilizador, este pode optar por analisar o comando em si ou por repetir a consulta, analisando os resultados retornados. Dessa forma ele ganha maior confiança e conhecimento através da análise dos dois métodos.
2. Como o comando é guardado em forma de string, o tamanho do registo é independente da complexidade da consulta. Dessa forma, o SP vai funcionar de igual forma quer se trate de um comando simples a uma tabela, quer se trate de um comando mais complexo que inclua mais tabelas e até filtros.
3. Visto os registos de retorno da consulta não serem guardados, mas sim o comando que os chamou, torna-se mais seguro para a memória da base de dados. Na primeira situação, caso se optasse por gravar os registos da consulta, se um investigador tivesse intenções de bloquear a base de dados, simplesmente fazia consultas sequenciais com enorme quantidade de registos até ultrapassar a memória disponível, colocando em risco o funcionamento da base de dados.

2.4 Organização de Views, outros Triggers e Stored Procedures

2.4.1 Criação de Views para controlo de acesso dos investigadores

Para garantir um controlo eficaz da informação pertencente a cada investigador, optou-se por criar uma view por cada tabela (tabela essa que possa ter registos de vários investigadores). Nesse sentido, foram desenvolvidas cinco views para as tabelas: Investigador, Cultura, Variaveis, VariaveisMedidas e Medicoes. Cada view deverá devolver a informação da respetiva tabela respeitante somente ao investigador que se encontra ligado. O objetivo das views será impedir que cada investigador consulte informação de outros investigadores (política de privacidade).

A listagem das views encontra-se na Tabela 3:

Nome View	Tabela respeitante	Notas
v_InvestigadorPorInvestigador	Investigador	
v_CulturaPorInvestigador	Cultura	
v_VariaveisPorInvestigador	Variaveis	
v_VariaveisMedidasPorInvestigador	VariaveisMedidas	
v_MedicoesPorInvestigador	Medicoes	

Tabela 3 - Listagem de views

2.4.2 Criação de Trigger para controlo de alterações dos investigadores

À semelhança das views anteriores, também é necessário implementar uma trigger que evite que os investigadores insiram ou alterem informações da tabela Medicoes que não lhes pertençam. Essa situação é possível de acontecer devido ao facto dos mesmos terem permissões de Insert e Update (inclusive o SoftDelete) nessa tabela.

As características do trigger são as seguintes:

Nome Trigger	Tabela	Tipo de Operação	After / Before	Notas
tr_beforeInsUpdMedicoes	Medicoes	Insert, Update	Before	

Tabela 4 – Identificação do trigger de controlo das medições

2.4.3 Stored Procedures para criação e eliminação de investigadores

Uma das funções do grupo de administradores é a gestão de investigadores. Na base de dados eles têm permissão para inserir, atualizar e eliminar investigadores, por isso, torna-se necessário que sejam criados ou eliminados utilizadores do grupo de Investigadores sempre que isso aconteça. Foram criados três Stored Procedures que têm por objetivo criar, atualizar ou eliminar um utilizador no grupo dos Investigadores e, após, acrescentar, alterar ou eliminar o registo na tabela Investigadores.

As características do Stores Procedures são as seguintes:

Nome SP	Parâmetro Entrada	Parâmetro Saída	BD	Nota
sp_createInvestigador	Email e password do investigador	NULL	Sybase	
sp_alterInvestigador	Id do investigador, novo email e novo nome	NULL	Sybase	
sp_dropInvestigador	Email do investigador	NULL	Sybase	

Tabela 5 – Identificação do trigger de controlo das medições

2.4.4 Stored Procedures para updates SoftDelete

Nesta base de dados, optou-se por implementar um sistema de deletes que permite garantir uma melhor gestão de informação e fiabilidade dos dados. Existem dois tipos de delete dos dados, o HardDelete que vai de encontro ao procedimento normal de DELETE, sendo essa informação irrecuperável, e o SoftDelete que consiste na alteração da flag *deleted* nos registos das tabelas. Para que esse sistema funcione corretamente é necessário criar Stored Procedures que "eliminam" os registos através do método de SoftDelete.

As características do Stores Procedures são as seguintes:

Nome SP	Parâmetro Entrada	Parâmetro Saída	BD	Nota
sp_softDeleteInvestigador	Id do investigador	NULL	Sybase	
sp_softDeleteCultura	Id da cultura	NULL	Sybase	
sp_softDeleteVariaveis	Id da variável	NULL	Sybase	
sp_softDeleteVariaveisMedidas	Id da variável e da cultura	NULL	Sybase	
sp_softDeleteMedicoes	Id da variável, cultura e numero da medição	NULL	Sybase	

Tabela 6 – Identificação do trigger de controlo das medições

Nota:

É importante garantir que um registo que tenha a flag *deleted* ativa não apareça nos registos de consulta dos investigadores.

2.5 Avaliação da especificação do próprio grupo Gestão de Logs

Qualidade (Fraca, Razoável, Boa ou Muito Boa): Muito Boa

A gestão de logs através do sistema especificado torna-se bastante robusta e escalável. Apesar da complexidade de funções e comandos, é uma mais-valia a flexibilidade conciliada à segurança e privacidade dos dados implementada na base de dados.
Através dos triggers e stored procedures implementados, todos os registos são corretamente registados nos logs (quer os comandos INSERT, UPDATE e DELETE, quer os comandos SELECT) ao mesmo tempo que a informação é corretamente filtrada de acordo com as permissões dos vários utilizadores.
A utilização de um procedimento genérico para registo das consultas nas tabelas torna a base de dados mais escalável e robusta a comandos mais complexos vindos dos utilizadores.

2.6 Implementação Gestão de Logs

2.6.1 Utilizadores implementados

Permissões	Administrador Principal	Administradores	Investigadores	MongoDB
Tabelas				
Investigador	U, I, SD, HD	U, I, SD, HD (a)	-	-
Cultura	U, I, SD, HD	U, I, SD, HD	-	-
Variaveis	U, I, SD, HD	U, I, SD, HD	-	-
VariaveisMedidas	U, I, SD, HD	U, I, SD, HD	-	-
Medicoes	U, I, SD, HD	U, I, SD, HD	U, I, SD (c)	-
HumidadeTemperatura	-	-	-	I
Tabelas Logs	-	-	-	-
Views	S	S	S (b)	-
Store Procedures				
SoftDelete Investigador	C	C	-	-
SoftDelete Cultura	C	C	-	-
SoftDelete Variaveis	C	C	-	-
SoftDelete VariaveisMedidas	C	C	-	-
SoftDelete Medicoes	C	C	C	-
Registo Log Selects	C	C	C	-
Create User Investigador	C	C	-	-
Alter User Investigador	C	C	-	-
Drop User Investigador	C	C	-	-

2.6.2 Lista de Triggers

Lista de Triggers (para cada trigger assinalar com x em célula correspondente)

	Implementado de Acordo com Especificado	Implementado mas diferente de Especificado	Não Implementado	Não Especificado (criado de novo)
tr_del_Cultura	X			
tr_ins_Cultura	X			
tr_upd_Cultura	X			
tr_del_Investigador	X			
tr_ins_Investigador	X			
tr_upd_Investigador	X			
tr_del_Medicoes	X			
tr_ins_ Medicoes	X			
tr_upd_ Medicoes	X			
tr_del_Variaveis	X			
tr_ins_ Variaveis	X			
tr_upd_ Variaveis	X			
tr_del_VariaveisMedidas	X			
tr_ins_ VariaveisMedidas	X			
tr_upd_ VariaveisMedidas	X			
tr_beforeInsUpd Medicoes	X			

2.6.3 Triggers Implementados

```
1. Nome Trigger: tr_del_Cultura

// Inserção na tabela logCultura do registo eliminado
create trigger tr_after_del_Cultura after delete order 1 on
Cultura
referencing old as old_del for each row
begin
   declare user_defined_exception exception for SQLSTATE
     '99999';
   declare found integer;
   INSERT INTO LogCultura (idCultura,
     idInvestigador,
     nomeCultura,
     limiteInferiorTemperatura,
     limiteSuperiorTemperatura,
     limiteInferiorHumidade,
     limiteSuperiorHumidade,
     deleted,
     utilizador,
     operacao,
     dataOperacao)
   VALUES (old_del.idCultura,
     old_del.idInvestigador,
     old_del.nomeCultura,
     old_del.limiteInferiorTemperatura,
     old_del.limiteSuperiorTemperatura,
     old_del.limiteInferiorHumidade,
     old_del.limiteSuperiorHumidade,
     old_del.deleted,
     user_name(),
     'D',
     now() );
end;
```

```
2. Nome Trigger: tr_ins_Cultura

// Inserção na tabela logCultura do registo inserido
create trigger tr_ins_Cultura after insert order 1 on
Cultura
referencing new as new_ins for each row
begin
  INSERT INTO LogCultura (idCultura,
    idInvestigador,
    nomeCultura,
    limiteInferiorTemperatura,
    limiteSuperiorTemperatura,
    limiteInferiorHumidade,
    limiteSuperiorHumidade,
    deleted,
    utilizador,
    operacao,
    dataOperacao)
  VALUES (new_ins.idCultura,
    new_ins.idInvestigador,
    new_ins.nomeCultura,
    new_ins.limiteInferiorTemperatura,
    new_ins.limiteSuperiorTemperatura,
    new_ins.limiteInferiorHumidade,
    new_ins.limiteSuperiorHumidade,
    new_ins.deleted,
    user_name(),
    'I',
    now());
end;


3. Nome Trigger: tr_upd_Cultura

// Inserção na tabela logCultura do registo modificado
create trigger tr_after_upd_Cultura after update of
idCultura, idInvestigador order 1 on Cultura
referencing new as new_upd old as old_upd for each row
begin
  declare found integer;
  INSERT INTO LogCultura (idCultura,
    idInvestigador,
    nomeCultura,
    limiteInferiorTemperatura,
    limiteSuperiorTemperatura,
    limiteInferiorHumidade,
    limiteSuperiorHumidade,
    deleted,
    utilizador,
    operacao,
```

```
    dataOperacao)
  VALUES (new_upd.idCultura,
    new_upd.idInvestigador,
    new_upd.nomeCultura,
    new_upd.limiteInferiorTemperatura,
    new_upd.limiteSuperiorTemperatura,
    new_upd.limiteInferiorHumidade,
    new_upd.limiteSuperiorHumidade,
    new_upd.deleted,
    user_name(),
    'U',
    now());
end;


4. Nome Trigger: tr_del_Investigador

// Inserção na tabela logInvestigador do registo eliminado
create trigger tr_del_Investigador after delete order 1 on
Investigador
referencing old as old_del for each row
begin
  declare found integer;
  INSERT INTO LogInvestigador (idInvestigador,
    email,
    nomeInvestigador,
    deleted,
    utilizador,
    operacao,
    dataOperacao)
  VALUES (old_del.idInvestigador,
    old_del.email,
    old_del.nomeInvestigador,
    old_del.deleted,
    user_name(),
    'D',
    now());
end;


5. Nome Trigger: tr_ins_Investigador

// Inserção na tabela logInvestigador do registo inserido
create trigger tr_ins_Investigador after insert order 1 on
Investigador
referencing new as new_ins for each row
begin
  INSERT INTO LogInvestigador (idInvestigador,
                               email,
                               nomeInvestigador,
                               deleted,
```

```
                                    utilizador,
                                    operacao,
                                    dataOperacao)
    VALUES                         (new_ins.idInvestigador,
                                    new_ins.email,
                                    new_ins.nomeInvestigador,
                                    new_ins.deleted,
                                    user_name(),
                                    'I',
                                    now() );

end;


6. Nome Trigger: tr_upd_Investigador

// Inserção na tabela logInvestigador do registo modificado
create trigger tr_after_upd_Investigadorr after update of
idInvestigador order 1 on Investigador
referencing new as new_upd old as old_upd for each row
begin
  declare found integer;
  insert into LogInvestigador (idInvestigador,
    email,
    nomeInvestigador,
    utilizador,
    deleted,
    operacao,
    dataOperacao)
  VALUES (new_upd.idInvestigador,
    new_upd.email,
    new_upd.nomeInvestigador,
    new_upd.deleted,
    user_name(),
    'U',
    now());
end;


7. Nome Trigger: tr_del_Medicoes

// Inserção na tabela logMedicoes do registo eliminado
create trigger tr_del_Medicoes after delete order 1 on
Medicoes
referencing old as old_del for each row
begin
  declare found integer;
  INSERT INTO LogMedicoes (idCultura,
    idVariavel,
    numeroMedicao,
    dataHoraMedicao,
```

```
    valorMedicao,
    deleted,
    utilizador,
    operacao,
    dataOperacao)
  VALUES (old_del.idCultura,
    old_del.idVariavel,
    old_del.numeroMedicao,
    old_del.dataHoraMedicao,
    old_del.valorMedicao,
    old_del.deleted,
    user_name(),
    'D',
    now() );
end;


8. Nome Trigger: tr_ins_Medicoes

// Inserção na tabela logMedicoes do registo inserido
create trigger tr_after_ins_Medicoes after insert order 1
on Medicoes
referencing new as new_ins for each row
begin
  INSERT INTO LogMedicoes (idCultura,
    idVariavel,
    numeroMedicao,
    dataHoraMedicao,
    valorMedicao,
    deleted,
    utilizador,
    operacao,
    dataOperacao)
  VALUES (new_ins.idCultura,
    new_ins.idVariavel,
    new_ins.numeroMedicao,
    new_ins.dataHoraMedicao,
    new_ins.valorMedicao,
    new_ins.deleted,
    user_name(),
    'I',
    now());
end;
```

```
9. Nome Trigger: tr_upd_Medicoes

// Inserção na tabela logMedicoes do registo modificado
create trigger tr_upd_Medicoes after update of idCultura,
idVariavel, numeroMedicao
order 1 on Medicoes
referencing new as new_upd old as old_upd for each row
begin
  declare found integer;
  INSERT INTO LogMedicoes (idCultura,
    idVariavel,
    numeroMedicao,
    dataHoraMedicao,
    deleted,
    utilizador,
    operacao,
    dataOperacao)
  VALUES (new_upd.idCultura,
    new_upd.idVariavel,
    new_upd.numeroMedicao,
    new_upd.dataHoraMedicao,
    new_upd.deleted,
    user_name(),
    'U',
    now());
end;


10. Nome Trigger: tr_del_Variaveis

// Inserção na tabela logVariaveis do registo eliminado
create trigger tr_del_Variaveis after delete order 1 on
Variaveis
referencing old as old_del for each row
begin
  declare found integer;
  INSERT INTO LogVariaveis (idVariavel,
    nomeVariavel,
    deleted,
    utilizador,
    operacao,
    dataOperacao)
  VALUES (old_del.idVariavel,
    old_del.nomeVariavel,
    old_del.deleted,
    user_name(),
    'D',
    now());
end;
```

```
11. Nome Trigger: tr_ins_Variaveis

// Inserção na tabela logVariaveis do registo inserido
create trigger tr_ins_Variaveis after insert order 1 on
Variaveis
referencing new as new_ins for each row
begin
  INSERT INTO LogVariaveis (idVariavel,
    nomeVariavel,
    deleted,
    utilizador,
    operacao,
    dataOperacao)
  VALUES (new_ins.idVariavel,
    new_ins.nomeVariavel,
    new_ins.deleted,
    user_name(),
    'I',
    now());
end;


12. Nome Trigger: tr_upd_Variaveis

// Inserção na tabela logVariaveis do registo modificado
create trigger tr_upd_Variaveis after update of idVariavel
order 1 on Variaveis
referencing new as new_upd old as old_upd for each row
begin
  declare found integer;
  INSERT INTO LogVariaveis (idVariavel,
    nomeVariavel,
    deleted,
    utilizador,
    operacao,
    dataOperacao)
  VALUES (new_upd.idVariavel,
    new_upd.nomeVariavel,
    new_upd.deleted,
    user_name(),
    'U',
    now() );
end;
```

```
13. Nome Trigger: tr_del_VariaveisMedidas

// Inserção na tabela logVariaveisMedidas do eliminado
create trigger tr_del_VariaveisMedidas after delete order 1
on VariaveisMedidas
referencing old as old_del for each row
begin
  declare found integer;
  INSERT INTO LogVariaveisMedidas (idCultura,
    idVariavel,
    limiteInferior,
    limiteSuperior,
    deleted,
    utilizador,
    operacao,
    dataOperacao)
  VALUES (old_del.idCultura,
    old_del.idVariavel,
    old_del.limiteInferior,
    old_del.limiteSuperior,
    old_del.deleted,
    user_name(),
    'D',
    now());
end;


14. Nome Trigger: tr_ins_VariaveisMedidas

// Inserção na tabela logVariaveisMedidas do inserido
create trigger tr_ins_VariaveisMedidas after insert order 1
on VariaveisMedidas
referencing new as new_ins for each row
begin
  INSERT INTO LogVariaveisMedidas (idCultura,
    idVariavel,
    limiteInferior,
    limiteSuperior,
    deleted,
    utilizador,
    operacao,
    dataOperacao)
  VALUES (new_ins.idCultura,
    new_ins.idVariavel,
    new_ins.limiteInferior,
    new_ins.limiteSuperior,
    new_ins.deleted,
    user_name(),
    'I',
    now());
end;
```

```
15. Nome Trigger: tr_upd_VariaveisMedidas
// Inserção na tabela logVariaveisMedidas do modificado
create trigger tr_upd_VariaveisMedidas after update of
idCultura, idVariavel
order 1 on VariaveisMedidas
referencing new as new_upd old as old_upd for each row
begin
  declare found integer;
  INSERT INTO LogVariaveisMedidas (idCultura,
    idVariavel,
    limiteInferior,
    limiteSuperior,
    deleted,
    utilizador,
    operacao,
    dataOperacao)
  VALUES (new_upd.idCultura,
    new_upd.idVariavel,
    new_upd.limiteInferior,
    new_upd.limiteSuperior,
    new_upd.deleted,
    user_name(),
    'U',
    now());
end;


16. Nome Trigger: tr_beforeInsUpdMedicoes
// Verifica se o investigador ter permissões para alterar
ou inserir os valores da tabela Medicoes
create trigger tr_beforeInsUpdMedicoes before insert,
update order 1 on Medicoes
referencing new as new_medicao
for each row
begin
  DECLARE emailInvestigador VARCHAR(50);
  DECLARE utilizadorLigado VARCHAR(50);

  SELECT FIRST Investigador.email INTO emailInvestigador
  FROM Cultura, Medicoes, Investigador
  WHERE new_medicao.idCultura = Cultura.idCultura
  AND Cultura.idInvestigador = Investigador.idInvestigador;

  SELECT FIRST Investigador.email INTO utilizadorLigado
  FROM Investigador WHERE Investigador.email = user_name();

  IF emailInvestigador <> user_name() AND utilizadorLigado
  IS NOT NULL THEN RAISERROR 23000 'Nao pode alterar
  medicoes de culturas de outros investigadores'
  END IF;
END;
```

2.6.4 Lista de Stored Procedures

Lista de SP (para cada SP assinalar com x em célula correspondente)

	Implementado de Acordo com Especificado	Implementado mas diferente de Especificado	Não Implementado	Não Especificado (criado de novo)
sp_selectLogs	X			
sp_createInvestigador	X			
sp_alterInvestigador	X			
sp_dropInvestigador	X			
sp_softDeleteInvestigador	X			
sp_softDeleteCultura	X			
sp_softDeleteVariaveis	X			
sp_softDeleteVariaveisMedidas	X			
sp_softDeleteMedicoes	X			

2.6.5 Stored Procedures Implementados

```
1. Nome SP: sp_selectLogs
// SP invocado ao invés de selects às tabelas e views. Este
retorna o SELECT efetuado e armazena na tabela logSelects o
comando efetuado devidamente adaptado ao auditor

create procedure "DBA"."sp_insLogSelects"(
    IN arg_command VARCHAR(500) DEFAULT '')
BEGIN
  DECLARE fix_command VARCHAR(500);
  EXECUTE IMMEDIATE WITH RESULT SET ON arg_command;

  /* Substituição das views pelas tabelas LOG */
  SELECT (replace(replace(replace(replace
    (replace(arg_command,
    'DBA.CulturaPorInvestigador', 'LogCultura'),
    'DBA.InvestigadorPorInvestigador', 'LogInvestigador'),
    'DBA.VariaveisPorInvestigador', 'LogVariaveis'),
    'DBA.MedicoesPorInvestigador', 'LogMedicoes'),
    'DBA.HumidadeTemperatura', 'LogHumidadeTemperatura'))
  INTO fix_command;

  /* Aplicação do fator data de operação anterior */
  IF (charindex('WHERE', fix_command) > 0)
  THEN SELECT replace(fix_command, 'WHERE ',
  'WHERE dataOperacao <= ' + dateformat(CURRENT TIMESTAMP,
  'YYYY-MM-DD') + ' AND ') INTO fix_command;

  ELSE IF (charindex('ORDER BY', fix_command) > 0)
  THEN SELECT replace(fix_command, 'ORDER BY ',
  'WHERE dataOperacao <= ' + dateformat(CURRENT TIMESTAMP,
  'YYYY-MM-DD') + ' ORDER BY ') INTO fix_command;

  ELSE IF (charindex('INNER JOIN', fix_command) > 0)
  THEN SELECT replace(fix_command, 'INNER JOIN ',
  'WHERE dataOperacao <= ' + dateformat(CURRENT TIMESTAMP,
  'YYYY-MM-DD') + ' INNER JOIN ') INTO fix_command;

  ELSE IF (charindex('LEFT JOIN', fix_command) > 0)
  THEN SELECT replace(fix_command, 'LEFT JOIN ',
  'WHERE dataOperacao <= ' + dateformat(CURRENT TIMESTAMP,
  'YYYY-MM-DD') + ' LEFT JOIN ') INTO fix_command;

  ELSE IF (charindex('RIGHT JOIN', fix_command) > 0)
  THEN SELECT replace(fix_command, 'RIGHT JOIN ',
  'WHERE dataOperacao <= ' + dateformat(CURRENT TIMESTAMP,
  'YYYY-MM-DD') + ' RIGHT JOIN ') INTO fix_command;
```

```
  ELSE IF (charindex('JOIN', fix_command) > 0)
  THEN SELECT replace(fix_command, 'JOIN ',
  'WHERE dataOperacao <= ' + dateformat(CURRENT TIMESTAMP,
  'YYYY-MM-DD') + ' JOIN ') INTO fix_command;

  ELSE SELECT (fix_command + ' WHERE dataOperacao <= ' +
  dateformat(CURRENT TIMESTAMP, 'YYYY-MM-DD'))
  INTO fix_command;

  END IF; END IF; END IF; END IF; END IF; END IF;

  /* Adaptação da data de operação à tabela existente */
  IF (charindex('LogCultura', fix_command) > 0)
  THEN SELECT replace(fix_command, 'dataOperacao',
  'LogCultura.dataOperacao') INTO fix_command;

  ELSE IF (charindex('LogInvestigador', fix_command) > 0)
  THEN SELECT replace(fix_command, 'dataOperacao',
  'LogInvestigador.dataOperacao') INTO fix_command;

  ELSE IF (charindex('LogVariaveis', fix_command) > 0)
  THEN SELECT replace(fix_command, 'dataOperacao',
  'LogVariaveis.dataOperacao') INTO fix_command;

  ELSE IF (charindex('LogMedicoes', fix_command) > 0)
  THEN SELECT replace(fix_command, 'dataOperacao',
  'LogMedicoes.dataOperacao') INTO fix_command;

  ELSE IF (charindex('LogHumidadeTemperatura',
  fix_command) > 0)
  THEN SELECT replace(fix_command, 'dataOperacao',
  'LogHumidadeTemperatura.dataOperacao') INTO fix_command;

  END IF; END IF; END IF; END IF; END IF;

  /* Inserção do comando na tabela logSelect */
  INSERT INTO LogSelect (comandoSelect,
    utilizador, dataOperacao)
  VALUES (fix_command, user_name(), CURRENT TIMESTAMP);
END


2. Nome SP: sp_createInvestigador
//  Procedimento que cria um utilizador "investigador" e
adiciona-o na tabela Investigadores
CREATE PROCEDURE sp_createInvestigador(IN email VARCHAR(50)
DEFAULT '', IN pass VARCHAR(50) DEFAULT '')
BEGIN
  IF EXISTS (
    SELECT * FROM dbo.sysusers
```

```
    where dbo.sysusers.name = 'email') THEN DROP USER email
  END IF;

  CREATE USER email IDENTIFIED BY pass;
  IF EXISTS (
    SELECT * FROM dbo.sysusers
    where dbo.sysusers.name = 'Investigadores') THEN
    GRANT MEMBERSHIP IN GROUP "Investigadores" TO email
  END IF;

  IF NOT EXISTS (
    SELECT * FROM DBA.Investigador
    where DBA.Investigador.email = 'email') THEN
    INSERT INTO DBA.Investigador (
      email, nomeInvestigador, deleted)
    VALUES ('email', 'email', 0)
  END IF;
END;


3. Nome SP: sp_alterInvestigador
// Procedimento que altera os dados de um investigador
CREATE PROCEDURE sp_alterInvestigador (
  IN INVID INTEGER, EMAIL VARCHAR(50) DEFAULT '',
  USERNAME VARCHAR(100) DEFAULT '')
BEGIN
  DECLARE @old_email varchar(15);
  SELECT (SELECT DBA.Investigador.email
    FROM DBA.Investigador
    WHERE DBA.Investigador.idInvestigador = INVID)
  INTO @old_email;

  IF EXISTS (SELECT DBA.Investigador.email
    FROM DBA.Investigador
    WHERE DBA.Investigador.idInvestigador = INVID)
  THEN
    UPDATE DBA.Investigador
    SET DBA.Investigador.email = EMAIL,
      DBA.Investigador.nomeInvestigador = USERNAME
    WHERE DBA.Investigador.idInvestigador = INVID;
    DROP USER @old_email;
    CREATE USER EMAIL IDENTIFIED BY 'password';
  END IF;
END;


4. Nome SP: sp_dropInvestigador
// Procedimento que elimina um utilizador investigador e a
sua entrada na tabela
CREATE PROCEDURE sp_dropInvestigador (
  EMAIL VARCHAR(50) DEFAULT '')
```

```
BEGIN
  IF EXISTS (SELECT * FROM dbo.sysusers
    where dbo.sysusers.name = EMAIL) THEN
    EXECUTE IMMEDIATE 'DROP USER ' || EMAIL;
  END IF;

  IF EXISTS (SELECT * FROM DBA.Investigador
    where DBA.Investigador.email = EMAIL) THEN
    EXECUTE IMMEDIATE 'DELETE FROM DBA.Investigador
    WHERE DBA.Investigador.email = ' || EMAIL;
  END IF;
END;


5. Nome SP: sp_softDeleteInvestigador
// Altera o atributo deleted para 1 (eliminado)

create    procedure    "DBA"."sp_softDeleteInvestigador"(IN
arg_id INTEGER)
BEGIN
  UPDATE Investigador SET deleted = 1
  WHERE idInvestigador = arg_id
END;


6. Nome SP: sp_softDeleteCultura
// Altera o atributo deleted para 1 (eliminado)

create  procedure  "DBA"."sp_softDeleteCultura"(IN  arg_id
INTEGER)
BEGIN
  UPDATE Cultura SET deleted = 1
  WHERE idCultura = arg_id
END;


7. Nome SP: sp_softDeleteVariaveis
// Altera o atributo deleted para 1 (eliminado)

create  procedure  "DBA"."sp_softDeleteVariaveis"(IN  arg_id
INTEGER)
BEGIN
  UPDATE Variaveis SET deleted = 1
  WHERE idVariavel = arg_id
END;


8. Nome SP: sp_softDeleteVariaveisMedidas
// Altera o atributo deleted para 1 (eliminado)

create  procedure  "DBA"."sp_softDeleteVariaveisMedidas"(IN
```

```
arg_idVar INTEGER, IN arg_idCul INTEGER)
BEGIN
  UPDATE VariaveisMedidas SET deleted = 1
  WHERE idVariavel = arg_idVar AND idCultura = arg_idCul
END;


9. Nome SP: sp_softDeleteMedicoes
// Altera o atributo deleted para 1 (eliminado)

create  procedure  "DBA"."sp_softDeleteMedicoes"(IN  arg_id
INTEGER, IN arg_idCult INTEGER, IN arg_idVar INTEGER)
BEGIN
  UPDATE Medicoes SET deleted = 1
  WHERE numeroMedicao = arg_id AND idCultura = arg_idCult
  AND idVariavel = arg_idVar
END;
```

2.6.6 Lista de Views

Lista de Views (para cada SP assinalar com x em célula correspondente)

	Implementado de Acordo com Especificado	Implementado mas diferente de Especificado	Não Implementado	Não Especificado (criado de novo)
v_InvestigadorPorInvestigador	X			
v_CulturaPorInvestigador	X			
v_VariaveisPorInvestigador	X			
v_VariaveisMedidasPorInvestigador	X			
v_MedicoesPorInvestigador	X			

2.6.7 Views Implementadas

```
1. Nome View: v_InvestigadorPorInvestigador
//Filtra a tabela Investigadores para o utilizador ligado
create view "DBA"."InvestigadorPorInvestigador"()
AS
SELECT * FROM Investigador WHERE email = user_name() AND
deleted = 0;


2. Nome View: v_CulturaPorInvestigador
//Filtra a tabela Cultura para o utilizador ligado
create view "DBA"."CulturaPorInvestigador”()
AS
SELECT
  Cultura.idCultura,
  Cultura.idInvestigador,
  Cultura.nomeCultura,
  Cultura.limiteInferiorTemperatura,
  Cultura.limiteSuperiorTemperatura,
  Cultura.limiteInferiorHumidade,
  Cultura.limiteSuperiorHumidade,
  Cultura.deleted
FROM Cultura, Investigador
WHERE Cultura.idInvestigador = Investigador.idInvestigador
AND Investigador.email = user_name()
AND Cultura.deleted = 0;


3. Nome View: v_VariaveisPorInvestigador
//Filtra a tabela Variaveis para o utilizador ligado
create view "DBA"."VariaveisPorInvestigador"()
AS
SELECT
  Variaveis.idVariavel,
  Variaveis.nomeVariavel,
  Variaveis.deleted
FROM Variaveis, VariaveisMedidasPorInvestigador
WHERE Variaveis.idVariavel =
VariaveisMedidasPorInvestigador.idVariavel AND
Variaveis.deleted = 0;


4. Nome View: v_VariaveisMedidasPorInvestigador
//Filtra a tabela VariaveisMedidas para o utilizador ligado
create view "DBA"."VariaveisMedidasPorInvestigador"()
AS
SELECT
```

```
  VariaveisMedidas.idCultura,
  VariaveisMedidas.idVariavel,
  VariaveisMedidas.limiteInferior,
  VariaveisMedidas.limiteSuperior,
  VariaveisMedidas.deleted
FROM VariaveisMedidas, CulturaPorInvestigador
WHERE VariaveisMedidas.idCultura =
CulturaPorInvestigador.idCultura AND
VariaveisMedidas.deleted = 0;


5. Nome View: v_MedicoesPorInvestigador
//Filtra a tabela Medicoes para o utilizador ligado
create view "DBA"."MedicoesPorInvestigador"()
AS
SELECT
  Medicoes.idCultura,
  Medicoes.idVariavel,
  Medicoes.numeroMedicao,
  Medicoes.dataHoraMedicao,
  Medicoes.deleted
FROM Medicoes, CulturaPorInvestigador
WHERE CulturaPorInvestigador.idCultura = Medicoes.idCultura
AND Medicoes.deleted = 0;
```

2.7 *Especificação de Migração entre Bases de Dados*

2.7.1 Esquema relacional da base de Dados Mysql especificada (destino)

O diagrama de tabelas para implementação no Sistema de Gestão de Bases de Dados (SGBD) "MySQL" contém as tabelas a disponibilizar ao perfil "Auditor" para o exercício das suas funções. A generalidade das tabelas tem uma definição exatamente igual à das suas correspondentes com o mesmo nome no SGBD "SQL Anywhere", com apenas uma modificação: no MySQL as chaves primárias das tabelas não são auto-incrementadas. Pelo contrário, pretende-se que o conteúdo das chaves primárias nestas tabelas seja uma cópia fiel das chaves das suas correspondentes no "SQL Anywhere", de modo a permitir a execução do processo de migração incremental dos dados sem falhas.

No caso específico da tabela "LogHumidadeTemperatura", e por ser entendido que um utilizador apenas (o sensor) fará única e exclusivamente operações de inserção de dados, não existe uma tabela "LogHumidadeTemperatura" implementada no "SQL Anywhere", o que seria meramente redundante, sendo esperado que a migração de dados seja, portanto, neste caso feita diretamente a partir da tabela base (HumidadeTemperatura).

As tabelas implementadas no "MySQL" são vistas do lado do "SQL Anywhere" na qualidade de "proxy tables", com o sufixo "_remote" (exemplo: a tabela "LogInvestigador" é vista como "proxy table" do lado do "SQL Anywhere" com o nome "LogInvestigador_remote").

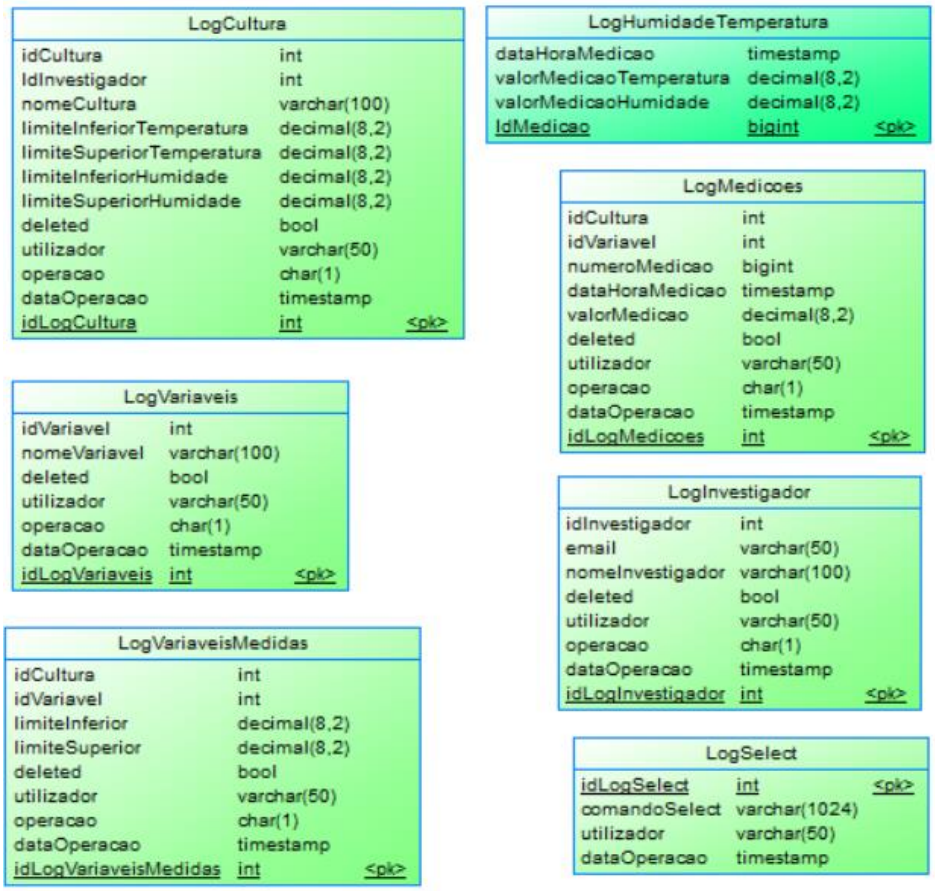


Figura 6 - Diagrama físico da BD MySQL

2.7.2 Forma de Migração Especificada

Depois de alguma análise, optamos por escolher a migração recorrendo a “proxy tables”. Para além de maior facilidade de implementação, este método tem também vantagens ao nível da segurança e a possibilidade de fazer “rollback” das ações, em caso de ocorrência de erros, tais como falhas de ligação ou dados incoerentes, automaticamente. No caso das “remote procedures”, este tipo de fiabilidade contra falhas teria de ser efetuado manualmente.

De maneira a tornar o sistema robusto, têm de ser garantidos os seguintes requisitos:

- O sistema é tolerante a perdas quando (em detalhe mais me baixo):
  - O Servidor MySQL vai a baixo e o Sybase tenta contacta-lo via ODBC;
  - O Servidor MySQL vai a baixo durante uma transferência de dados ODBC;
  - O Servidor Sybase vai a baixo durante uma transferência de dados ODBC.
- A Ação de migração deve acontecer quando (ver secção 1.5.6):
  - O servidor Sybase inicia (evento “Database started”);
  - Manualmente;
  - Periodicamente a cada hora (frequência ajustável na configuração do respetivo evento).
- Quando a migração acontece (seja por qualquer um dos três eventos referidos em cima), terá de garantir que apenas dados com id superior (“primary key” das tabelas de “logs”), ao que se encontra nas tabelas de “logs” no “MySQL” (ou “proxy tables”), são enviados. Esta verificação deve ser feita para cada tabela e antes da migração de quaisquer dados. Tal é assegurado por uma “stored procedure” (ver secção 1.5.5.2).

A migração de dados a partir da utilização de “proxy tables”, é efetuada usando o túnel seguro do “driver” ODBC via TCP, o que adiciona fiabilidade à ligação. A migração é iniciada pelo Sybase, enviando a informação diretamente às “proxy tables”. Uma “proxy table” não é mais que uma tabela virtual do lado do servidor Sybase, relativamente à qual podem ser executados “queries” como se de uma tabela própria se tratasse.

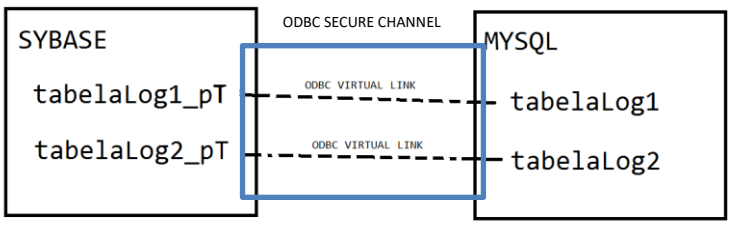


Figura 7 - Diagrama do túnel ODBC

Esta solução tem também a vantagem de ser muito mais fácil fazer a gestão das tabelas do servidor “MySQL”, uma vez que as tabelas se encontram disponíveis para consultas e ações no próprio Sybase. No caso das “remote procedures” tal não era possível, a menos que a implementação fosse desenvolvida para que as tabelas fossem replicadas para outras tabelas

na base de dados, o que seria péssimo pois assim teríamos as tabelas de “logs” replicadas no Sybase “SQL Anywhere”. Sendo as “proxy tables” definições virtuais de tabela existentes fisicamente, sempre que uma dessas tabelas é acedida no Sybase, a informação é pedida ao MySQL.

Para que o Sybase consiga aceder ao MySQL remotamente através do “driver” ODBC é necessário ter instalado e configurado o “driver” ODBC do MySQL na máquina que correrá o Sybase “SQL Anywhere”. O contrário não é necessário, a driver ODBC do Sybase não tem de estar instalada na máquina que correrá o MySQL. Depois de configurada a driver ODBC, é necessário criar um “remote server”, de maneira a que o Sybase “SQL Anywhere” lhe consiga aceder, criar o “link” às tabelas MySQL e correr “queries” diretamente nessas tabelas.

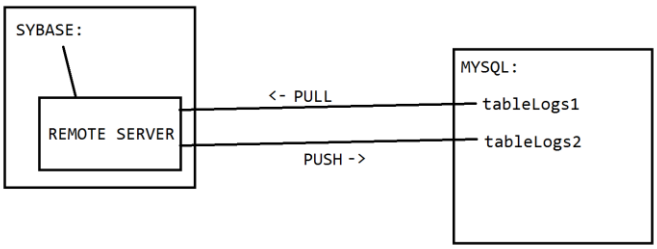


Figura 8 - Diagrama de implementação do remote server

Depois de criado o “remote server” e as “proxy tables”, o Sybase “SQL Anywhere” tem a capacidade de:

- Inserir dados nas “proxy tables” (PUSH), sendo tais dados automaticamente replicados para as tabelas MySQL;
- Recolher a informação proveniente nas tabelas de “logs” no “MySQL”, para as “proxy tables” no Sybase “SQL Anywhere” (PULL).

Para que a comunicação seja conseguida é necessário também criar um “external login”, o qual não é mais que uma “simulação” de um utilizador do “MySQL “, onde o ODBC terá de se ligar. Por motivos de segurança, não é aconselhável que o “external login” tenha as credenciais “root” do servidor “MySQL”, pelo que deverá ser criado um utilizador especifico do lado do “MySQL”, com apenas as permissões necessárias para que a migração ocorra (ver secção 1.5.3).

Uma das vantagens de usar as “proxy tables”, tal como referido anteriormente, é que o Sybase “SQL Anywhere” trata de fazer “rollback” à migração da tabela de “logs” original para a “proxy table” automaticamente, em caso de acontecer qualquer umas das seguintes três situações:

- O Servidor “MySQL” vai a baixo e o Sybase tenta contacta-lo via ODBC.
- O Servidor “MySQL” vai a baixo durante uma transferência de dados ODBC.
- O Servidor Sybase “SQL Anywhere” vai a baixo durante uma transferência de dados ODBC.

Desta forma temos garantia que não existirão dados duplicados ou em falta usando as “proxy tables”.

2.7.3 Utilizadores Especificados

No MySQL, irá existir apenas um utilizador: o auditor, sem excluir o utilizador DBA, que terá acessos e permissões completas na base de dados.

Auditor

O auditor apenas poderá fazer selects às tabelas, não tendo restrições do comando, como implementado nos Investigadores do lado do Sybase. O select incluirá todas as colunas de todas as tabelas. No entanto não poderá inserir, eliminar ou atualizar dados nas tabelas, nem alterar a estrutura da base de dados, esta função será da responsabilidade do DBA, que, à semelhança do Sybase, irá pertencer à equipa de desenvolvimento.

Permissões	Auditor
Tabelas	
LogCultura	S
LogHumidadeTemperatura	S
LogVariaveis	S
LogVariaveisMedidas	S
LogMedicoes	S
LogInvestigador	S
LogSelect	S

Tabela 7 - Lista de permissões do utilizador Auditor

MySQLRemote

Para que o sistema possa ser utilizado é necessária a criação de um external login. Esta ação pode ser executada através da interface do Sybase, a quando da criação do remote server, ou através de script SQL.

Como dito anteriormente é necessário que haja um login do lado do MySQL, onde as credenciais são replicadas no external login. Por motivos de segurança é aconselhável não usar o utilizador root do MySQL. Deve-se então criar um utilizador “mysqlremote”, que terá acesso apenas às tabelas de Logs MySQL.

Permissões	mysqlremote
Tabelas	
logcultura	S, I
loghumidadetempertura	S, I
loginvestigador	S, I
logmedicoes	S, I
logelect	S, I
logvariaveis	S, I
logvariaveismedidas	S, I
Views	-
Store Procedures	-

Tabela 8 - Permissões do utilizador MySQLRemote

2.7.4 Triggers de suporte à migração de dados especificados

Para a base de dados em questão não se torna necessária a implementação de quaisquer triggers, para além do referido no parágrafo 1.3 - Gestão de Logs.

2.7.5 Stored Procedures de suporte à migração de dados especificados

SP para sincronização

Para que a migração (transferência dos dados das tabelas de Logs para as proxy tables) aconteça, é necessário criar um SP.

Nome SP	Parâmetro Entrada	Parâmetro Saída	BD	Nota
sp_syncRemote	-	NULL	Sybase	

Este SP deve executar as seguintes tarefas:

1. Verificar o último id de cada tabela (por exemplo idLogCultura) na proxy table correspondente à tabela log (por exemplo na tabela logcultura_remotemysql)
2. Exportar para a proxy table todos os dados caso o último id encontrado não exista (seja NULL).
3. Exportar todos os dados em que o id (por exemplo idLogCultura) da tabela log, seja maior que o último id encontrado no passo um.

SP para gerar dados

Para ajudar nos testes do sistema é aconselhável (mas opcional) a criação de um outro script SQL, para a criação de dados brutos.

Nome SP	Parâmetro Entrada	Parâmetro Saída	BD	Nota
sp_geraData	Número de linhas a adicionar em cada tabela	NULL	Sybase	

O script não é mais do que um ciclo while, que percorre o número dado no parâmetro de entrada, onde para cada tabela faz um insert completo simulando os dados de alguma maneira.

Desta maneira é possível testar a migração ODBC a 100%, criando dados sempre que preciso, consistentes e de certa forma uma boa simulação a dados reais. Sabemos que os testes por aleatoriedade não são os mais aconselháveis, mas neste modelo são bastante fáceis de implementar já que muitas tabelas se compõem apenas de datas, doubles e strings.

2.7.6 Eventos de suporte à migração de dados especificados

Como referido na secção 1.5.2, o sistema deve assegurar que a migração acontece nos seguintes três casos:

1. O servidor Sybase inicia.
2. Manualmente.
3. Periodicamente a cada hora.

Para isso será necessário a implementação de eventos Sybase, um para cada caso.

Nome Evento	**DB**	**Nota**
manualStartSync	Sybase	Execução manual da migração através do SP
onStartDbSync	Sybase	Inicia o SP assim que o server é iniciado
scheduledStartSync	Sybase	Inicia o SP a cada hora

Tabela 9 - Listagem dos eventos de suporte à migração

2.8 Avaliação das especificações do próprio grupo Migração

Qualidade (Fraca, Razoável, Boa ou Muito Boa): Muito Boa

O método escolhido para a migração dos dados traz uma série de vantagens. Começando pela simplicidade, a utilização das proxy tables traz uma facilidade na implementação e migração dos dados.
Como poderá ser lido nos capítulos conclusivos da especificação, apesar de ser um processo mais lento, traz uma série de vantagens a nível de robustez, fiabilidade e segurança.
Através de eventos calendarizados, é possível garantir um correto tempo de migração dos dados, não permitindo acumular grande quantidade de registos entre as migrações.

2.9 Implementação da Migração de Dados

2.9.1 Utilizadores Implementado

Permissões	Auditor	mySqlRemote
Tabelas		
LogCultura	S	S, I
LogHumidadeTemperatura	S	S, I
LogVariaveis	S	S, I
LogVariaveisMedidas	S	S, I
LogMedicoes	S	S, I
LogInvestigador	S	S, I
LogSelect	S	S, I

2.9.2 Lista Triggers

Lista de Triggers (para cada trigger assinalar com x em célula correspondente)

Nada a referir

2.9.3 Triggers Implementados

```
Nada a referir
```

2.9.4 Lista de Stored Procedures

Lista de SP (para cada SP assinalar com x em célula correspondente)

	Implementado de Acordo com Especificado	Implementado mas diferente de Especificado	Não Implementado	Não Especificado (criado de novo)
sp_syncRemote	X			
sp_geraData	X			

2.9.5 Stored Procedures Implementados

```
1. Nome SP: sp_syncRemote
// Procedimento que sincroniza a informação a transportar
entre as BDs (só será colocado um excerto do código devido
à dimensão do mesmo)
CREATE PROCEDURE "DBA"."sp_syncRemote"(IN arg_id INTEGER
DEFAULT NULL)

BEGIN
  DECLARE lastid INTEGER;

  SET lastid = (SELECT MAX(idLogCultura)
  FROM logcultura_remote);
  IF (arg_id IS NULL) THEN
    IF (lastid IS NULL) THEN
      INSERT logcultura_remote SELECT * FROM LogCultura;
    ELSE
      INSERT logcultura_remote SELECT * FROM LogCultura
      WHERE idLogCultura > lastid;
    END IF;
  ELSE
    IF (lastid IS NULL) THEN
      INSERT logcultura_remote SELECT TOP arg_id *
      FROM LogCultura ORDER BY idLogCultura ASC;
    ELSE
      INSERT logcultura_remote SELECT TOP arg_id *
      FROM LogCultura WHERE idLogCultura > lastid
      ORDER BY idLogCultura ASC;
    END IF;
  END IF;
  COMMIT;

  SET lastid = (SELECT MAX(idMedicao)
  FROM loghumidadetemperatura_remote);
  IF (arg_id IS NULL) THEN
    IF (lastid IS NULL) THEN
      INSERT loghumidadetemperatura_remote SELECT *
      FROM HumidadeTemperatura;
    ELSE
      INSERT loghumidadetemperatura_remote SELECT *
      FROM HumidadeTemperatura WHERE idMedicao > lastid;
    END IF;
  ELSE
    IF (lastid IS NULL) THEN
      INSERT loghumidadetemperatura_remote SELECT
      TOP arg_id * FROM HumidadeTemperatura
      ORDER BY idMedicao ASC;
```

```
    ELSE
      INSERT loghumidadetemperatura_remote SELECT
      TOP arg_id * FROM HumidadeTemperatura
      WHERE idMedicao > lastid ORDER BY idMedicao ASC;
    END IF;
  END IF;
  COMMIT;

  (…)

  SET lastid = (SELECT MAX(idLogVariaveisMedidas)
  FROM logvariaveismedidas_remote);
  IF (arg_id IS NULL) THEN
    IF (lastid IS NULL) THEN
      INSERT logvariaveismedidas_remote SELECT *
      FROM LogVariaveisMedidas;
    ELSE
      INSERT logvariaveismedidas_remote SELECT *
      FROM LogVariaveisMedidas
      WHERE idLogVariaveisMedidas > lastid;
    END IF;
  ELSE
    IF (lastid IS NULL) THEN
      INSERT logvariaveismedidas_remote SELECT TOP arg_id *
      FROM LogVariaveisMedidas
      ORDER BY idLogVariaveisMedidas ASC;
    ELSE
      INSERT logvariaveismedidas_remote SELECT TOP arg_id *
      FROM LogVariaveisMedidas
      WHERE idLogVariaveisMedidas > lastid
      ORDER BY idLogVariaveisMedidas ASC;
    END IF;
  END IF;
  COMMIT;
END


2. Nome SP: sp_geraData
//  Procedimento  para  gera  n  registos  para  testes  de
migração e funções (só será colocado um excerto do código
devido à dimensão do mesmo)
CREATE PROCEDURE "DBA"."sp_GeraDados"( IN arg_id INTEGER )

BEGIN
  WHILE intCount < arg_id LOOP
    SET lastIdInvestigador = lastIdInvestigador + 1;
    SET lastIdCultura = lastIdCultura + 1;
    SET lastIdVariaveis = lastIdVariaveis + 1;
    SET intCount = intCount + 1;

    INSERT INTO Investigador (
```

```
      email, nomeInvestigador, deleted)
    VALUES ('user' + CAST(lastIdInvestigador AS VARCHAR) +
      '@sid.pt', 'Investigador ' +
      CAST(lastIdInvestigador AS VARCHAR),0);

    (…)

    INSERT INTO Variaveis (nomeVariavel, deleted)
    VALUES ('Temp' + CAST(lastIdVariaveis AS VARCHAR), 0);

    SET idCulturaParaVariaveisMedidas =
     (SELECT FIRST idCultura FROM Cultura ORDER BY RAND());
    SET idVariavelParaVariaveisMedidas =
     (SELECT FIRST idVariavel FROM Variaveis
     ORDER BY RAND());

    WHILE (
      SELECT idCultura FROM VariaveisMedidas
      WHERE idCultura = idCulturaParaVariaveisMedidas
      AND idVariavel = idVariavelParaVariaveisMedidas)
      IS NOT NULL LOOP

      SET idCulturaParaVariaveisMedidas =
       (SELECT FIRST idCultura FROM Cultura
       ORDER BY RAND());

      SET idVariavelParaVariaveisMedidas =
       (SELECT FIRST idVariavel FROM Variaveis
       ORDER BY RAND());
    END LOOP;

(…)

  END LOOP;
END
```

2.9.6 Lista Eventos

Lista de Eventos (para cada evento assinalar com x em célula correspondente)

	Implementado de Acordo com Especificado	Implementado mas diferente de Especificado	Não Implementado	Não Especificado (criado de novo)
manualStartSync	X			
onStartDbSync	X			
scheduledStartSync	X			

2.9.7 Eventos Implementados

```
1. Nome Evento: manualStartSync
// Para despoletar manualmente o evento calendarizado
TRIGGER EVENT Export_to_MySQL_Scheduled;



2. Nome Evento: onStartDbSync
// Breve Descrição
CREATE EVENT "DBA"."onStartDbSync"
TYPE databasestart
HANDLER
BEGIN
    TRIGGER EVENT Export_to_MySQL_Scheduled;
END;


3. Nome Evento: scheduledStartSync
// Breve Descrição
CREATE EVENT "DBA"."scheduledStartSync"
SCHEDULE schedule_every_hour
     START TIME '12:00:00'
     EVERY 1 HOURS
HANDLER BEGIN
     ----------------------------
     CALL DBA.sp_syncRemote();
     ----------------------------
     MESSAGE STRING (
     'Evento ' ,
     EVENT_PARAMETER ( 'EventName' ),
        ' despoletado em ',
     CURRENT TIMESTAMP,
     ' devido ao schedule ',
     EVENT_PARAMETER ( 'ScheduleName' )) TO CONSOLE;
END;
```

2.10 Avaliação Global da Qualidade das Especificações do próprio grupo

Avaliação (A,B,C,D,E) : E

Utilize a seguinte escala:
A: - 1 – 5 valores B: 6 – 9 valores C: 10 – 13 Valores D: 14 – 17 valores E: 18 – 20 valores

Três principais deficiências de especificação que tiveram impacto mais negativo na qualidade da implementação

Falta de processos de validação de dados nos vários SP's, permitindo o disparo de erros não tratados.

Complexidade na implementação do SP logSelect que, apesar de tornar a BD mais robusta no registo das consultas, poderá dificultar a manutenção futura da mesma.

Resumo de Avaliações de Qualidade Anteriores (para cada linha assinalar com x em célula correspondente)

	Fraco	Razoável	Bom	Muito Bom
BD Sybase				X
Triggers Log				X
SP Log				X
Views Log				X
Utilizadores Log				X
BD Mysql				X
Forma Migração				X
Triggers Migração	n/a			
SP Migração				X
Eventos Migração				X
Utilizadores Migração				X

2.11 Comparação de Implementações (ficheiro versos ODBC)

Durante a realização do projeto implementamos várias versões e várias abordagens para efetuar a migração entre base de dados, de maneira a escolher a mais eficaz em termos de segurança, facilidade de implementação e manutenção, rapidez e fiabilidade. De entre todos os testes apresentamos no relatório alguns:

- Migração ODBC através de Proxy Tables (nossa especificação)

- Migração ODBC através de Remote Procedures

- Migração FILE através de ficheiros TXT

- Migração FILE através de ficheiros CSV

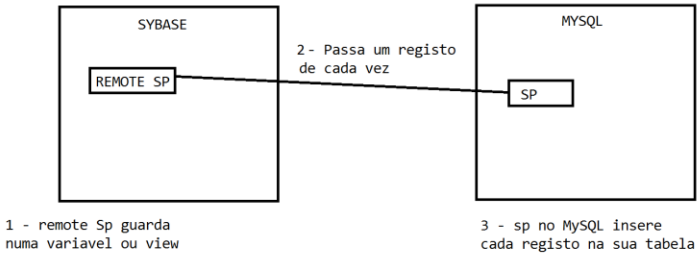
Antes de estudar as diferenças entre estas é importante perceber as grandes diferenças destas abordagens.

Migração ODBC através de Proxy Tables

No caso da migração por ODBC, tornou-se muito mais eficaz a utilização de proxy tables. Este mecanismo que o Sybase e o MySQL proporcionam, torna a migração entre base de dados algo bastante fácil. Esta migração foi a especificada ao grupo 11 e por isso a explicação completa encontra-se no capítulo 2.8.

Migração ODBC através de Remote Procedures

Para garantirmos a qualidade da especificação ao grupo 11 na etapa A, fizemos alguns testes a partir de Remote Procedures, mas rapidamente se tornou uma ideia a não considerar. Em primeiro lugar, e para o caso da migração, toda a implementação relativa a robustez teria de ser efetuada manualmente, ao contrário da implementação com proxy tables que automaticamente gere toda a ligação e em caso de falha faz roolback. Ainda em termos de implementação, toda a implementação é bastante mais complicada, visto que é necessário carregador todos os dados para uma variável ou uma view, e num ciclo while através de cursores, invocar o remote sp para cada linha da tabela, passando os dados como argumento ao remote sp. Este remote sp, invoca o sp correspondente ao sp do lado do MySql, que recebe como argumentos os dados e através de por exemplo uma variável indicando a tabela, insere nessa tabela os dados correspondentes. Este método, e a partir de alguns testes (apenas para uma tabela e um número alto de linhas), verificamos que era muito mais lento que qualquer um dos outros mencionados. No entanto os remote sp's, têm uma grande vantagem, podem dar jeito para algumas ações na base de dados mysql, ou mesmo para receber dados do mysql, como demonstrado em alguns casos na migração por ficheiro a seguir.



Migração FILE através de ficheiros TXT e CSV

Para a implementação da migração TXT e CSV, usamos o especificado pelo grupo 11. Esta abordagem para a migração demonstrou-se bem mais rápido que o ODBC, e com menos processamento para o Sybase e MySQL, no entanto tem os seus problemas e questões que se podem levantar, questões essas que não acontecem com o método ODBC, como:

a. **O ficheiro desapareceu** – explicado no capítulo 2.11.2
b. **Transferência do ficheiro entre máquinas –** explicado no capítulo 2.11.2
c. **A migração começa a meio da transferência do ficheiro de uma máquina Sybase para uma máquina MySQL ou o ficheiro não atualiza automaticamente na máquina –** explicado 2.11.2
d. **O ficheiro bat pode conter informação sobre login na base de dados MySQL** - explicado no capítulo 2.11.4
e. **Os ficheiros podem conter informação privada -** explicado no capitulo 2.11.4
f. **Utilizar um ficheiro novo a cada migração ou escrever sempre no mesmo ficheiro –** explicado no capitulo 2.11.2
g. **A máquina MySQL está em baixo ou vai a baixo a meio de uma migração –** explicado no capitulo 2.11.2
h. **TXT vs CSV -** por último, mas não menos importante, nasce a questão, se a migração por ficheiro deve ser feita em formato CSV ou TXT. O grupo 11 escolheu o formato TXT. Para o estudo da questão, começamos por verificar os tamanhos dos ficheiros em disco e, após os testes, chegamos à conclusão que para até meio milhão de dados o tamanho é igual para ambos os casos.

# lines	Tamanho total dos ficheiros em disco (bytes)
10	28672
50	32768
100	61440
500	266240
1000	516096
5000	2568192
10000	5136384
25000	12918784
50000	25899008
100000	51867648
500000	261931008

De seguida, passamos a algumas pesquisas na internet. De facto, e em praticamente todos os casos, CSV tornou-se menos suscetível a erros que o TXT. Verificamos também que utilizar txt não faz sentido, visto que na documentação Sybase o exemplo aparece com ficheiro csv, e os dados exportados para o ficheiro TXT, são na verdade ficheiros CSV (podemos verificar isso pelo delimitador).

```
UNLOAD TABLE Employees TO 'employee_data.csv';
```

```
1,1964-07-03,05:24:37.756,'A',5,'Cultura1',4,2,3,5,'user1@sid.pt','Coluna1','55.5','55.5'
```

O CSV apresenta apenas um problema, que é no caso de existir um caracter no meio dos dados (que não sejam strings) igual ao delimitador. No nosso caso isso não é um problema, porque todas os dados decimais são passados comos Strings e os Integers, não podem obviamente conter virgulas.

Com isto, e comparando as desvantagens e vantagens da abordagem ODBC com a bordagem por ficheiro, torna-se evidente que a melhor abordagem é a abordagem por ODBC.

2.11.1 Eficiência de Migração

Testes

Todos os testes foram efetuados através de uma rede bridge de 10Mbps e entre duas VM's Windows 10 (todo o sistema é apresentado e explicado no capítulo 2.13). Foi garantido também que ambas as máquinas não tinham processos paralelos a correr de grande dimensão e que ambas partilhavam de recursos idênticos da máquina host (2 cores e 4gb de ram). Para os dados foram utilizados os testes com número de linhas igual a:

lines
10
50
100
500
1000
5000
10000
25000
50000
100000
500000

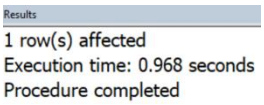
Desta forma, conseguimos garantir a qualidade dos nossos testes para as duas abordagens implementadas:

- ODBC Proxy Tables
- FILE TXT (sem transferência de ficheiro)

Teste ODBC Proxy Tables

Na máquina onde corre o Sybase, foi retirado os seguintes valores:

a. Valor apresentado na interface Sql iteractive (tempo total – processamento + envio):



b. Tempo de transferência através do Wireshark (tempo de envio):

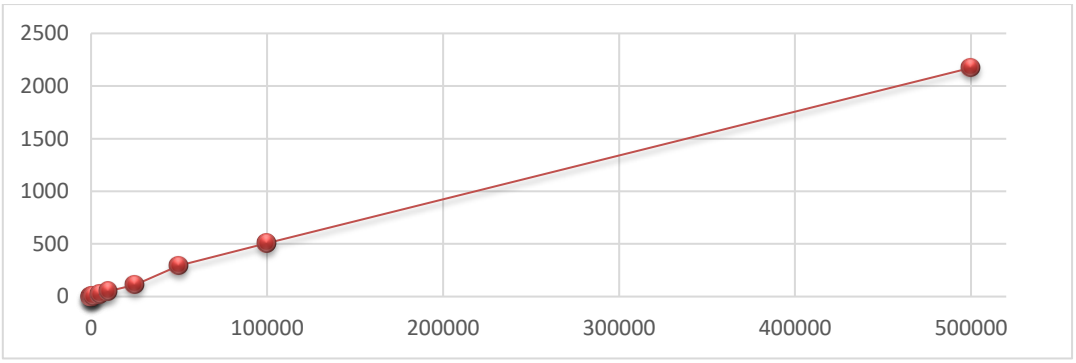
valor do último pacote TCP recebido – (menos) valor do último pacote TCP recebido

c. O tempo de processamento, não é mais que o total apresentado na interface menos o tempo de envio:

tempo de processamento = total - tempo de envio

Com os testes terminados obtivemos os seguintes resultados, para cada conjunto de linhas:

ODBC Proxy Tables			
# lines	tempo de envio (Wireshark)	tempo de processamento (total - envio)	TOTAL (processamento sybase + tempo de envio)
10	0,085607	0,007393	0,093
50	0,252471	0,012529	0,265
100	0,581276	0,027724	0,609
500	2,460431	0,008569	2,469
1000	4,925363	0,011637	4,937
5000	24,634628	0,084372	24,719
10000	49,877081	0,013919	49,891
25000	112,145553	0,276447	112,422
50000	293,74434	0,39466	294,139
100000	507,421558	0,145442	507,567
500000	2172,699852	0,285148	2172,985



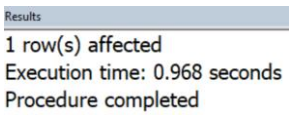
Analisando os dados, conseguimos verificar que o tempo de processamento do lado do Sybase de preparação para o envio de dados é praticamente nulo e pode ser desprezado, sendo que todo o restante tempo é processamento para o envio dos dados através da rede.

Podemos analisar também que o aumento do tempo em função dos dados enviados é bastante linear, por exemplo, para 100000 linhas temos um tempo de 507 segundos, já no caso dos 500000 temos 2172, que é mais ao menos 5 vezes o tempo com 100000 linhas, o que faz todo o sentido.

Teste FICHEIRO TXT

Na máquina onde corre o Sybase, foi retirado os seguintes valores:

a. Valor apresentado na interface Sql iteractive - exportação (tempo de processamento sybase)



b. Diferença dos valores apresentados no ficheiro bat – importação (tempo de processamento mysql)

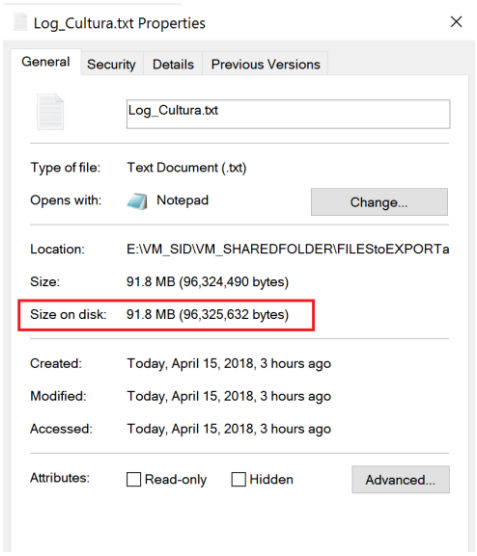
```
C:\Users\sid18\Desktop>echo Start Time: 19:19:20,11
Start Time: 19:19:20,11

C:\Users\sid18\Desktop>echo Finish Time: 19:19:53,19
Finish Time: 19:19:53,19
```

c. Total = processamento sybase + processamento mysql.

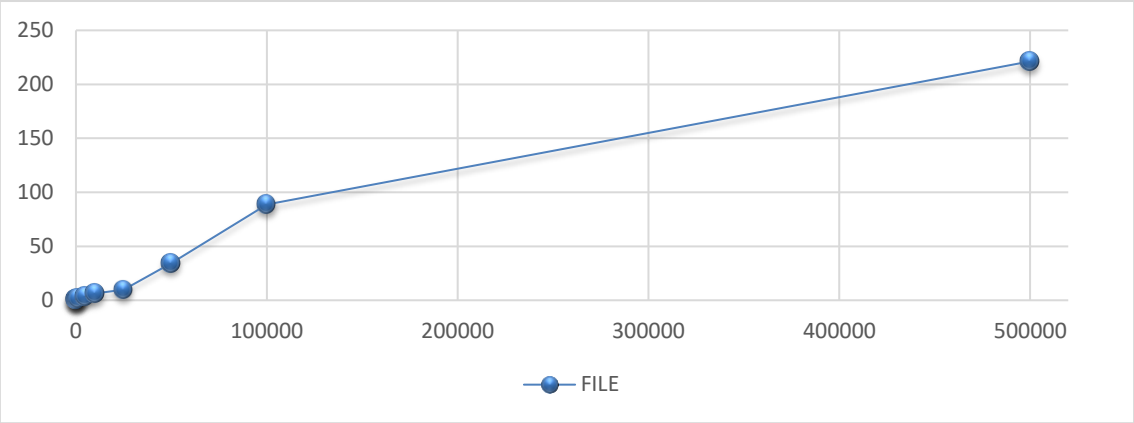
Nota: Os testes não incluem a transferência de o ficheiro entre máquinas (dado Wireshark). Todos os testes foram feitos através de uma pasta partilhada na máquina host. Com o atraso de envio do ficheiro, e dependendo do ficheiro, era possível que tivéssemos um aumento de alguns segundos no tempo total e completo de migração.

d. Tamanho do ficheiro através das propriedades do ficheiro.



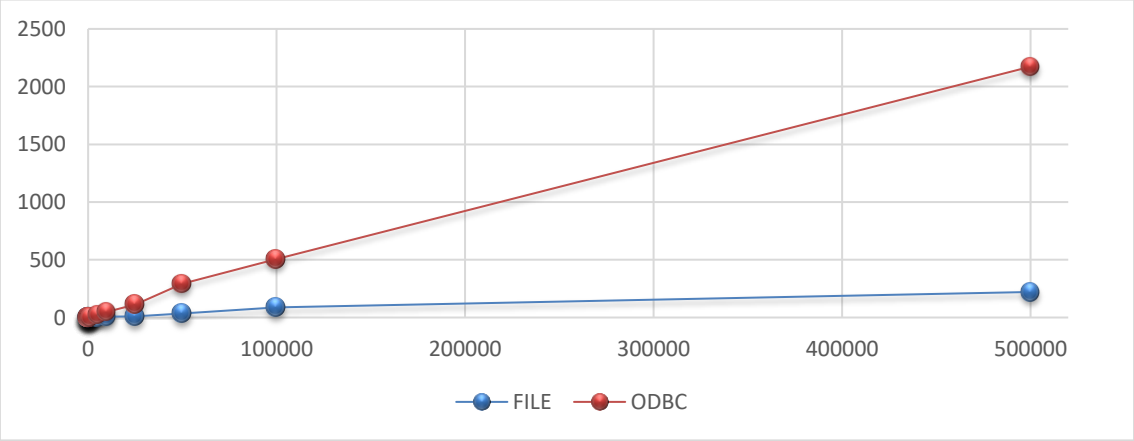
Com os testes terminados obtivemos os seguintes resultados, para cada conjunto de linhas:

FILE TXT				
# lines	tempo de processamento Sybase	tempo de processamento MYSQL (bat file)	TOTAL (processamento sybase + processamento mysql)	Tamanho total dos ficheiros em disco (bytes)
10	0,171	0,66	0,831	28672
50	0,178	0,78	0,958	32768
100	0,213	1,01	1,223	61440
500	0,314	1,19	1,504	266240
1000	0,359	1,38	1,739	516096
5000	0,406	3,53	3,936	2568192
10000	0,656	5,6	6,256	5136384
25000	1,171	8,75	9,921	12918784
50000	3,11	31,43	34,54	25899008
100000	5,687	83,14	88,827	51867648
500000	12,719	208,51	221,229	261931008



Analisando então os dados obtidos relativos à migração por ficheiro identificamos também que é bastante linear face à quantidade de dados. Quando ao nível de processamento do sybase e do mysql, torna-se muito mais eficaz, pois só processa durante o tempo de exportação e importação para o ficheiro, enquanto que no ODBC é mais extenso, pois o processamento está presente durante todo o envio e receção de dados através da rede, como explicado anteriormente.

ODBC VS FICHEIRO



Comparando os dois, verificamos que a solução file é muito mais rápida que a solução ODBC. Mesmo adicionando o tempo de transferência do ficheiro entre máquinas, e mesmo que esse tempo fosse de 10 segundos, o que é um tempo considerável, continuaria a ser mais rápido.

No entanto por várias razões explicadas anteriormente, e que irão ser explicadas nos próximos capítulos, o ODBC é a solução melhor para o caso em que os ficheiros não são muito elevados e a periodicidade das migrações é reduzida, visto que acrescenta o rollback automaticamente. Já no caso de ficheiros com tamanho muito elevado, e assumindo que a periodicidade entre as migrações seria elevada, então poder-se-ia optar pela solução FICHEIRO.

O método ODBC é mais lento, porque, como este tem de conseguir fazer roolback da informação automaticamente em caso de falhas, então a informação tem de ser toda guardada, o que pode ocupar bastantes recursos do computador, como a memória.

2.11.2 Robustez

De seguida serão listados problemas e respetivas soluções, algumas implementadas (mais no caso do ODBC) e outras que poderiam ser implementadas de forma a dar mais robustez ao sistema (mais no caso de migração por ficheiro).

Problemas:

1 - O servidor MySQL está em baixo ou vai a baixo durante a migração

Usando ODBC PROXY TABLES

Com as proxy tables e odbc, isto não apresenta um problema, porque como explicado anteriormente, o roolback das acções (inserções), é feito automaticamente em caso de falha. O sistema do grupo 23, está também pronto para lidar, caso o servidor MySQL esteja desativado durante algum tempo, pois verifica sempre o último id antes de indicar a migração.

Usando Ficheiro

Já no caso do ficheiro, isto é um problema, pois os dados vão sendo inseridos, e a menos que seja programado no script a opção de roolback, tal não é efetuado automaticamente.

2 - O ficheiro desapareceu

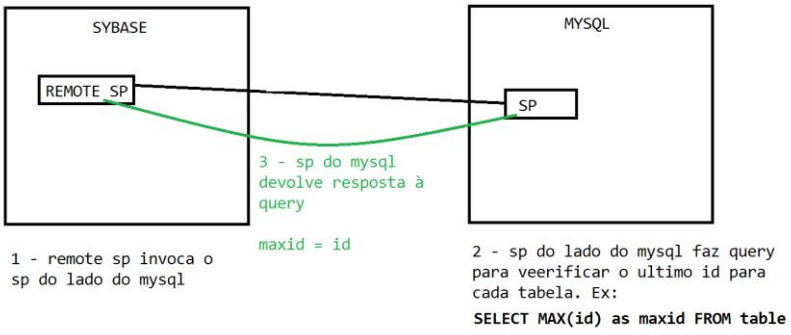
Usando ODBC PROXY TABLES

Não se aplica

Usando Ficheiro

Isto não apresenta um problema, porque o "UNLOAD TO FILE", cria o ficheiro caso ele não exista. N entanto o ficheiro pode ser eliminado na diretoria após o Sybase exportar os dados, ou do MySQL antes de importar os dados. O sybase não tem então maneira de saber que dados foram transferidos. É então sugerida a seguinte solução:

Usando Remote sp's (um para cada tabela de logs), podemos verificar qual é o último id na tabela correspondente no MySQL. Desta maneira, garantimos que a exportação do Sybase para o ficheiro é apenas das linhas com id maior ao último id verificado através do remote sp.



3 - Transferência de ficheiros entre máquinas

Usando ODBC PROXY TABLES

A driver ODBC simplifica toda a comunicação entre máquinas.

Usando Ficheiro

No caso do ficheiro tal não acontece. A transferência do ficheiro entre máquinas torna-se um problema, que poderia ser corrigido usando por exemplo o protocolo FTP, o que adiciona mais complexidade ao sistema.

4 - A migração começa a meio da transferência do ficheiro de uma máquina Sybase para uma máquina MySQL ou o ficheiro não atualiza automaticamente na máquina

Usando ODBC PROXY TABLES

Não se aplica

Usando Ficheiro

A migração do lado do MySQL (através do ficheiro bat) pode começar quando o ficheiro ainda não chegou à sua máquina completamente. Esta é uma das maiores desvantagens da migração por ficheiro, devido à sua complexidade. Juntando a solução apresentada ao problema anterior, poderia ser verificado através do ficheiro bat, verificando se existem pacotes FTP a passar na rede. Caso existissem, teria de reagendar, caso contrário poderia efetuar a migração a partir do ficheiro.

5 - Utilizar um ficheiro novo a cada migração ou escrever sempre no mesmo ficheiro

Usando ODBC PROXY TABLES

Não se aplica

Usando Ficheiro

Esta foi a discussão que mais se debateu no grupo, no entanto podemos afirmar que a abordagem implementada pelo grupo 11 de escrever sempre no mesmo ficheiro é sem dúvidas a que mais problemas pode dar. Como por exemplo, a transferência começar a meio da migração (caso contrário ao apresentado em no problema anterior), originando assim problemas de acesso ao ficheiro, visto já estar a ser utilizado. Já na abordagem de utilizar um ficheiro novo em cada migração de dados, foi comprovada como a mais eficaz, utilizando para isso um remote sp de suporte.

Todo o processo é demonstrado de seguida:

a. O Sybase faz a migração para um novo ficheiro, com um novo name e único. Por exemplo, adicionando um timestamp ao final.
b. O Sybase invoca um remote sp, passando como argumento o nome do ficheiro e transfere o ficheiro.
c. O sp do lado do MySQL insere essa linha numa tabela chamada ‘’log_migracao”.
d. Quando o ficheiro bat inicia a migração, verifica o último ficheiro inserido na base de dados e tenta migrar esse ficheiro. Caso não exista, tenta novamente mais tarde, caso contrário efetua a migração.

2.11.3 Flexibilidade / Dependência

Flexibilidade

Alterar um evento

Usando ODBC PROXY TABLES

Como a migração por ODBC é executada através de eventos, qualquer pessoa (assumindo que tem conhecimento básico do Sybase), pode mudar, executar ou adicionar outros eventos para executar a migração de uma maneira bastante simplificada (como mostrado na especificação do grupo 11).

Usando Ficheiro

No caso do ficheiro, o processo é dividido em dois tempos, e por isso caso quiséssemos alterar a periodicidade a que o Sybase exporta os dados, seria igual é explicada anteriormente para o ODBC. Já no caso da importação, teria de ser mudado no gestor de tarefas, na máquina correspondente MySQL. É de igual maneira bastante simples (supondo que o utilizador tem conhecimentos básicos de Windows).

Migrar número específico de registos

Usando ODBC PROXY TABLES e FICHEIRO

Em ambos os casos é possível definir o número de registos a exportar/transferir. Para isso basta enviar no argumento do stored procedure, um NULL caso queiramos migrar todos, ou um INTEGER que define o número de linhas a importar.

Por exemplo:

```
CALL sp_syncRemote(NULL); ou CALL sp_syncRemote(50000);
```

Este valor é também facilmente editado no evento(s) correspondentes.

Já no caso do MySQL, e para a migração por ficheiro, teria de se editar o ficheiro bat, para importar apenas k registos.

Mudar o nome base do ficheiro

Usando ODBC PROXY TABLES

Não se aplica

Usando Ficheiro

Para que fosse possível mudar o nome do ficheiro, teria de se alterar o sp do lado do sybase, e o bat file do lado do mysql. O que acrescente complexidade e por sua vez gerar problemas caso seja manuseado por pessoas sem conhecimento.

Desta forma, concluímos que o sistema ODBC é de longe muito mias flexível. No entanto, poderíamos aumentar a flexibilidade do sistema por ficheiro adicionando alguns sp's que fizessem as ações anteriores, mas, todas as ações não poderiam ser facilmente agrupadas e executadas em conjunto de uma maneira simplificada.

Dependência

No caso de utilizadores inserirem dados que possam gerar problemas na base de dados:

Usando ODBC PROXY TABLES

Este problema não se aplica, pois, o ODBC não usa delimitadores como o ficheiro.

Usando Ficheiro

No caso do ficheiro, podem surgir problemas por causa do delimitador usado na separação das linhas do ficheiro. Por exemplo, no caso de um campo for enviado com uma "," no meio, podem surgir problemas, porque quando o MySQL tentar fazer o import, irá processar essa linha como uma divisão e como se fosse o começo de outra coluna.

Mais uma vez, e em caso de dependências, o ODBC mostrou-se mais eficaz que o ficheiro.

2.11.4 Segurança

Segurança no caso do ODBC:

No caso do ODBC, o único local onde a informação poderia ser apanhada, era através de um sniffer na rede, por exemplo o Wireshark. Se a informação não fosse devidamente codificada, os dados seriam facilmente visíveis.

Segurança no caso do ficheiro:

Já no caso do ficheiro, os dados, e caso não fossem encriptados, poderiam ser visíveis na máquina ODBC e na máquina MySQL, mesmo sem o utilizador ter acesso á base de dados. Isto apresenta um problema evidente de segurança, pois os dados podiam conter informação sensível.

Outro problema de segurança relacionado com a migração por ficheiro, é a questão do ficheiro bat conter a informação de login:

```
"C:/Program Files/MySQL/MySQLServer/bin/mysql.exe" -u DBA -psqll -h 127.0.0.1 mysqlfile
```

Este problema poderia ser contornado chamando o bat file remotamente, a partir do sybase, e com os dados de login enviados nesse pedido.

2.12 Auditoria de Dados Mysql

A auditoria, no caso da base dados MySQL referente à migração via ODBC, disponibiliza a possibilidade de efetuar "queries" diretamente às tabelas de dados de auditoria, ou através de "views". Estas foram especialmente concebidas para benefício do Auditor, tendo em conta a opção de registar nos "logs" apenas os valores (novos) alterados (necessariamente através de uma operação "insert" ou "update" prévia) e facilitando-lhe, assim, a obtenção por cada linha dos resultados da consulta que eventualmente pretenda efetuar aos "logs", o valor antigo a par com o valor novo de cada atributo de cada linha alterada, no caso das operações de "update", sendo esta facilidade aplicável às tabelas "LogInvestigador", "LogCultura", "LogVariaveis", "LogVariaveisMedidas" e "LogMedicoes"; as duas restantes tabelas de "logs" – "logSelect" e "LogHumidadeTemperatura" – não carecem de vistas deste tipo, dado que não estão sujeitas a operações do tipo "update".

Em todos os casos, é possível obter de cada tabela quem efetuou cada operação sobre os dados, quando foi a mesma efetuada e qual o tipo de operação em apreço. A partir de cada "view", além da informação anterior, é possível obter também os dados existentes na tabela antes da operação "update".

Os chamados "soft deletes" podem ser retornados mediante seleção, nas tabelas ou nas views respetivas, das operações do tipo "update" ("U") com o atributo "deleted" com valor verdadeiro ("True"/1). Na eventualidade de ocorrer um "undelete", o mesmo será detectável nas operações "update" ("U") com o atributo "deleted" com valor falso ("False"/0) em que o valor anterior da coluna "deleted" era verdadeiro.

De um modo não exaustivo, apresentam-se de seguida alguns exemplos de "queries" que poderão ser executados pelo Auditor:

Exemplo de um "query" para obter dados de auditoria a partir da tabela "LogMedicoes", referentes a operações de "insert", "update" (incluindo os "soft deletes") e "delete" efetuadas na última semana:

```
select * from LogMedicoes where dataHoraMedicao between '2018-
04-08' and '2018-04-15';
```

Exemplo de um "query" para obter dados de auditoria a partir da tabela "LogSelect", relativos a consultas efetuadas na última semana à tabela "Medicoes" e que permitem obter no conteúdo da coluna "comandoSelect" o "query" que foi efetuado por alguém:

```
select * from LogSelect where comandoSelect like '%LogMedicoes%'
and dataOperacao between '2018-04-08' and '2018-04-15';
```

Exemplo de um "query" para obter dados de auditoria a partir da "view" "v_upd_cultura", relativos a operações de "update" efetuadas na última semana à tabela "Cultura":

```
select * from v_upd_cultura where dataOperacao_new between
'2018-04-08' and '2018-04-15';
```

As Views acima referidas têm a seguinte definição:

```
/* VIEW V_Upd_Investigador */
DROP VIEW IF EXISTS V_Upd_Investigador;
CREATE VIEW V_Upd_Investigador AS
SELECT
novo.idLogInvestigador as idLogInvestigador_new,
novo.idInvestigador,
novo.email as email_new,
novo.nomeInvestigador as nomeInvestigador_new,
novo.deleted as deleted_new,
novo.utilizador as utilizador_new,
novo.operacao as operacao_new,
novo.dataOperacao as dataOperacao_new,
antigo.idLogInvestigador as idLogInvestigador_old,
antigo.email as email_old,
antigo.nomeInvestigador as nomeInvestigador_old,
antigo.deleted as deleted_old,
antigo.utilizador as utilizador_old,
antigo.operacao as operacao_old,
antigo.dataOperacao as dataOperacao_old
from LogInvestigador novo, LogInvestigador antigo
where novo.operacao = 'U'
AND antigo.idLogInvestigador = (select max(idlogInvestigador)
from LogInvestigador
where idInvestigador = novo.idInvestigador
and idLogInvestigador < novo.idLogInvestigador);

/* VIEW V_Upd_Cultura */
DROP VIEW IF EXISTS V_Upd_Cultura;
CREATE VIEW V_Upd_Cultura AS
```

```
SELECT novo.idLogCultura as idLogCultura_new,
novo.idCultura,
novo.idInvestigador as idInvestigador_new,
novo.nomeCultura as nomeCultura_new,
novo.limiteInferiorTemperatura as limiteInferiorTemperatura_new,
novo.limiteSuperiorTemperatura as limiteSuperiorTemperatura_new,
novo.limiteInferiorHumidade as limiteInferiorHumidade_new,
novo.limiteSuperiorHumidade as limiteSuperiorHumidade_new,
novo.deleted as deleted_new,
novo.utilizador as utilizador_new,
novo.operacao as operacao_new,
novo.dataOperacao as dataOperacao_new,
antigo.idLogCultura as idLogCultura_old,
antigo.idInvestigador as idInvestigador_old,
antigo.nomeCultura as nomeCultura_old,
antigo.limiteInferiorTemperatura as limiteInferiorTemperatura_old,
antigo.limiteSuperiorTemperatura as limiteSuperiorTemperatura_old,
antigo.limiteInferiorHumidade as limiteInferiorHumidade_old,
antigo.limiteSuperiorHumidade as limiteSuperiorHumidade_old,
antigo.deleted as deleted_old,
antigo.utilizador as utilizador_old,
antigo.operacao as operacao_old,
antigo.dataOperacao as dataOperacao_old
from LogCultura as novo, LogCultura as antigo
where novo.operacao ='U'
AND antigo.idLogCultura =
      (select max(idlogCultura) from LogCultura
      where idCultura = novo.idCultura
      and idLogCultura < novo.idLogCultura);
```

```
/* VIEW V_Upd_Variaveis */
DROP VIEW IF EXISTS V_Upd_Variaveis;
CREATE VIEW V_Upd_Variaveis AS
SELECT
novo.idLogVariaveis as idLogVariaveis_new,
novo.idVariavel,
novo.deleted as deleted_new,
novo.utilizador as utilizador_new,
novo.operacao as operacao_new,
novo.dataOperacao as dataOperacao_new,
antigo.idLogVariaveis as idLogVariaveis_old,
antigo.deleted as deleted_old,
antigo.utilizador as utilizador_old,
antigo.operacao as operacao_old,
antigo.dataOperacao as dataOperacao_old
from LogVariaveis as novo, LogVariaveis as antigo
where novo.operacao ='U'
AND  antigo.idLogVariaveis =(select max(idLogVariaveis)
      from LogVariaveis
      where  idVariavel = novo.idVariavel
      and idLogVariaveis < novo.idLogVariaveis);

/* VIEW V_Upd_VariaveisMedidas */
DROP VIEW IF EXISTS V_Upd_VariaveisMedidas;
CREATE VIEW V_Upd_VariaveisMedidas AS
SELECT
novo.idLogVariaveisMedidas as idLogVariaveisMedidas_new,
novo.idCultura,
novo.idVariavel,
novo.limiteInferior as limiteInferior_new,
```

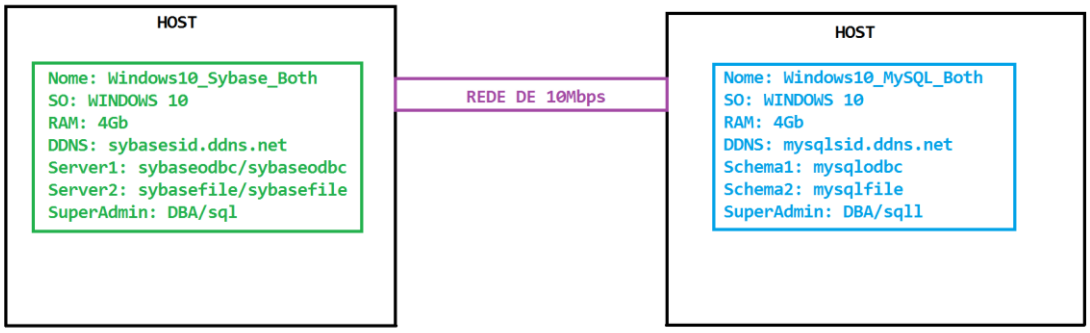
```
novo.limiteSuperior as limiteSuperior_new,
novo.deleted as deleted_new,
novo.utilizador as utilizador_new,
novo.operacao as operacao_new,
novo.dataOperacao as dataOperacao_new,
antigo.idLogVariaveisMedidas as idLogVariaveisMedidas_old,
antigo.limiteInferior as limiteInferior_old,
antigo.limiteSuperior as limiteSuperior_old,
antigo.deleted as deleted_old,
antigo.utilizador as utilizador_old,
antigo.operacao as operacao_old,
antigo.dataOperacao as dataOperacao_old
from LogVariaveisMedidas as novo, LogVariaveisMedidas as antigo
where novo.operacao ='U'
AND  antigo.idLogVariaveisMedidas =
(select max(idLogVariaveisMedidas)
      from   LogVariaveisMedidas
      where  idCultura = novo.idCultura and    idVariavel =
      novo.idVariavel     and    idLogVariaveisMedidas <
      novo.idLogVariaveisMedidas);
```

```
/*VIEW V_Upd_Medicoes */
DROP VIEW IF EXISTS V_Upd_Medicoes;
CREATE VIEW V_Upd_Medicoes AS
SELECT
novo.idLogMedicoes as idLogMedicoes_new,
novo.idCultura,
novo.idVariavel,
novo.numeroMedicao,
novo.dataHoraMedicao as dataHoraMedicao_new,
```

```
novo.deleted as deleted_new,
novo.utilizador as utilizador_new,
novo.operacao as operacao_new,
novo.dataOperacao as dataOperacao_new,
antigo.idLogMedicoes as idLogMedicoes_old,
antigo.dataHoraMedicao as dataHoraMedicao_old,
antigo.deleted as deleted_old,
antigo.utilizador as utilizador_old,
antigo.operacao as operacao_old,
antigo.dataOperacao as dataOperacao_old
from LogMedicoes as novo, LogMedicoes as antigo
where novo.operacao ='U'
AND  antigo.idLogMedicoes =(select max(idLogMedicoes)
      from LogMedicoes
      where idCultura = novo.idCultura
      and idVariavel = novo.idVariavel
      and numeroMedicao = novo.numeroMedicao
      and idLogMedicoes < novo.idLogMedicoes);
```

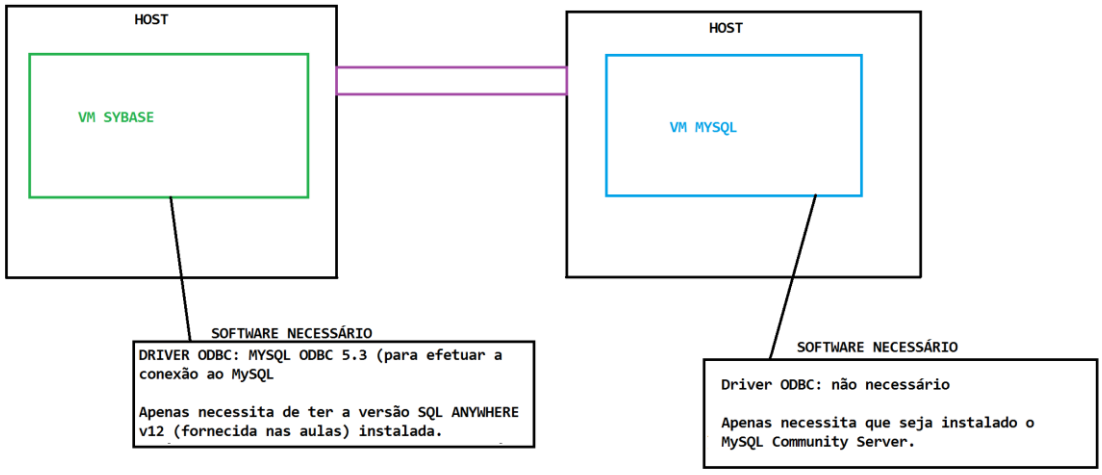
2.13 Utilização de VM's na implementação

Todo a implementação do projeto foi feita a 100% sobre um sistema de Máquinas Virtuais, com o sistema operativo Windows 10. O esquema das máquinas virtuais é mostrado de seguida:



Este sistema permitiu-nos desde muito cedo detetar erros que usando métodos mais fracos, como o XAMMP, não teríamos. Mas, é garantido que quando o sistema fosse transferido para um sistema de produção problemas iriam aparecer, como firewalls, falta de drivers no sitio certo, etc …

De seguida mostramos apenas o software necessário em cada máquina de maneira a poder correr o sistema na perfeição, incluindo os testes:



Outros softwares foram instalados em ambas as máquinas para dar suporte a outros fatores como: notepad++, google chrome, wireshark (usado nos testes), github, no-ip (de maneira a usar DDNS).

Toda a firewall, de ambas as máquinas foi também configurado, regra a regra de maneira a tornar o sistema seguro.

Na máquina VM Sybase, correm, automaticamente quando a sessão é inicia, dois servidores, o sybaseodbc e o sybasefile. Ambos os servidores produzem para a sua base de dados 3000 dados por hora. E a migração ocorre segundo o especificado em cado caso.

Na máquina VM MySQL, corre apenas um servidor, mas com dois schemas. O Schema, mysqlodbc e o schema mysqlfile.

Estes recebem os dados via ODBC quando o servidor sybase assim quiser. E via ficheiro, quando o programador de tarefas inicia o ficheiro bat.

O uso das VM's, verificou-se, para todo o grupo uma mais valia, dando apenas algum trabalho a configurar no inicio, mas não dando quaisquer problemas durante o resto das implementações.

Desta forma garantíamos todos também, com o auxílio do github que os nossos locais de trabalho eram idênticos, o que retira a permissão da tão usada desculpa "mas funciona no meu".