Manual de instalação de conector MySQL v5.7.6 para o intermediário Kafka

Ivo M. F. Silva and Carlos C. Teixeira

DI–FCUL–TR–2017–02

DOI:10.451/31159

(http://hdl.handle.net/10451/31159)

November 2017

Publication without review in the repository of the Department of Informatics
of the University of Lisbon, Faculty of Sciences
(http://repositorio.ul.pt/handle/10451/12254).

Índice

1. Introdução 1
2. Requisitos para instalar o conector 2
3. Como instalar o conector 2
    3.1. Instalação normal 2
    3.2. Casos especiais da instalação 3
4. Descrição do script de instalação 4
    4.1. Operações realizadas pelo script de instalação 4
5. Descrição dos procedimentos MySQL 4
6. Descrição das funções C 5
7. Lista de procedimentos, funções e tabelas auxiliares 7
    7.1. Procedimentos MySQL 7
    7.2. Funções C 7
    7.3. Tabelas 9
8. Configurações 10
    8.1. Descrição da tabela de configuração 10
    8.2. Na tabela de configuração 11
    8.3. No ficheiro de configuração 11
    8.4. Incluídas no código C 12
9. Conclusão 13
10. Bibliografia 14

1. Introdução

Este documento pretende auxiliar a instalação do conector MySQL que permite obter dados a partir do ficheiro de registo de alterações binário, convertê-los para JSON e publicá-los num intermediário Kafka usando o consumidor JSON Kafka-REST-Proxy [13].

O conector foi desenvolvido como parte de um sistema de integração de dados de uma multinacional de telecomunicações no âmbito de um projeto de mestrado do Departamento de Informática da Faculdade de Ciências da Universidade de Lisboa.

2. Requisitos para instalar o conector

1. Permissões de *root* num servidor MySQL pós-v5.7.6 (para versões anteriores ver ponto 6 do próximo capítulo).

2. Acesso ao sistema com um utilizador que pertença ao grupo *mysql* (senão será necessário utilizar permissões de administrador).

3. Os scripts *instalationScript.sql*, *registerTableInCDC.sql* e *unregisterTableInCDC.sql.*

4. O ficheiro *conectorMySQL.c*.

5. Acesso a um consumidor [5] JSON para publicar as alterações detetadas nas tabelas.

3. Como instalar o conector

3.1. Instalação normal

1. Aceder ao servidor MySQL com um utilizador da máquina que pertença ao grupo *mysql* e estabelecer uma ligação à BD com um login que tenha permissões de *root*.

2. Mudar o contexto para a BD onde se pretende instalar o conector (comando *USE*) ou criá-la caso ainda não exista.

3. Executar os três scripts sql pela seguinte ordem para criar os procedimentos com o mesmo nome dos ficheiros:

    instalationScript.sql

    unregisterTableInCDC.sql

    registerTableInCDC.sql

4. Executar o procedimento recém-criado *instalationScript* para concluir a instalação:

    (CALL instalationScript;)

5. Executar os procedimentos *registerTableInCDC* e *unregisterTableInCDC* para respetivamente registar e cancelar o registo das tabelas que o conector deve monitorizar indicando o nome do esquema/base de dados seguido do nome da tabela:

    (CALL registerTableInCDC(schema_name, table_name);)

    (CALL unregisterTableInCDC(schema_name, table_name);)

6. Para verificar se o registo de alterações binário está ativo executar na BD [10]:

    (SHOW BINARY LOGS;)

7. Caso este registo esteja desactivado, seguir o caso especial nº2 presente no final desta secção.

8. Para verificar o formato atual do registo de alterações binário executar:

    (SHOW VARIABLES LIKE 'binlog_format';)

a) Se o formato for STATEMENT e a versão do MySQL for superior à v5.1.5 seguir o caso especial nº3 presente no final desta secção.

b) Se a versão for anterior à v5.1.5 é necessário alterar o conector como descrito no caso especial nº1.c) presente no final desta secção.

9. Após garantir que o formato é 'ROW' executar o comando seguinte para registar todas as alterações [12]:

(SET GLOBAL binlog_row_image=FULL;)

10. Atualizar na tabela de configuração da BD (descrita em secção própria neste documento) os dados do consumidor [5] JSON;

11. Caso o utilizador da máquina não pertença ao grupo *mysql* executar na linha de comandos em UNIX:

(sudo adduser utilizador_atual mysql)

12. Compilar o ficheiro c*onectorMySQL.c* na linha de comandos executando [6]:

(gcc conectorMySQL.c -o conectorMySQL `mysql_config –cflags` `mysql_config –libs`)

13. a) Colocar o ficheiro de configuração fornecido junto ao executável e executar sem parâmetros:

(./conectorMySQL)

13. b) Escrever um ficheiro de configuração (como indicado em secção própria neste documento) e indicar a sua localização e o seu nome com a opção *-f*:

(./conectorMySQL -f ~/novoConfigFile.cfg)

14. Executar o ficheiro recém-criado *conectorMySQL* para começar a capturar as alterações das tabelas registadas (caso o utilizador não tenha as permissões necessárias executar com *sudo)*:

(./conectorMySQL)

3.2. Casos especiais da instalação

1. a) Para suportar as versões MySQL após a v5.6.10 (Fev 2013) é necessário:

- alterar o BIN LOG pré-definido da BD para registar as linhas da tabela alteradas (*row based*) [7];

1. b) Para suportar as versões após a v5.1.5 (Nov 2008) é necessário:

- chamar função getLastLogFilename() na getActualLogFilename() para substituir obtenção do valor da variável global do sgbd "log_bin_basename" que não existia antes da v5.6.10 [8]. Esta alteração pode implicar que se leia o ficheiro de indíce dos ficheiros de registo de alterações binários com *sudo*;

1. c) Para suportar as versões após a v3.23 (Jan 2001) é necessário:

- alterar profundamente o conector para utilizar uma BD auxiliar para aplicar as operações registadas no log (statement based) [9] e então registar as alterações produzidas.

2. a) No Windows, para ativar o registo de alterações binário aceder ao ficheiro de configuração *my.ini* para adicionar ou descomentar a linha (sem os parênteses):

(log-bin[=base_name])

2. b) No Unix, para ativar o registo de alterações binário é necessário aceder ao ficheiro de configuração *my.cnf* e definir o server-id (Bug #11763963, Bug #56739) [1] adicionando as seguintes linhas (sem os parênteses):

([mysqld]

server-id=master-01

log-bin)

3. Caso o formato seja STATEMENT e a versão do MySQL seja superior à v5.1.5 executar [11]:

(AVISO: a alteração do formato do registo de alterações binário afecta o funcionamento dos mecanismos de replicação pré-configurados.)

(SET GLOBAL binlog_format = 'ROW';)

4. Descrição do script de instalação

O script está no ficheiro *instalationScript.sql* e necessita de privilégios de *root* para ser executado.

4.1. Operações realizadas pelo script de instalação

1. Cria a tabela de configuração do conector caso ainda não exista.

2. Elimina todas as linhas que existam na tabela.

3. Insere todos os valores pré-definidos (descritos em secção própria neste documento).

4. Cria a tabela auxiliar que o conector utiliza para conhecer quais as tabelas onde deve capturar alterações.

5. Descrição dos procedimentos MySQL

O *instalationScript()* instala o conector criando ou restabelecendo as tabelas existentes para os valores pré-definidos.

O *registerTableInCDC(schema_name, table_name)* regista uma tabela no conector para este capturar as suas alterações.

O un*registerTableInCDC(schema_name, table_name)* cancela o registo de uma tabela no conector.

6. Descrição das funções C

A addCommaToJSON adiciona vírgulas a strings JSON quando são necessárias para adicionar novos elementos.

A checkAlterationsTimeout verifica o estado do timeout para obtenção de alterações do ficheiro de log binário em utilização pelo SGBD.

A convertTimestamp converte o número de segundos desde 1970 para uma data.

A convertToJSONdata converte os dados das colunas para JSON.

A convertToJSONtype converte os tipos MySQL para os tipos conhecidos pelo consumidor [5] JSON.

A divideRow divide uma linha do ficheiro de log binário nos seus elementos.

A divideSchemaTable divide uma string em duas schema e table separados por ".".

A flushBinLog pede ao SGBD para fechar o ficheiro de log binário em utilização.

A formatTimestamp formata os timestamp para o consumidor [5] JSON.

A freeHashMapContents liberta a memória alocada para o hash table [3].

A getActualLogFilename obtém o nome do ficheiro de log atual.

A getAuxRegistryTableName obtém o nome da tabela auxiliar onde o conector regista as tabelas onde se capturam alterações.

A getBetweenStrings extrai um segmento de uma string delimitado por duas strings fornecidas.

A getBinLogIndexFilename obtém o nome do ficheiro com a lista dos ficheiros de log binários.

A getColName obtém o nome de uma coluna dando o seu número ordinal numa tabela.

A getColNumber obtém o número ordinal de uma coluna a partir de uma linha do ficheiro de log.

A getConsumerIP obtém da tabela de configuração o endereço IP do consumidor [5] JSON.

A getConsumerPort obtém da tabela de configuração o porto do consumidor [5] JSON.

A getConsumerTopic obtém da tabela de configuração o tópico do consumidor [5] JSON onde o conector deve publicar as alterações.

A getContentType obtém da tabela de configuração o content-type que deve ser utilizado no POST para publicar o JSON.

A getData obtém os dados alterados de uma coluna da tabela.

A getFilenameIndex obtém o número de um ficheiro de log a partir do seu nome.

A getLastClosedLog obtém o nome do último log fechado.

A getLastClosedProcLogFile obtém o nome do último ficheiro processado a partir de um ficheiro de log fechado.

A getLastLogFilename obtém o nome do último ficheiro de log a partir do índice de ficheiros.

A getLastPosition obtém da tabela de configuração a última posição do log publicada.

A getLastTimestamp obtém da tabela de configuração o timestamp das últimas alterações publicadas.

A getLogType obtém o tipo de uma coluna a partir de uma linha do ficheiro de log.

A getOpenLogFilename obtém o nome do ficheiro de log binário em utilização pelo SGBD.

A getOpenProcLogFile obtém o nome do ficheiro processado a partir do ficheiro de log em utilização.

A getOperations obtém o resumo das operações realizadas numa tabela a partir da hash table [3].

A getRowSeqNumber obtém o número de sequência das linhas alteradas de uma tabela a partir da hash table [3].

A getSchemaName obtém o schema de uma tabela a partir da tabela auxiliar de registo do conector.

A getSlashDelimiterIndex devolve o índice da primeira ocorrência de um delimitador numa string.

A getSleepTimeout devolve o timeout entre duas capturas de alterações.

A getTableCount devolve o número de tabelas registadas na tabela auxiliar de registo do conector.

A getTableName obtém o nome de uma tabela a partir da tabela auxiliar de registo do conector.

A getTablesAlterations obtém da hash table [3] todas as alterações de uma tabela agrupadas em JSON.

A getUUID devolve um UUID produzido pelo SGBD [4].

A insertString insere uma string numa posição indicada de outra string.

A main inicializa o conector e captura de alterações periodicamente.

A mountRowIdHeader contrói o início de cada linha alterada em JSON.

A processBinLogFile converte um ficheiro de log binário fechado num ficheiro de texto utilizando a ferramenta mysqlbinlog [2].

A processOpenBinLogFile converte um ficheiro de log binário em utilização pelo SGBD num ficheiro de texto utilizando a ferramenta mysqlbinlog [2].

A readConfigFile lê o ficheiro de configuração do conector.

A readLastClosedLogFile lê o último ficheiro convertido a partir de um ficheiro de log fechado.

A readOpenBinLogFile lê o ficheiro convertido a partir do ficheiro de log binário em utilização.

A registerTableAlterations adiciona na hash table [3] uma linha alterada de uma tabela.

A removePrimeSymbols extrai dados entre aspas.

A replaceStr dada uma string substitui uma substring num determinado índice por outra.

A searchForAlterationsInOpenLogFile pesquisa alterações no ficheiro convertido a partir de um log em utilização.

A setAlterationsTimeout regista na tabela de configuração do conector o timestamp da primeira deteção de alterações relevantes do ficheiro de log em utilização.

A setOperations adiciona ao resumo de operações de uma tabela mantido na hash table [3] a operação realizada por uma linha alterada.

A setRowSeqNumber atualiza na hash table [3] o número de sequência das linhas alteradas numa tabela.

A setTablesAlterations adiciona uma linha alterada às alterações agrupadas por tabela mantidas na hash table [3] em formato JSON.

A shiftLeft desloca para a esquerda um determinado número de caracteres num índice indicado.

A unsetAlteratonsTimeout apaga da tabela de configuração o timestamp da primeira deteção de alterações relevantes no ficheiro de log em utilização.

A updateLogPosition atualiza na tabela de configuração a última posição do log publicada.

A updateTimestamp atualiza na tabela de configuração o timestamp da última alteração publicada.

A waitForAlterations inicia timeout até à próxima captura de alterações.

7. Lista de procedimentos, funções e tabelas auxiliares

7.1. Procedimentos MySQL

- instalationScript ;
- registerTableInCDC ;
- unregisterTableInCDC ;

7.2. Funções C

- addCommaToJSON;
- checkAlterationsTimeout;
- convertTimestamp;
- convertToJSONdata;
- convertToJSONtype;

- divideRow;
- divideSchemaTable;
- flushBinLog;
- formatTimestamp;
- freeHashMapContents;
- getActualLogFilename;
- getAuxRegistryTableName;
- getBetweenStrings;
- getBinLogIndexFilename;
- getColName;
- getColNumber;
- getConsumerIP;
- getConsumerPort;
- getConsumerTopic;
- getContentType;
- getData;
- getFilenameIndex;
- getLastClosedLog;
- getLastClosedProcLogFile;
- getLastLogFilename;
- getLastPosition;
- getLastTimestamp;
- getLogType;
- getOpenLogFilename;
- getOpenProcLogFile;
- getOperations;
- getRowSeqNumber;
- getSchemaName;
- getSlashDelimiterIndex;
- getSleepTimeout;

- getTableCount;
- getTableName;
- getTablesAlterations;
- getUUID;
- insertString;
- main;
- mountRowIdHeader;
- processBinLogFile;
- processOpenBinLogFile;
- readConfigFile;
- readLastClosedLogFile;
- readOpenBinLogFile;
- registerTableAlterations;
- removePrimeSymbols;
- replaceStr;
- searchForAlterationsInOpenLogFile;
- setAlterationsTimeout;
- setOperations;
- setRowSeqNumber;
- setTablesAlterations;
- shiftLeft;
- unsetAlteratonsTimeout;
- updateLogPosition;
- updateTimestamp;
- waitForAlterations;
-

7.3. Tabelas

- cdcConectorConfig: tabela de configuração do conector;
- cdcConectorTables: tabela de registo das tabelas cujas alterações o conector deve considerar.

8. Configurações

8.1. Descrição da tabela de configuração

Esta tabela organiza-se em duas colunas *configKey* e *configValue* com uma configuração por linha:

- *cdcTablesRegistry* é o nome da tabela auxiliar de registo de tabelas;
- *lastSentLogPosition* é o número da última posição do registo de alterações binário enviada para o consumidor [5] JSON;
- *lastSentTimestamp* é o timestamp da última alteração do registo de alterações binário enviada para consumidor [5] JSON;
- *lastAlterationsDetection* é o timestamp da última deteção de alterações relevantes no ficheiro do registo de alterações binário ainda em utilização pelo SGBD;
- *lastAlterationsTimeoutSeconds* é o timeout para fechar o ficheiro de registo de alterações binário em utilização após a primeira deteção de alterações relevantes;
- *captureAlterationsTimeoutSeconds* é o tempo entre duas pesquisas por alterações nos ficheiros de registo de alterações binários;
- *consumerJSONip* é o endereço IP do consumidor [5] JSON;
- *consumerJSONport* é o porto do consumidor [5] JSON;
- *consumerJSONtopic* é o tópico do consumidor [5] JSON onde se publicam as alterações;
- *consumerJSONcontent-type* é o content-type utilizado no método POST para enviar as alterações para o consumidor [5] JSON;

Todos os parâmetros presentes na tabela de configuração *ConfigCDCTable* podem ser alterados nas definições das variáveis com o mesmo nome no início do script *instalationScript.sql* ou diretamente na BD.

8.2. Na tabela de configuração

Estes são os valores pré-definidos criados pelo script de instalação:

configKey	***configValue***
cdcTablesRegistry	“cdcConectorTables”
lastSentLogPosition	“0”
lastSentTimestamp	“01-01-1970 00:00:01”
lastAlterationsDetection	“01-01-1970 00:00:01”
lastAlterationsTimeoutSeconds	“300”
captureAlterationsTimeoutSeconds	“60”
consumerJSONip	“10.64.104.4”
consumerJSONport	“8082”
consumerJSONtopic	“IVO”
consumerJSONcontent-type	“application/vnd.kafka.json.v1+json”

8.3. No ficheiro de configuração

Apenas os utilizadores que pertençam ao grupo *mysql* deverão ter permissões de leitura.

O ficheiro de configuração:

- deve ter 4 linhas que indicam respetivamente: server, login, password e schema/database;
- não pode ter linhas vazias no início nem entre as configurações;
- as linhas não podem ter espaços no início nem no fim;
- um exemplo de ficheiro pode ser:

  “localhost

  root

  pass

  learningcdc

  “

8.4. Incluídas no código C

O nome da tabela de configuração *cdcConectorConfig* pode ser alterado pesquisando por esta string no ficheiro *conectorMySQL.c* e substituindo as suas ocorrências pelo novo nome da tabela de configuração.

O nome das *configKey* da tabela de configuração só pode ser alterado pesquisando pelas atuais no ficheiro *conectorMySQL.c* e substituindo todas as suas ocorrências.

9. Conclusão

Seguindo os passos de instalação descritos neste documento é possível colocar o conector MySQL em funcionamento numa base de dados em operação acedendo ao ficheiro de registo de alterações binário para detetar alterações com um impacto reduzido no desempenho da mesma.

Com a instalação deste conector é possível integrar e persistir dados de sistemas em operação publicando-os num intermediário Kafka que depois pode alimentar subscritores com requisitos muito distintos, permitindo inclusive analisar posteriormente os dados recolhidos.

10. Bibliografia

1. MySQL :: MySQL 5.7 Reference Manual :: 16.1.6.4 Binary Logging Options and Variables, https://dev.mysql.com/doc/refman/5.7/en/replication-options-binary-log.html#option_mysqld_log-bin
2. MySQL :: MySQL 5.7 Reference Manual :: 4.6.7 mysqlbinlog — Utility for Processing Binary Log Files, https://dev.mysql.com/doc/refman/5.7/en/mysqlbinlog.html
3. hcreate(3) - OpenBSD manual pages, http://man.openbsd.org/hcreate.3
4. MySQL :: MySQL 5.7 Reference Manual :: 12.20 Miscellaneous Functions, https://dev.mysql.com/doc/refman/5.7/en/miscellaneous-functions.html#function_uuid
5. Kafka REST Proxy — Confluent Platform 3.2.2 documentation, http://docs.confluent.io/current/kafka-rest/docs/intro.html
6. MySQL :: MySQL 5.7 Reference Manual :: 27.8.4.1 Building C API Client Programs, https://dev.mysql.com/doc/refman/5.7/en/c-api-building-clients.html
7. MySQL :: MySQL 5.7 Reference Manual :: 5.4.4.2 Setting The Binary Log Format, https://dev.mysql.com/doc/refman/5.7/en/binary-log-setting.html
8. MySQL :: MySQL 5.6 Reference Manual :: 17.1.4.4 Binary Log Options and Variables, https://dev.mysql.com/doc/refman/5.6/en/replication-options-binary-log.html#sysvar_log_bin_basename
9. MySQL :: MySQL Internals Manual :: 20.10 Row-Based Binary Logging, https://dev.mysql.com/doc/internals/en/row-based-binary-logging.html
10. MySQL :: MySQL 5.7 Reference Manual :: 13.7.5.1 SHOW BINARY LOGS Syntax, https://dev.mysql.com/doc/refman/5.7/en/show-binary-logs.html
11. MySQL :: MySQL 5.7 Reference Manual :: 5.4.4.2 Setting The Binary Log Format, https://dev.mysql.com/doc/refman/5.7/en/binary-log-setting.html
12. MySQL :: MySQL 5.7 Reference Manual :: 16.1.6.4 Binary Logging Options and Variables, https://dev.mysql.com/doc/refman/5.7/en/replication-options-binary-log.html#sysvar_binlog_row_image
13. GitHub - confluentinc/kafka-rest: Confluent REST Proxy for Kafka, https://github.com/confluentinc/kafka-rest