Cofinanziato dall'Unione europea

REGIONE PIEMONTE

# FinTech Software Developer

# Basi di dati SQL

Docente: Roi Davide Simone

Titolo argomento: Sviluppo su Postgresql – Scrittura di Stored Procedures e Trigger (2° parte)

Roi Davide
Dispense

Intervento realizzato da ITS Tecnologie dell'Informazione e della Comunicazione

# Tipologie di lavori su DATABASE

Roi Davide Dispense

Le principali tipologie di lavori per i quali è richiesta la progettazione e l'implementazione di PROCEDURE/FUNZIONI sui database si può riassumere in queste categorie:

- TRATTAMENTO DATI
  - CONTROLLI CONSISTENZA DATI
  - AGGREGAZIONE DATI
  - CORREZIONE DATI
- ANALISI DEI DATI
  - VERIFICHE VOLUMI DATI
  - CALCOLO DI KPI (Indicatori) da' info in valori
- TRASPORTO DATI
  - GESTIONE E REGISTRAZIONE INVII
  - GESTIONE E REGISTRAZIONE RESTITUZIONI

# Monitoraggio delle Procedure



## Informatici

La Mail è il mio avvocato !


## Sviluppatori Database

Il Log è il mio avvocato !

# Monitoraggio delle Procedure

(Tabella di LOG)

Roi Davide Dispense

Per una corretta gestione delle procedure su database è buona norma avere sempre una propria gestione dei LOG che tracci **tutte le attività svolte dalle proprie procedure**, in modo da avere il pieno controllo di quello che è accaduto, soprattutto quando i processi vengono automatizzati. Il log può essere scritto su file o su tabella, quando possibile consiglio sempre di creare una **tabella di LOG** in quanto più versatile da interrogare puntualmente con I filtri, che contenga almeno le seguenti informazioni di base:

**identificativo_sequenziale**

Un sequenziale numerico autoincrementale che permette sempre di ordinare temporalmente le scritture dei log .

**tracciato**

Campo che contenga il nome del tracciato o della procedura che si vuole monitorare, in modo da poter leggere il log filtrando solo il tracciato di interesse e ordinandolo per identificativo.

**livello**

Campo che descriva il livello di gravità del LOG, in modo da poter discriminare tra log di ERRORI, SEGNALAZIONI, WORNING, ecc…

**data_update**

Campi che registra il timestamp di quando viene scritto il log (informazione importante per capire quando è avvenuto un certo evento)

**testo**

Campo libero per scrivere il testo del log

# Monitoraggio delle Procedure

(Tabella di LOG)



**Esempio di come scrivere i LOG in una procedura/funzione:**

```
DECLARE
    ris1 integer;
BEGIN
    insert into ft_log_monitoraggio (livello, tracciato,testo) values ('','tracciato1','inizio procedura: XXXX');

    select count(*) into ris1 from nome_tabella;
    if ris1 > 1 then
        ....
        ....
    end if;

    insert into ft_log_monitoraggio (livello, tracciato,testo) values ('','tracciato1','fine procedura: XXXX');
END;
```

# Gestione delle ECCEZIONI

Roi Davide Dispense

Ogni qual volta una procedura/funzione tenti di fare una operazione non consentita (inserimento/selezione/update/operazione non consentita) il DBMS forzerà l'uscita dalla procedura tramite un comando di **EXCEPTION che viene passato alla procedura chiamante.**
Se la procedura chiamante dovesse non avere una gestione delle eccezioni, anch'essa forzerà l'uscita tramite EXCEPTION, e così via in cascata fino a terminate tutte le procedure contenenti altre procedure.

Questa uscita forzata può essere gestita nelle vostre procedure implementando esplicitamente il commando di EXCEPTION che permete di verificare il tipo di eccezione ed eventualmente decidere di eseguire le operazioni precise da voi scritte.

Di buona norma ricordatevi sempre di implementare la "Gestione delle eccezioni" in quanto dobbiamo garantire che qualsiasi tipo di ECCEZIONE venga da noi intercettata e gestita correttamente.

Il commando EXCEPTION gestisce 2 variabili di Sistema (di tipo TESTO) che possono essere intercettate:

**Sqlerrm** = contiene la DESCRIZIONE DELLA ECCEZIONE

**Sqlstate** = contiene il CODICE DELLA ECCEZIONE

# Gestione delle ECCEZIONI



**Esempio di come scrivere un record di LOG a fronte di una eventuale eccezione riscontrata:**

```
DECLARE
    var1 integer;
BEGIN
    insert into ft_log_monitoraggio (livello, tracciato,testo) values ('tracciato1','inizio procedura: XXXX', '');
    var1 := 2/0;
    insert into ft_log_monitoraggio (livello, tracciato,testo) values ('tracciato1','fine procedura: XXXX', '');
EXCEPTION
    WHEN OTHERS THEN
        BEGIN
            insert into ft_log_monitoraggio (livello, tracciato, testo)
            values ('tracciato1', 'procedura __aaa_test1 - Descrizione:' || sqlerrm || ' - Codice errore:' || sqlstate, 'ERRORE');
            -- la seguente riga è commentata ma se la scommento propago l'errore alla funzione chiamante
            --RAISE EXCEPTION '% %',sqlstate,sqlerrm;
        END;
END;
```

termina e riconsegna....?

# Gestione delle ECCEZIONI



**Esempio di procedura dedicata che scriva il log:**

```
CREATE OR REPLACE PROCEDURE scrivi_log(p_tracciato varchar(1000), p_testo varchar(4000), p_livello varchar(1000) DEFAULT '')
 LANGUAGE plpgsql
AS $procedure$
begin
    insert into ft_log_monitoraggio (tracciato, testo, livello) values (p_tracciato, p_testo, p_livello);
 end; $procedure$
;
```

**Esempio di utilizzo della procedura scrivi_log() all'interno di una pricedura chiamata procedura_test1:**

```
CREATE OR REPLACE PROCEDURE procedura_test1()
LANGUAGE plpgsql
AS $procedure$
DECLARE
   var1 integer;
BEGIN
    call scrivi_log('tracciato1','inizio procedura_test1');
   var1 := 2/0;
    call scrivi_log('tracciato1','fine procedura_test1');
EXCEPTION
   WHEN OTHERS THEN
      BEGIN
         call scrivi_log( 'tracciato1','procedura_test1: ' || sqlerrm, 'ERRORE');
      END;
END;  $procedure$
;
```

# Gestione delle ECCEZIONI

(tipi di condizioni principalmente verificate)

Roi Davide Dispense

```
exception
    when no_data_found then /*quando una select non trova nessun record*/
        code_exception;
    when too_many_rows then  /*quando una select restituisce più di una riga*/
        code_exception;
    when division_by_zero then  /*quando si tenta di fare una divisione del numero 0*/
        code_exception;
    when others then /*in tutti gli altri casi*/
        code_exception;
end;
```

P.S. Per verificare tutte le possibili condizioni fare riferimento alla documentazione ufficiale su: https://www.postgresql.org/docs/current/errcodes-appendix.html

# FLOW CHART

(documentare il nostro lavoro in modo professionale)

Roi Davide Dispense

La rappresentazione grafica FLOW CHART (diagramma di flusso) è lo strumento principale e per eccellenza che gli sviluppatori di «codice sequenziale» utilizzano per descrivere e documentare un processo sequenziale (una operazione dopo l'altra).
Questo strumento si adatta perfettamente per descrivere le procedure e le funzioni che operano all'interno del DATABASE in quanto il codice PL/PGSQL è un codice sequenziale.

**Le forme più comuni utilizzate in un flowchart includono:**

- Ovali: rappresentano l'inizio o la fine del processo.
- Rettangoli: rappresentano le azioni o le operazioni da svolgere.
- Rombi: rappresentano le decisioni o le scelte che devono essere prese.
- Frecce: indicano il flusso sequenziale delle operazioni da una forma all'altra.

# TRASPORTO DATI

(Gestione e registrazione invii)

Roi Davide Dispense

Procedure di
**CARICAMENTO**

Tutto il processo che legge i dati dalle «tabelle di origine» e li scrive su tabelle codificate come richiesto dal destinatario.

Queste tabelle possono essere STORICHE oppure di STACK

Procedure di
**ESTRAZIONE**

Tutto il processo che legge i dati «caricati» e li estrae su file/mqtt/ecc… richiamando il servizio di spedizione (manuale, mqtt, web-services, ecc..)

Procedure di
**ESITO**

Tutto il processo che legge (laddove presenti) le restituzioni con gli esiti puntuali da caricare nel sistema.

Intervento realizzato da ITS Tecnologie dell'Informazione e della Comunicazione

# Fonti:



- Documentazione ufficiale di Postgresql
  https://www.postgresql.org/docs/

Fine della Presentazione