General and Professional Education
2/2018 pp. 14-19
ISSN 2084-1469
DOI: 10.26325/genpr.2018.2.3

ZASTOSOWANIEFUNKCJI *PL/SQL* DO IMPLEMENTACJI POLITYKI HASEŁ APLIKACJI BAZODANOWEJ

APPLICATION OF PL / SQL FUNCTIONS FOR IMPLEMENTATION OF THE BATHODY APPROVAL POLICY IMPLEMENTATION

Marcin Kowalski
Wyższa Szkoła Ekonomii i Innowacji,
Wyższa Szkoła Ekonomii i Informatyki w Krakowie
ul. Św. Filipa 17
31-150 Kraków
e-mail: mkowalski@wsei.edu.pl

Abstract: The paper describes an implementation of a password policy in an Enterprise Resource Planning system going at *RDBMS Oracle*. The implementation uses a verification *PL/SQL* function and system of users' profiles. Database provides a sample password verification function in the *PL/SQL* script `UTLPWDMG.SQL` that, when enabled, checks whether users are correctly creating or modifying their passwords. The Verify Function is a quick and easy way to enforce quality of database passwords.

Keywords: personal data protection, ERP system, Oracle database, PL / SQL function, user profile, user password.

Wprowadzenie

Z uwagi na rosnące zagrożenie wykradzenia danych osobowych konieczne staje się wprowadzenie i przestrzeganie właściwej polityki haseł użytkowników przetwarzających takie dane. Przepisy wykonawcze do Ustawy o ochronie danych osobowych narzucają ścisłe wymogi jakości i częstotliwości zmiany hasła użytkownika przetwarzającego dane osobowe [1]. Wymagane jest, aby użytkownik przetwarzający dane osobowe zmie-niał hasło nie rzadziej niż raz na 30 dni. Hasło musi się składać przynajmniej z 8 znaków i musi zawierać przynajmniej jedną dużą literę, jedną małą literę oraz przynajmniej jedną cyfrę lub znak specjalny.

Opis rozwiązania

Przykładem przetwarzania danych osobowych jest eksploatacja systemu klasy *Enterprise Resource Planning* (*ERP*), w których gromadzone są i przetwarzane zarówno dane osobowe pracowników jak i kontrahentów firmy eksploatującej *ERP*. Narzędzia systemu ERP są obecnie szeroko wykorzystywane w przedsiębiorstwach, zwłaszcza przez operatorów logistycznych. Ochrona danych wrażliwych firm stanowi ważny element efektywnego funkcjonowania przedsiębiorstw. Studenci - przyszli użytkownicy systemów klasy ERP poznają w ten sposób niektóre z metod ochrony danych. Autor jest administratorem systemu *ERP* firmy komercyjnej. System pracuje na bazie danych *Oracle*. *Oracle* jest skrótem często używanym w żargonie informatyków, poprawna pełna nazwa tego systemu bazodanowego to *System Zarządzania Relacyjna Bazą Danych Oracle* (*Relational DataBase Management SystemOracle Oracle – RDBMS Oracle*).
Wymogi dotyczące hasła użytkownika *Oracle* zapisane są w profilu użytkownika. Profil użytkownika jest zbiorem ograniczeń nakładanych na użytkownika systemu bazodanowego dotyczących wykorzystania zasobów serwera i restrykcji związanych z hasłem użytkownika. Nowy użytkownik *Oracle* ma domyślnie przypisany profil o nazwie *DEFAULT*. Profil *DEFAULT* jest dostępny od razu po zainstalowaniu bazy danych [2]. Profil domyślny nie nakłada prawie żadnych ograniczeń na hasła użytkowników.
Za pomocą następującego zapytania *SQL* można uzyskać informacje o nastawach dotyczących hasła, zapisanych w profilu *DEFAULT*:

```
SELECT PROFILE, RESOURCE_NAME, LIMIT FROM DBA_PROFILES
WHERE PROFILE ='DEFAULT'AND RESOURCE_TYPE ='PASSWORD';
```

Wynik zapytania:

PROFILE	RESOURCE_NAME	LIMIT
DEFAULT	FAILED_LOGIN_ATTEMPTS	10
DEFAULT	PASSWORD_LIFE_TIME	180
DEFAULT	PASSWORD_REUSE_TIME	UNLIMITED
DEFAULT	PASSWORD_REUSE_MAX	UNLIMITED
DEFAULT	PASSWORD VERIFY FUNCTION	NULL
DEFAULT	PASSWORD_LOCK_TIME	1
DEFAULT	PASSWORD_GRACE_TIME	7

Ograniczenia dotyczące hasła zawarte w profilu domyślnym dotyczą jedynie:
- liczby prób logowania zakończonych niepowodzeniem, po których konto użytkownika zostaje blokowane (FAILED_LOGIN_ATTEMPTS);
- liczby dni, przez które konto jest blokowane po przekroczeniu liczby prób logowania zakończonych niepowodzeniem (PASSWORD_LOCK_TIME), po upływie tego czasu konto zostaje automatycznie odblokowane;
- liczby dni, przez które można używać hasła, jeśli hasło nie zostanie zmienione. Po upływie liczby dni określonej w PASSWORD_LIFE_TIME plus PASSWORD_GRACE_TIME konto użytkownika zostanie zablokowane;

Na pozostałe parametry nie są nałożone żadne ograniczenia (UNLIMITED). Nie jest też włączone sprawdzanie jakości hasła za pomocą funkcji *PL/SQL*. Decyduje o tym parametr PASSWORD_VERIFY_FUNCTION ustawiony na NULL.

Aby wprowadzić ograniczenia dotyczące hasła najwygodniej jest nie modyfikować profilu domyślnego, tylko zdefiniować nowy profil, dostosowany do wymogów polityki haseł obowiązującej w organizacji. Nie należy modyfikować profilu domyślnego, ponieważ korzystają z niego specjalne konta, używane na przykład do wykonywania cyklicznych czynności w systemie *ERP*. Czynności takich jak raportowanie działania aplikacji, wykonywane w celu jej optymalizacji, czyszczenie tabel tymczasowych lub import danych z systemów współpracujących z główną aplikacją systemu *ERP*. Zmieniając parametry profilu domyślnego (na przykład wprowadzając okres ważności hasła) administrator bazy danych musiałby pamiętać o okresowym odświeżaniu haseł tych kont specjalnych.

Nowy profil (o przykładowej nazwie *POLICY_PASSWD*) tworzy się za pomocą następującego polecenia *SQL*:

```
CREATEPROFILE POLICY_PASSWD LIMIT
SESSIONS_PER_USERDEFAULT
CPU_PER_SESSIONDEFAULT
CPU_PER_CALLDEFAULT
CONNECT_TIMEDEFAULT
IDLE_TIMEDEFAULT
LOGICAL_READS_PER_SESSIONDEFAULT
LOGICAL_READS_PER_CALLDEFAULT
COMPOSITE_LIMITDEFAULT
PRIVATE_SGADEFAULT
FAILED_LOGIN_ATTEMPTS10
PASSWORD_LIFE_TIME30
PASSWORD_REUSE_TIMEDEFAULT
PASSWORD_REUSE_MAXDEFAULT
PASSWORD_LOCK_TIMEDEFAULT
PASSWORD_GRACE_TIMEDEFAULT
PASSWORD_VERIFY_FUNCTIONVERIFY_F;
```

Te parametry nowego profilu, przy których w powyższym poleceniu wybrano opcję DEFAULT dziedziczą wartości z profilu o nazwie *DEFAULT*. Nowy profil różni się od domyślnego skróceniem czasu ważności hasła do 30 dni oraz włączeniem weryfikacji jakości hasła przez funkcję *PL/SQL* onazwie VERIFY_F.

Funkcja sprawdzająca jakość hasła użytkownika w *RDBMS Oracle*

Aby sprawdzać jakość hasła użytkownika, należy oprócz zmiany profilu, zapisać w bazie danych iskompilować odpowiednią funkcję *PL/SQL*. Funkcja ta musi mieć taką samą nazwę, jaką ma parametr PASSWORD_VERIFY_FUNCTION w profilu użytkownika. W rozpatrywanym przypadku funkcja ta powinna nazywać się VERIFY_F. *Oracle* udostępnia szablon takiej funkcji. W wersji 11g jest on zapisany w skrypcie *$ORACLE_HOME/rdbms/utlpwdmg.sql* [1]. Oprócz funkcji o nazwie *verify_function_11G* dla wersji *Oracle 11g*, szablon zawiera też wersję funkcji weryfikującej o nazwie *verify_function* pochodzącą ze starszej wersji Oracle.

Funkcje w formie dostarczanej przez szablon sprawdzają jakość hasła w minimalnym stopniu. Administrator może w prosty sposób zmodyfikować funkcję weryfikującą, tak aby spełniała ona wymogi obowiązującej w organizacji polityki haseł.

Poniżej podany jest kod funkcji VERIFY_F, używanej do sprawdzania jakości hasła użytkowników systemu *ERP* opartego na bazie danych *Oracle 11g*. Systemem tym jest *IMPULS EVO* produkcji firmy BPSC S.A. Zamieszczony kod jest kodem funkcji języka *PL/SQL* (*Procedural Language/Structured Query Language*), napisanym na bazie szablonu dostarczonego z instalacją bazy danych. Język *PL/SQL* jest rozwinięciem standardu *SQL* stosowanym w *RDBMS Oracle* i umożliwia dodawanie do kodu *SQL* konstrukcji programistycznych. Można w nim przechwytywać i obsługiwać wyjątki, stosować pętle i zaawansowane instrukcje warunkowe oraz tworzyć wyzwalacze, kursory, funkcje, procedury i pakiety [2].

Funkcja *PL/SQL* jest przechowywana w bazie danych w postaci kodu źródłowego (podanego poniżej) oraz w postaci skompilowanej. Kod źródłowy funkcji został sformatowany przez edytor wbudowany programu *Toad for Oracle*. Edytor ten formatuje tekst w sposób przyjazny dla użytkownika, tworząc wcięcia i kolorując czcionki. Dużymi literami napisane są słowa kluczowe *PL/SQL*.

```
1   CREATEORREPLACEFUNCTIONsys.verify_f
2   (username varchar2, passwordvarchar2, old_password varchar2)
3   RETURNbooleanIS
4     n boolean; m integer; differ integer; isdigit boolean;
5     islowchar boolean; iscapchar  boolean; ispunct boolean;
6     digitarray varchar2(20);punctarray varchar2(60);
7   cap_chararray varchar2(52); low_chararray varchar2(52);
```

```
8   BEGIN
9     digitarray:='0123456789';
10    cap_chararray:='ABCDEFGHIJKLMNOPQRSTUVWXYZ';
11    low_chararray:='abcdefghijklmnopqrstuvwxyz';
12    punctarray:='!"#$%&()``*+,-/:;<=>?_';
13  IFNLS_LOWER(password)=NLS_LOWER(username)THEN
14      raise_application_error(-20001,'Hasło takie same lub podobne do nazwy użytkownika');
15  ENDIF;
16  IFLENGTH(password)< 8 THEN
17       raise_application_error(-20002,'Hasło ma długość krótszą niż 8 znaków');
18  ENDIF;
19    isdigit:=FALSE;
20  m:=LENGTH(password);
21  FOR i IN 1..10 LOOP
22  FOR j IN 1..m LOOP
23  IFSUBSTR(password,j,1)=SUBSTR(digitarray,i,1)THEN
24           isdigit:=TRUE;
25  GOTO findlowchar;
26  ENDIF;
27  ENDLOOP;
28  ENDLOOP;
29  GOTO findpunct;
30  <<findlowchar>>
31    islowchar:=FALSE;
32  FOR i IN 1..LENGTH(low_chararray)LOOP
33  FOR j IN 1..mLOOP
34  IFSUBSTR(password,j,1)=SUBSTR(low_chararray,i,1)THEN
35           islowchar:=TRUE;
36  GOTO findcapchar;
37  ENDIF;
38  ENDLOOP;
39  ENDLOOP;
40  IF islowchar =FALSETHEN
41       raise_application_error(-20003,'Brak małej litery w haśle');
42  ENDIF;
43  <<findcapchar>>
44    iscapchar:=FALSE;
45  FOR i IN 1.. LENGTH(cap_chararray)LOOP
46  FOR j IN 1..mLOOP
47  IFSUBSTR(password,j,1)=SUBSTR(cap_chararray,i,1)THEN
48  iscapchar:=TRUE;
49  GOTO endsearch;
50  ENDIF;
51  ENDLOOP;
52  ENDLOOP;
53  IF iscapchar =FALSETHEN
54       raise_application_error(-20003,'Brak dużej litery w haśle');
55  ENDIF;
56  <<findpunct>>
57  ispunct:=FALSE;
58  FOR i IN 1..LENGTH(punctarray)LOOP
59  FOR j IN 1..mLOOP
60  IFSUBSTR(password,j,1)=SUBSTR(punctarray,i,1)THEN
61           ispunct:=TRUE;
62  GOTO findlowchar;
63  ENDIF;
64  ENDLOOP;
65  ENDLOOP;
```

```
66      IF ispunct =FALSETHEN
67          raise_application_error(-20003,'Brak cyfry lub znaku specjalnego w haśle');
68      ENDIF;
69      <<endsearch>>
70      IF old_password ISNOTNULLTHEN
71         differ :=LENGTH(old_password)-LENGTH(password);
72      IFABS(differ)< 3 THEN
73      IFLENGTH(password)<LENGTH(old_password)THEN
74      m:=LENGTH(password);
75      ELSE
76      m:=LENGTH(old_password);
77      ENDIF;
78          differ :=ABS(differ);
79      FOR i IN1..mLOOP
80      IFSUBSTR(password,i,1)!=SUBSTR(old_password,i,1)THEN
81              differ := differ + 1;
82      ENDIF;
83      ENDLOOP;
84      IF differ < 3 THEN
85      raise_application_error(-20004,'Hasła nie różnia się przynajmniej 3 znakami');
86      ENDIF;
87      ENDIF;
88      ENDIF;
89      RETURN(TRUE);
90      END;
91      /
```

Poniżej omówione będą punkty charakterystyczne dla funkcji *PL/SQL*. W nawiasach podane są numery wierszy kodu źródłowego.

(2) W nawiasach podana jest lista parametrów przesyłanych do funkcji: *(username varchar2, passwordvarchar2, old_password varchar2)*

Charakterystyczne dla funkcji *PL/SQL* jest to, że przy parametrach podaje się typ zmiennej a nie podaje wielkości zmiennej [3]. Przy deklaracji parametrów funkcji można podać typ parametru. Do wyboru są trzy tryby przekazywania parametrów:

IN – Oznacza parametr *przekazywany przez wartość*. Parametr taki przekazuje wartość ze środowiska wywołującego funkcję do funkcji. Wewnątrz funkcji zachowuje się jak stała, jego wartość nie może być zmieniona. Tryb *IN* jest domyślny;

OUT – Oznacza parametr *przekazywany przez referencję*, parametr taki przekazuje wartość z wnętrza funkcji do środowiska wywołującego funkcję. Wewnątrz funkcji parametr taki zachowuje się jak niezainicjalizowana zmienna, której wewnątrz funkcji należy nadać wartość;

IN OUT - Oznacza parametr *przekazywany przez referencję*, parametr tego typu jest połączeniem poprzednich typów. Za jego pomocą funkcja pobiera wartość parametru z zewnątrz i po wykonaniu obliczeń zwraca go do środowiska wywołującego.

Domyślną typem jest *IN*.

W wierszu (2) nie podano trybu przekazywania parametrów, dlatego przyjmują one domyślny tryb *IN*.

(3) Po obowiązkowej klauzuli *RETURN* podany jest typ wartości, jaką zwróci funkcja:

RETURNbooleanIS

Jeśli wszystkie wymogi dotyczące hasła zostaną spełnione (nie zostanie zgłoszony żaden wyjątek oznaczający, że hasło nie spełnia zakładanych wymogów) to w wierszu (89) zostanie wykonana obowiązkowa dla każdej funkcji *PL/SQL* instrukcja *RETURN* i zwrócony do środowiska wywołującego wynik działania tej funkcji. W rozpatrywanej funkcji będzie to wartość *TRUE*. Brak w ciele funkcji instrukcji *RETURN* skutkowałby błędem kompilacji.

(4-7) Pomiędzy obowiązkowymi klauzulami *IS* oraz *BEGIN* podane są parametry występujące wewnątrz funkcji oraz ich typ. W tym miejscu już muszą być podane wielkości zmiennych. Sekcja ta jest opcjonalna, mogą być w niej zadeklarowane oprócz zmiennych również stałe, wyjątki i kursory.

Pomiędzy obowiązkowymi klauzulami *BEGIN* (8) i *END* (90) jest podane *ciało funkcji*.

(9-12) W tym miejscu zmiennym zadeklarowanym wewnątrz funkcji (pomiędzy *IS* a *BEGIN*) przypisuje się wartości:

Digitarray:='0123456789';

cap_chararray:='ABCDEFGHIJKLMNOPQRSTUVWXYZ';

low_chararray:='abcdefghijklmnopqrstuvwxyz';
punctarray:='!"#$%&()``+,-/:;<=>?_';*
(13-15) W instrukcji warunkowej *IF WARUNEK THEN POLECENIE; END IF;* sprawdzane jest czy hasło nie jest takie same jak nazwa użytkownika.

```
IFNLS_LOWER(password)=NLS_LOWER(username)THEN
raise_application_error(-20001,'Hasło takie same lub podobne do nazwy użytkownika');
ENDIF;
```

Przy czym litera pisana małą i dużą literą traktowana jest jako taka sama. Do zamiany wszystkich znaków na małe litery służy funkcja *PL/SQL NLS_LOWER* (nowsza wersja funkcji *LOWER* pozwalająca na obsługę *NLS – National Languages Support*). Jeśli hasło jest takie same jak nazwa użytkownika, to uruchamiana jest funkcja *raise_application_error*. Funkcja ta służy do obsługi dynamicznych wyjątków zdefiniowanych przez użytkownika. Jest prosta w użyciu, nie wymaga deklarowania wyjątków, czy używania dyrektywy kompilatora *PRAGMA EXCEPTION_INIT*. Po uruchomieniu funkcji *raise_application_error* wyświetlany jest kod błędu (programista ma do wyboru kody błędów w zakresie od -20999 do -20000) oraz komunikat błędu, również zdefiniowany przez programistę. Po czym działania funkcji weryfikującej *VERIFY_F* kończy się i następuje powrót do środowiska wywołującego funkcję.
(16-18) Za pomocą wbudowanej funkcji *LENGTH* sprawdza się, czy hasło nie składa się z mniej niż z 8 znaków. Jeśli tak jest, to również zgłaszany jest dynamiczny wyjątek za pomocą funkcji *raise_application_error*. i powrót do środowiska wywołującego.

```
IFLENGTH(password)<8THEN
    raise_application_error(-20002,'Hasło ma długość krótszą niż 8 znaków');
ENDIF;
```

(19) Inicjalizuje się zmienną typu *Boolean*, przechowującą wynik sprawdzenia, czy hasło zawiera cyfrę.

```
  isdigit:=FALSE;
```

(20-28) W dwu pętlach zewnętrznej i wewnętrznej następuje sprawdzenie, czy którykolwiek znak hasła jest równy jakiejkolwiek cyfrze (wszystkie 10 cyfr zapisane są w zmiennej *punctarray*).

```
m:=LENGTH(password);
FOR i IN1..LENGTH(punctarray)LOOP
FOR j IN1..mLOOP
IFsubstr(password,j,1)=substr(digitarray,i,1)THEN
        isdigit:=TRUE;
GOTO findlowchar;
ENDIF;
ENDLOOP;
ENDLOOP;
```

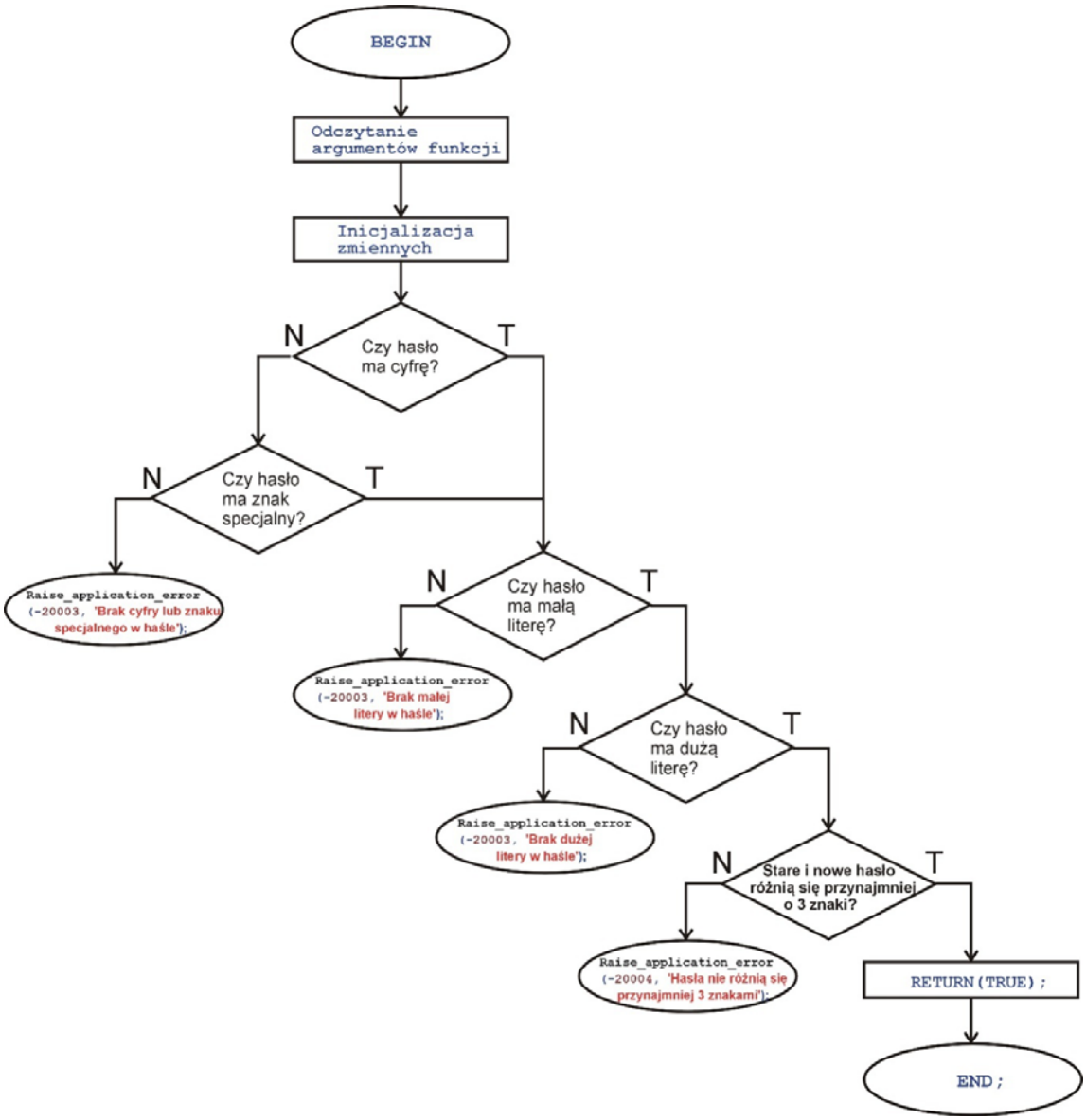
Po napotkaniu zgodności (hasło zawiera cyfrę), zmiennej *isdigit* przypisywana jest wartość *TRUE*, następnie występuje przejście do miejsca kodu funkcji oznaczonego etykietą *<<findlowchar>>*(30). Podręczniki *PL/SQL* nie zalecają co prawda stosowania instrukcji *GOTO* [2], jednak zastosowany w tym miejscu (25) skok do miejsca oznaczonego etykietą *<<findlowchar>>* znacznie upraszcza kod funkcji. Jeśli natomiast nie zostanie znaleziona cyfra w haśle, to wykonany zostaje skok (za pomocą instrukcji *GOTO*) do kolejnej instrukcji warunkowej *IF ... END IF* sprawdzającej, czy w haśle zawarty jest znak specjalny (zawarty w zmiennej *punctarray*).
W opisie działania funkcji pominięto kolejne fragmenty typowych instrukcji warunkowych, podobnych do omówionego powyżej fragmentu sprawdzającego istnienie cyfry w haśle. Algorytm działania funkcji *VERIFY_F* został niżej zaprezentowany w formie graficznej.
(89) Jeśli spełnione są wymogi dotyczące hasła zawarte w kodzie funkcji, to funkcja zwraca do środowiska wywołującego wartość *TRUE*.

```
RETURN(TRUE);
```

(90) działanie funkcji kończy słowo kluczowe *END*.
(91) Kod funkcji zakończony jest ukośnikiem. W języku *PL/SQL* ukośnik oznacza koniec kodu funkcji.
Na rys. 1 przedstawino algorytm działanie instrukcji warunkowych i skoków w formie graficznej.

Funkcja sprawdzająca jakość hasła w *Oracle 12c*

Najnowsza wersja *Oracle 12c* również dostarcza szablon funkcji *PL/SQL* sprawdzającej, czy hasło użytkownika spełnia założone wymagania. Szablon ten również tak jak w *11g* jest zapisany w pliku *utlpwdmg.sql*. Domyślny profil użytkownika w *12c* również nie włącza funkcji sprawdzającej złożoności hasła. Aby hasła użytkowników były sprawdzane, należy tak jak w 11g, utworzyć nowy profil użytkownika i skompilować funkcję o nazwie użytej w profilu w miejscu *PASSWORD_VERIFY_FUNCTION*. W stosunku do wcześniejszej wersji w *12c*wprowadzono zmiany. W szablonie, oprócz kodu funkcji sprawdzającej, zawarty jest kod dwu nowych funkcji. Są to funkcje *complexity_check*, sprawdzająca złożoność hasła oraz funkcja *string_distance*, obliczająca *odległość Levenshtein'a* pomiędzy starym a nowym hasłem. Właściwa funkcja sprawdzająca o nazwie *ora12c_verify_function* wykorzystuje obydwie wyżej wymienione funkcje. Administrator ma do dyspozycji w szablonie jeszcze jedną funkcje sprawdzającą o nazwie *ora12c_strong_verify_function*. Różni się ona od poprzedniej funkcji tym, że wymaga, aby hasła miało przynajmniej 9 znaków (w tym co najmniej 2 duże litery, 2 małe litery, 2 cyfry i znaki specjalne) oraz aby nowe hasło różniło się od starego przynajmniej o 4 znaki. Nie sprawdza natomiast (tak jak *ora12c_verify_function*), czy hasło nie zawiera nazwy użytkownika, odwrotnej nazwy użytkownika, nazwy serwera bazy danych oraz słowa „oracle".



Rys. 1 Algorytm działania funkcji PL/SQL VERIFY_F sprawdzającej wymaganą złożoność hasła użytkownika.

Podsumowanie

Omówiona w pracy funkcja weryfikująca złożoność hasła w połączeniu z systemem profili użytkowników jest bardzo dobrym udogodnieniem dla administratora systemu ERP opartego na bazie danych Oracle. Umożliwia ona dostosowanie wymogów haseł użytkowników do aktualnych przepisów w bardzo prosty sposób. Jej kod jest przejrzysty, co ułatwi ewentualne późniejsze dostosowanie haseł użytkowników, w przypadku ewentualnej zmiany przepisów prawa, bądź wewnętrznej polityki bezpieczeństwa organizacji.

Bibliografia

1. Bryla, B., Loney, K., Oracle Database 11g. Podręcznik administratora baz danych, Wydawnictwo HELION, 2010.
2. McLaughlin, M., Oracle Database 12c Programowanie w języku PL/SQL, Wydawnictwo HELION, 2015.
3. Price, J., Oracle Database 12c i SQL programowanie, Wydawnictwo HELION, 2015.
4. Załącznik do rozporządzenia Ministra Spraw Wewnętrznych i Administracji z dnia 29 kwietnia 2004 r., B.VIII. – Środki bezpieczeństwa na poziomie podwyższonym.