八木沼 幸一

株式会社ミガロ.

システム事業部 プロジェクト推進室

[Delphi/400]
IBM i トリガー機能を活かしたセキュリティログ対応

略歴
1969 年 2 月 22 日生まれ
1991 年 獨協大学 経済学部卒業
2005 年 7 月 株式会社ミガロ. 入社
2005 年 7 月 システム事業部配属

現在の仕事内容
Delphi/400、SmartPad4i（JC/400）を利用したシステムの受託開発の内、RPG 開発をメインに担当し、HA ツールである MAXAVA の設定やサポート業務も担当している。

1.はじめに

トリガー機能というと、Microsoft SQL Server や Oracle Database などのオープン系データベースを思い浮かべる方も多いと思うが、IBM i でもトリガー機能を使用してさまざまな処理を実行できる。

更新ログを登録したり、明細登録した情報を自動集計したり、ログ出力時のタイムスタンプを保持したりと、使い方次第では非常に便利である。

さらに IBM i では更新トリガーだけではなく、レコードごとの読み取り（READ）トリガーも可能なので、セキュリティの観点からも有用である。

本稿では、トリガー機能の中でも多く用いられる、データの変更を監視するための更新ログファイルを出力する機能と Delphi/400 からの利用例について記述する。

2.トリガー機能の概要

2-1. トリガー機能とは

トリガー機能とは、IBM i のリレーショナルデータベースが備える一般的な機能の 1 つで、データベースファイルに、INSERT/UPDATE/DELETE/READ などのアクセスがあった際に、事前定義しておいたプログラムを自動的に実行できる。

2-2. トリガー機能のメリット

トリガー機能には、次のようなメリットがある。

・アプリケーション開発の効率化

トリガーはデータベースに保管されるため、アプリケーションごとにトリガーに相当するプログラミングを実装する必要がない。

・保守の簡素化

仕様や方針が変更になった場合でも、各アプリケーションプログラムではなく、該当するトリガープログラムを変更するだけで、共通で動作変更に対応できる。

2-3. トリガー機能使用時の考慮点

トリガー機能は便利な機能ではあるが、使用時に注意すべき点がいくつかある。

トリガーを一度設定すると、原理的に毎回実行されてしまうので、パフォーマンスへの影響を考慮する必要がある。たとえばファイル更新時にトリガーを設定している場合（*READ 以外）、入力画面プログラムや照会画面プログラムなどの少量のデータ更新では影響はほとんどないが、バッチ処理など大量にデータを扱う場合は、トリガー機能を一時的に無効にするなどの措置を実施すべきである。

トリガーを変更する際は、コマンド（CHGPFTRG）で一時的にトリガーを無効化できる。通常バッチ処理で、複数のトリガーを設定済みのファイルを扱う場合、トリガーを一括で無効にする『無

080_091_migaro_Report_八木沼_四.indd 80 17/10/25 14:18

図1

コマンド1

```
ADDPFTRG FILE(TRGLIB/SHAINP)
TRGTIME(*AFTER) TRGEVENT(*UPDATE)
PGM(TRGLIB/TRG_PGM1)
```

Delphi/400からの実行例

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  AS4001.RemoteCmd('ADDPFTRG FILE(TRGLIB/SHAINP) TRGTIME(*AFTER) TRGEVENT(*UPDATE) PGM(TRGLIB/TRG_PGM1)');
end;
```


080_091_migaro_Report_八木沼_四.indd 81 17/10/25 14:18

効化 CL』と、トリガーを一括で有効にする『有効化 CL』を作成しておき、バッチ処理の先頭で『無効化 CL』を起動して無効化し、バッチ処理本体が終了したあとに『有効化 CL』を起動して有効化するなどの工夫が必要になる場合もある。これは、Delphi/400 からデータを操作する場合も同様である。

2-4. トリガーのタイミング

トリガーの発行されるタイミングについては、対象ファイルに対してのトリガー設定方法で指定できる（詳しい設定方法は後述）。更新の前（Before）／後（After）が指定でき、更新前であれば何らかのチェック、更新後であればログファイルの登録などの用途に使われることが多い。

ちなみにトリガー機能は、Microsoft SQL Server では Before トリガーはなく After トリガーのみをサポートするなど、データベースの種類によって多少異なる。

2-5. その他の機能

セキュリティログを登録する目的でトリガー機能を使用する場合、ログファイルに対するトリガープログラムを一度作成し、対象ファイルに対してトリガーを設定してしまえば、そのあとは意識することなく、ログ出力を自動的に実行できる。

またファイルに対する更新処理だけではなく、レコードごとの READ が行われたタイミングでトリガーを起動することもできる。いわゆる、READ トリガーである。

これは、レコード単位にどのような情報を読み取ったかを把握できるので、監査目的にも使用できる。また、「ある特別のレコードだけをある特定のユーザーのみにしか見せない」といったセキュリティ目的にも使用できる。

Delphi/400 と連携してセキュリティログなどを出力する場合のメリットとして、通信回数の削減効果が期待できる。

Delphi 側のプログラムでセキュリティログの登録処理を行う場合、トランザクションファイルやマスタファイルのような本番データ 1 件の更新処理とログデータ 2 件（変更前後のイメージを持つ場合）の更新となるので、計 3 回の通信に分離される。しかしトリガー機能を使用すれば 1 回の通信のみで実行できるので、パフォーマンスの向上にも役立つ。

3.対象ファイルに対するトリガー設定

トリガー機能を使用する際には、対象となるファイルに対してトリガーを設定する必要がある。

トリガーの「追加」「除去」「有効／無効化」「定義の確認」は、IBM i のコマンドラインから各コマンド発行で実行できる。

トリガーを設定することで、RPG などのプログラムでファイルにアクセスした場合はもちろん、Delphi/400 から直接そのファイルにアクセスした場合もトリガーの効力が及ぶので非常に便利である。

物理ファイルにトリガーが追加されると、指定されたそのファイルの全メンバーおよび従属する論理ファイルが、トリガーの影響を受けることになる。

指定されたそのファイルにより、メンバーに対して変更操作が行なわれると、トリガープログラムが呼び出される。また物理ファイルに従属している論理ファイルに対して変更操作が行なわれた時にも、トリガープログラムが呼び出されるので注意が必要である。

3-1. トリガーの追加（ADDPFTRG）

対象ファイルに対するトリガーの追加は、「物理ファイルトリガーの追加（ADDPFTRG）」コマンドで設定できる。

1 つのデータベースファイルには、最大 300 までのトリガーを関連付けられ、それぞれの挿入、削除、または更新操作時の前後に複数のトリガーを呼び出せる。

またそれぞれの読み取り操作の場合は、その操作が行われたあとに複数のトリガーを呼び出せる。

それでは、具体的にトリガーの追加手順を解説する。【図 1】

(1) コマンド ADDPFTRG を入力

コマンド ADDPFTRG を入力し、F4 キーを実行する。

(2) 物理ファイル／ライブラリ パラメータの指定

トリガーを設定するファイル／ライブラリを指定する。

(3) トリガー時間（TRGTIME）パラメータの指定

*BEFORE および *AFTER のどちらか 1 つを指定する。

*BEFORE を指定した場合、トリガープログラムはファイルに対する変更操作の前に実行される。*AFTER を指定した場合、トリガープログラムはファイルに対する変更操作のあとに実行される。

(4) トリガーイベント（TRGEVENT）パラメータの指定

*INSERT、*DELETE、*UPDATE、*READ のうちの 1 つを指定する。

*INSERT を指定した場合、ファイルに対してのレコード挿入操作でトリガープログラムが実行される。*DELETE を指定した場合、ファイルに対してのレコード削除操作でトリガープログラムが実行される。*UPDATE を指定した場合、ファイルに対してのレコード更新操作でトリガープログラムが実行される。*READ を指定した場合、ファイルに対しての読み取り操作でトリガープログラムが実行される。

(5) プログラム（PGM）パラメータの指定

対象のファイルに対して操作が行われた際に、起動するプログラム名を設定する。

(6) トリガーの置き換え（RPLTRG）パラメータの指定

*NO および *YES のどちらか 1 つを指定する（デフォルト値は *NO である）。これは、同一のトリガー時間（TRGTIME）パラメータおよびトリガーイベント（TRGEVENT）パラメータを持つトリガーがすでに設定されていた場合に、今回のコマンドの内容で置き換えるかどうかを設定できる。

*NO を指定した場合は置き換えられない（既存のトリガー内容を優先する）。*YES を指定した場合は、既存のトリガーが置き換えられる。

(7) トリガー（TRG）パラメータの指

080_091_migaro_Report_八木沼_四.indd 82 17/10/25 14:18

図2

コマンド2

```
CHGPFTRG FILE(TRGLIB/SHAINP) TRG(*ALL)
TRGLIB(*ALL) STATE(*DISABLED)
```

Delphi/400からの実行例

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  AS4001.RemoteCmd('CHGPFTRG FILE(TRGLIB/SHAINP) TRG(*ALL) TRGLIB(*ALL) STATE(*DISABLED)');
end;
```


080_091_migaro_Report_八木沼_四.indd 83 17/10/25 14:18

定

*GEN および任意のトリガー名を指定する（デフォルト値は *GEN である）。

*GEN を指定した場合、システムがトリガー名を自動生成してくれるが、わかりやすい名前を付けておいたほうが、このあと便利である。

トリガーの有効／無効化（CHGPFTRG）やトリガーの除去（RMVPFTRG）を行う際に、すべて（*ALL）という選択肢も当然あるが、特定のトリガーだけに効力を効かせたいといった要望が出てくる可能性も充分考えられる。

その場合、トリガー名をコマンド内に明記する必要があるので、任意のトリガー名を付けておくようお勧めする。システムで自動生成されたトリガー名は DSPFD で調べる必要があり、名前も長くわかりにくい場合が多い。そこで任意の名前を付けておけば、調べる手間が省ける。

Delphi/400 からこれらを操作する場合は、【コマンド 1】の内容を TAS400 コンポーネントの RemoteCmd メソッドで実行するか、もしくは CL を作成して TCall400 から実行することでトリガーが追加できる。

3-2. トリガーの有効／無効化（CHGPFTRG）

対象となるファイルに設定済みのトリガー（1 つまたはすべて）の状態を変更する際には、CHGPFTRG コマンドを使用する。このコマンドは、一時的にトリガー機能を停止したい場合などによく使用される。

バッチ処理などの大量データを扱い、しかもその更新ログが必要ない場合は、一時的にトリガーを無効化し、バッチ処理後に再度有効化するといったように使用する。

Delphi/400 からトリガーの有効／無効を切り替える場合、有効化 CL および無効化 CL を作成しておき、それぞれの CL を呼び分けることで実現できる。

また CL を作成せず、Delphi/400 から直接上記の IBM i コマンドを発行するのも有効である。

それでは、具体的にトリガーの変更手順を解説する。【図 2】

（1）コマンド CHGPFTRG を入力

コマンド CHGPFTRG を入力し、F4 キーを実行する。

（2）物理ファイル／ライブラリ パラメータの指定

トリガーを設定済みのファイル／ライブラリを指定する。

（3）トリガー（TRG）パラメータの指定

*ALL もしくは任意のトリガー名を指定する。

*ALL を指定した場合、対象ファイルに設定されているすべてのトリガーが変更される。任意のトリガー名を指定した場合は、そのトリガーのみが変更される。

（4）トリガー状態（STATE）パラメータの指定

*SAME、*ENABLED、*DISABLED のうちの 1 つを指定する。

*SAME を指定した場合、値は変更されない。*ENABLED を指定した場合、指定されたトリガーが有効になる。*DISABLED を指定した場合、指定されたトリガーが無効になる。

Delphi/400 から操作する場合は、前述した「追加」と同様に、【コマンド 2】の内容を実行するか、CL を作成して実行することでトリガーを変更できる。

有効化の場合は、STATE パラメータを *ENABLED にする。

3-3. トリガーの除去（RMVPFTRG）

トリガー設定済みのファイルからトリガーを除去する際には、「物理ファイルトリガーの除去（RMVPFTRG）」コマンドを使用できる。

除去するトリガーは、トリガー時間（TRGTIME）、トリガーイベント（TRGEVENT）、トリガー名（TRG）の指定で個別に除去できるが、それぞれの項目に *ALL を指定することで、一括除去も可能である。

それでは、具体的にトリガーの除去手順を解説する。【図 3】

（1）コマンド RMVPFTRG を入力

コマンド RMVPFTRG を入力し、F4 キーを実行する。

（2）物理ファイル／ライブラリ パラメータの指定

トリガーを設定済みのファイル／ライブラリを指定する。

（3）トリガー時間（TRGTIME）、トリガーイベント（TRGEVENT）トリガー（TRG）の指定

上記パラメータはトリガーの追加（ADDPFTRG）の指定方法と同様だが、*ALL の指定も可能である。

Delphi/400 から操作する場合は、追加の場合と同様に、【コマンド 3】の内容を実行するか、CL を作成して実行することでトリガーを除去できる。

3-4. トリガー定義の確認（DSPFD）

対象ファイルに対するトリガーの状態や設定数などの現状内容を確認する場合は、ファイル記述表示（DSPFD）コマンドを実行する。

【図 4】は DSPFD コマンドで表示されるトリガー情報部分であるが、実際には DSPFD の内容の 4 ページ目以降に表示される。これによりトリガーの名前、状態、イベント、時刻、プログラム名などの情報がすべて確認できる。

4.トリガーを活かした更新ログ作成

ここでは、社員マスタ（SHAINP）への更新内容を社員マスタログファイル（SHAINPR）に出力する方法を例に説明する。

まず、更新対象ファイルの社員マスタ（SHAINP）は、【DDS1】のとおりにする。

次に、ログを出力する社員マスタログファイル（SHAINPR）を作成する。このファイルには社員マスタ（SHAINP）と同一の項目を持ち（項目 ID も同じ）、さらに【DDS2】のようにログ情報を保持する項目も追加している。

この項目の追加により、社員マスタの変更情報プラス、ログの出力日時や更新ユーザー、トリガー事象などの情報も把握できる。

続いて、社員マスタ（SHAINP）に対してトリガー設定を行う。今回は、データ更新（*UPDATE）後に、変更後の情報をログファイルに出力するよう設定す

080_091_migaro_Report_八木沼_四.indd 84 17/10/25 14:18

図3

コマンド3

```
RMVPFTRG FILE(TRGLIB/SHAINP) TRGTIME(*ALL)
TRGEVENT(*ALL) TRG(*ALL)
```

Delphi/400からの実行例

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  AS4001.RemoteCmd('RMVPFTRG FILE(TRGLIB/SHAINP) TRGTIME(*ALL) TRGEVENT(*ALL) TRG(*ALL)');
end;
```


080_091_migaro_Report_八木沼_四.indd 85 17/10/25 14:18

る。トリガープログラムは、このあとで作成するトリガープログラム（TRG_PGM1）を【図5】のように指定する。

最後に、TRG_PGM1をRPGで作成する例を示す（【ソース1】～【ソース5】参照）。

このRPGプログラムはIBM i側で設定しておくだけだが、対象ファイルをDelphi/400などのアプリケーションで操作した場合には、このトリガー機能が自動的に実行される。

【ソース1】～【ソース5】のRPGプログラムの概要は、次のとおりである。

F仕様書には、社員マスタログファイル（SHAINPR）のみを出力モードで定義する。社員マスタ（SHAINP）の変更前後の情報は、パラメータで受け取る。トリガープログラムのパラメータとしては、トリガーバッファとそのバッファの長さの2つを受け取る（受け取りパラメータの詳細は、【ソース1】～【ソース5】を参照）。

社員マスタ（SHAINP）の変更前後の情報をパラメータから取得するには、ポインターを使用してプログラム内の構造体に取り込む。

まず、変更前の情報を取り込む方法を解説する。

【ソース1】で構造体OLDRECを定義する際、EXTNAMEに社員マスタ（SHAINP）指定することで、構造体OLDRECのレイアウトと社員マスタのレイアウトを同期させる。

さらにBASEDで、基底ポインター名としてOLDRPを指定しておく。その上で、【ソース4】の81.00ステップで、%ADDR命令を使用してトリガーバッファ（PARM1）内の元レコード位置のポインターを基底ポインターのOLDRPにセットすることで、構造体OLDRECに変更前の社員マスタのレコード情報がセットされる。

同じ要領で、変更後の情報も取得できる。

そして【ソース4】【ソース5】で、ログ情報（ログ出力日時、端末ID、ユーザー名、トリガー事象）を付加し、社員マスタログファイル（SHAINPR）を出力（WRITE）する。

【ソース1】で、構造体OLDREC定義時にPREFIX（O）を指定しているので、構造体OLDRECの項目名の先頭には、”O”が付加されている。そのため、社員マスタログファイル（SHAINPR）の項目名とは一致していない。

逆に構造体NEWREC定義時にはPREFIXを指定していないので、社員マスタログファイル（SHAINPR）の項目名と一致している。その結果、社員マスタログファイル（SHAINPR）へのWRITE命令によって変更後の情報が出力される。

変更前の情報をログファイルに出力したい場合は、PREFIXの定義を逆転させることで実現できる。

5.まとめ

IBM iでシステムを構築する際にトリガー機能を取り入れていない環境も多いが、使い方次第ではDelphi/400側のプログラムも省略・共通化でき、開発面でも非常に有用な機能と言える。

本稿ではセキュリティログを出力する機能を例にしたが、データ更新前のチェック機能や登録情報の自動集計機能などにも有効なので、IBM iの運用に取り入れてぜひ活用いただきたい。

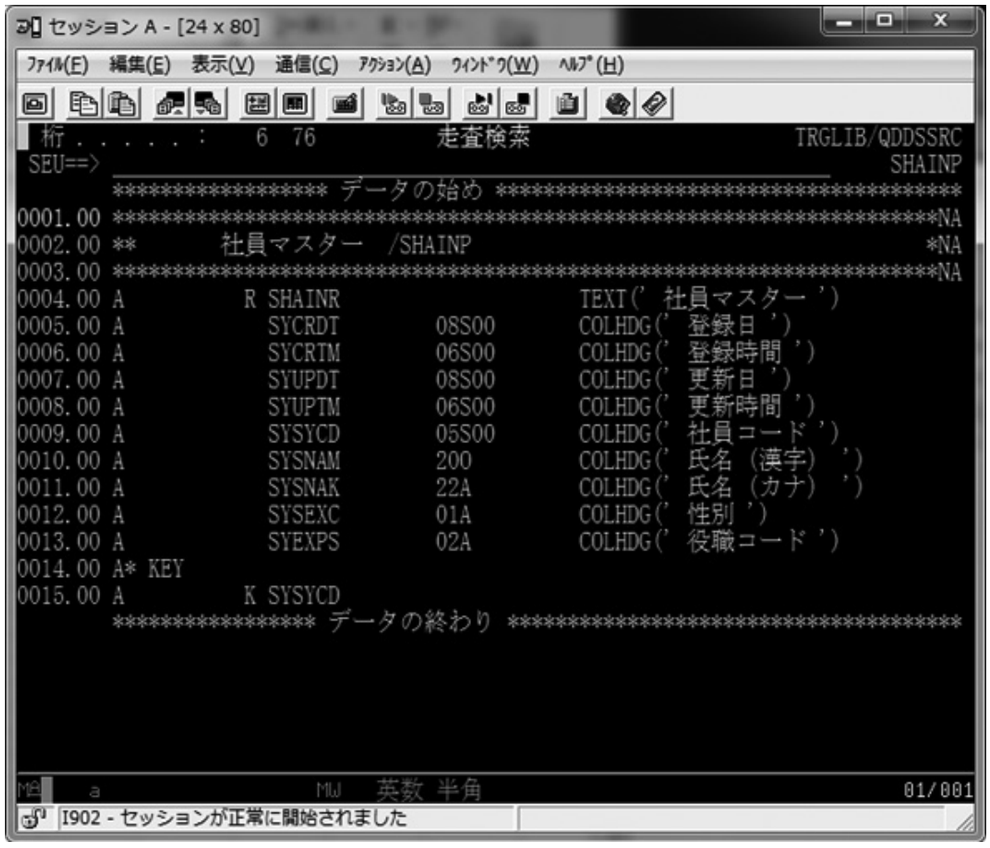
M

080_091_migaro_Report_八木沼_四.indd 86 17/10/25 14:18

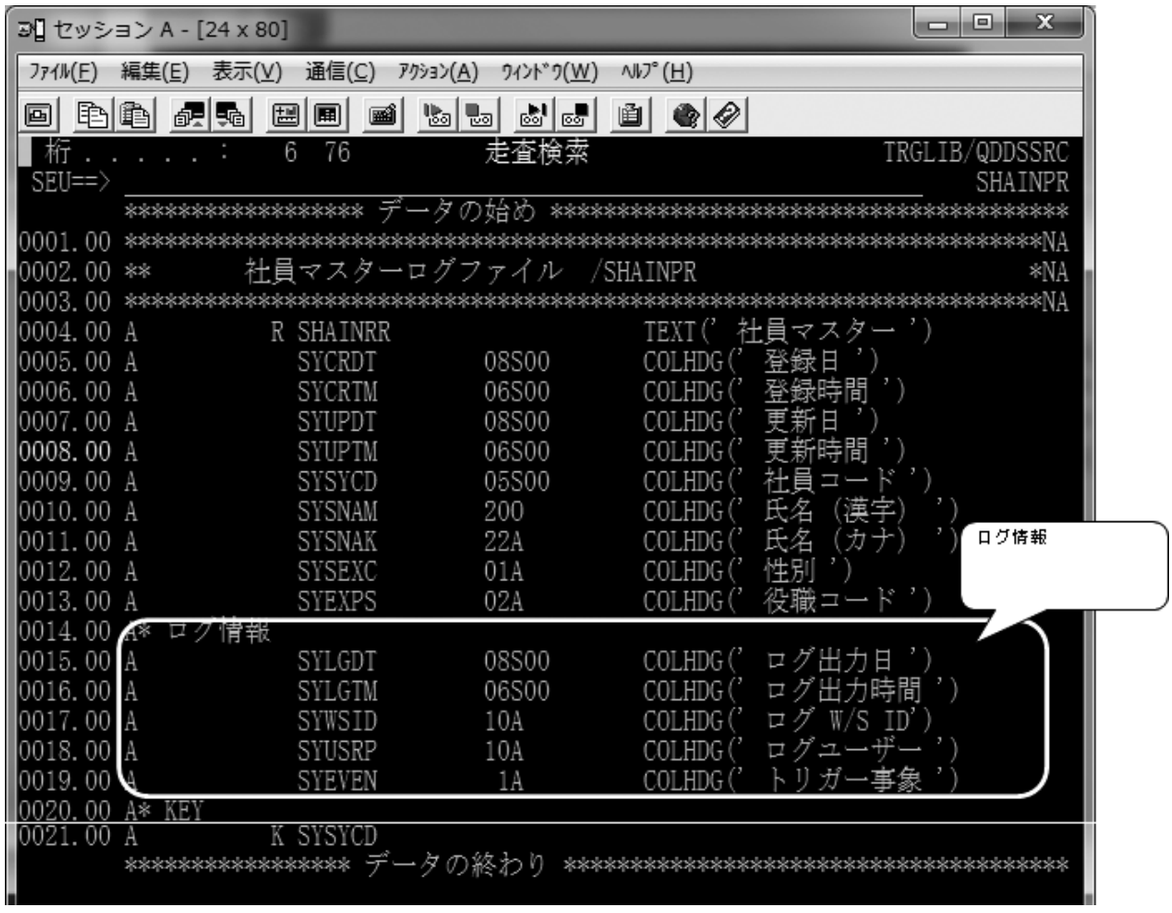
図4

080_091_migaro_Report_八木沼_四.indd 87 17/10/25 14:18

DDS1

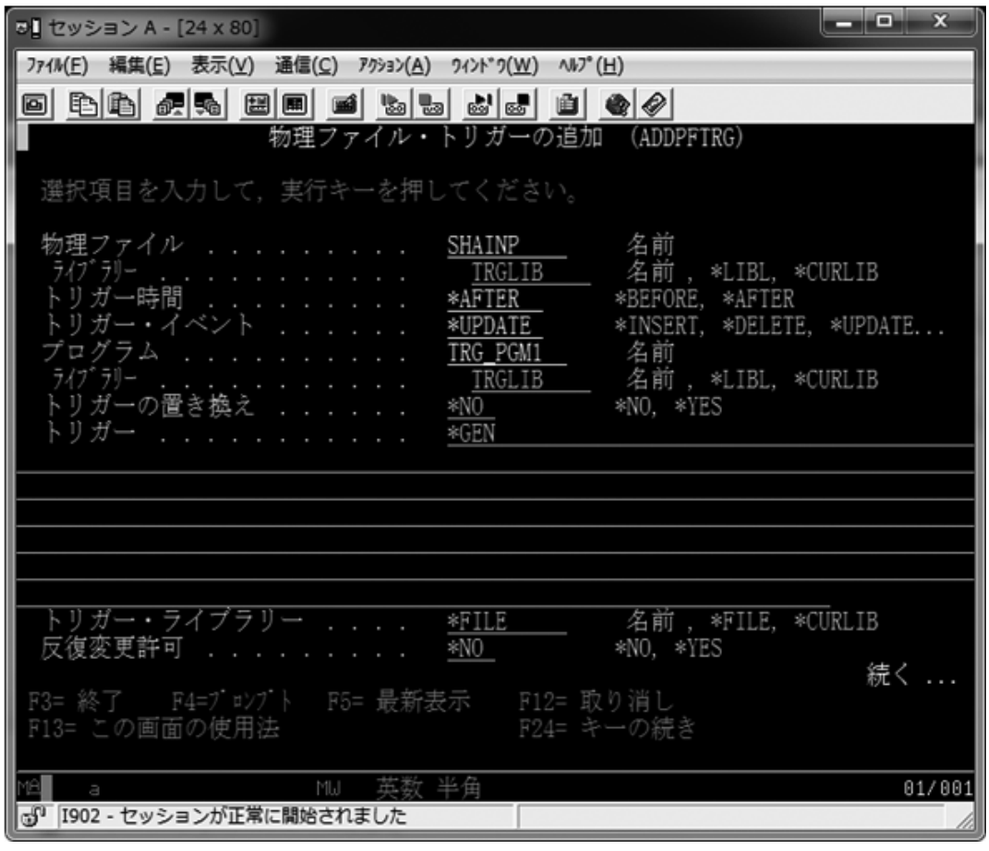


DDS2

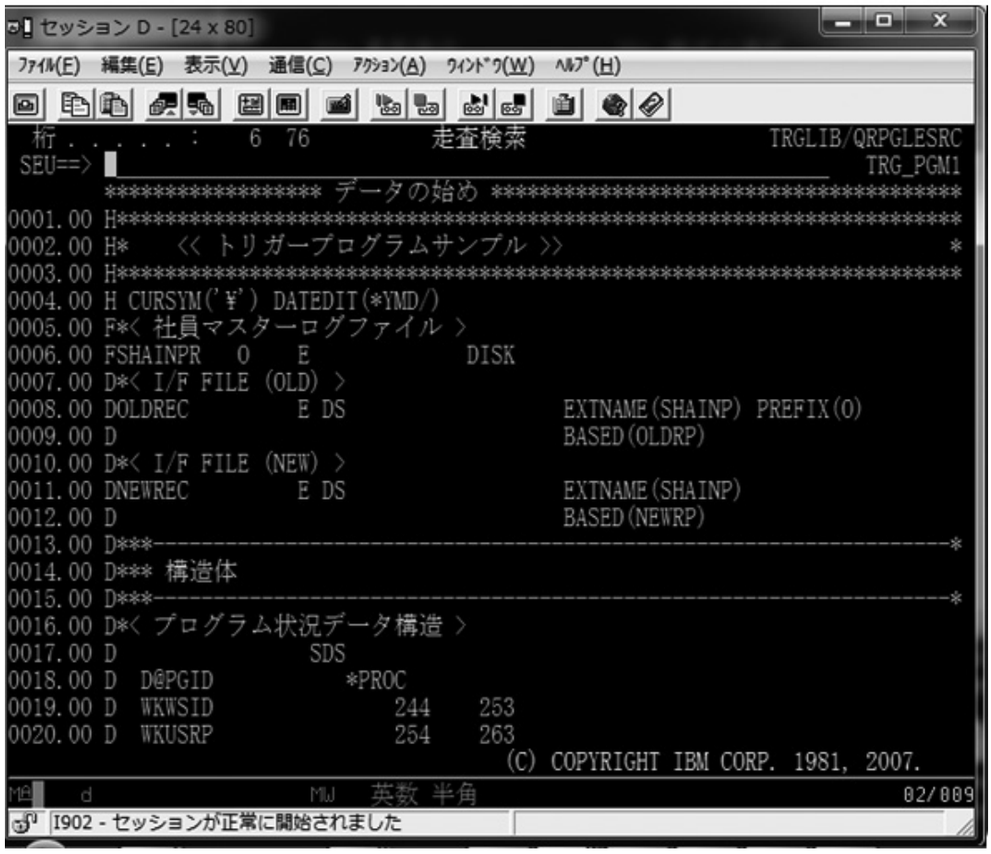


080_091_migaro_Report_八木沼_四.indd 88 17/10/25 14:18

図5



ソース1



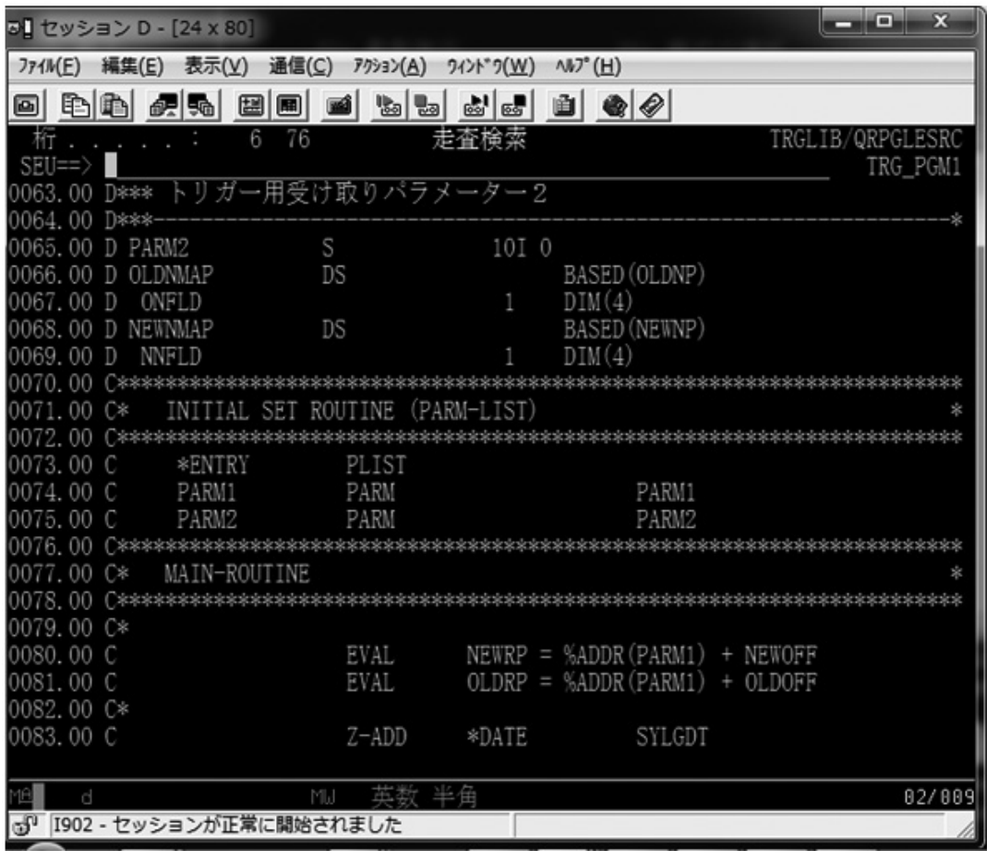
080_091_migaro_Report_八木沼_四.indd 89 17/10/25 14:18

ソース2

ソース3

080_091_migaro_Report_八木沼_四.indd 90 17/10/25 14:18

ソース4



ソース5

080_091_migaro_Report_八木沼_四.indd 91 17/10/25 14:18