Universidade do Minho

Ano Letivo: 2023/24
Turnos: PL3/PL7

Bases de Dados

PL11 – SQL Avançada

Docente: Diana Ferreira
Email: diana.ferreira@algoritmi.uminho.pt

Horário de Atendimento:
5ª feira 16h-17h

Sumário

Bibliografia:

- Connolly, T., Begg, C., Database Systems, A Practical Approach to Design, Implementation, and Management , Addison-Wesley, 4a Edição, 2004. **(Chapter 6 e 7)**
- Belo, O., "Bases de Dados Relacionais: Implementação com MySQL", FCA – Editora de Informática, 376p, Set 2021. ISBN: 978-972-722-921-5. **(Capítulo 4 e 5)**

SQL AVANÇADA

TRANSACTIONS

A transação permite executar um conjunto de operações para garantir que a BD nunca contém o resultado de operações parciais.

- Para iniciar uma transação, usa-se a instrução **START TRANSACTION**.
- Para confirmar a transação atual e tornar as suas alterações permanentes, usa-se a instrução **COMMIT**.
- Para reverter a transação atual e cancelar as suas alterações, usa-se a instrução **ROLLBACK**.

Síntaxe:

```
START TRANSACTION;
<body>
ROLLBACK | COMMIT;
```

Exemplo:

```
START TRANSACTION;
DELETE FROM medicos WHERE estado='S';
ROLLBACK;
```

SQL AVANÇADA

HANDLERS

Quando ocorre um erro é importante tratá-lo adequadamente, como continuar (CONTINUE) ou sair (EXIT) da execução do bloco de código atual e emitir uma mensagem de erro significativa. Para isso usa-se um handler, através da instrução DECLARE HANDLER.

Síntaxe:

A **instrução a executar** pode ser uma instrução simples ou uma instrução composta delimitada pelas palavras-chave BEGIN e END.

Condição que ativa o handler

```
DECLARE action HANDLER FOR condition_value statement;
```

Tipo de ação

A **acção** pode ser:

- **CONTINUE:** a execução do bloco de código envolvente continua.
- **EXIT:** a execução do bloco de código envolvente termina.

Instrução a executar

A **condição de ativação** pode ser:

- Um valor SQLSTATE padrão: SQLWARNING , NOTFOUND ou SQLEXCEPTION.

SQL AVANÇADA

HANDLERS

Quando ocorre um erro é importante tratá-lo adequadamente, como continuar (CONTINUE) ou sair (EXIT) da execução do bloco de código atual e emitir uma mensagem de erro significativa. Para isso usa-se um handler, através da instrução DECLARE HANDLER.

Exemplos:

```
DECLARE EXIT HANDLER FOR SQLEXCEPTION SELECT 'SQLException encountered' Message;
```

```
DECLARE Erro INT DEFAULT 0;
DECLARE CONITNUE HANDLER FOR NOT FOUND SET Erro = 1;
```

SQL AVANÇADA

➡ PROCEDURE WITH TRANSACTIONS AND HANDLERS

```
DELIMITER $$
CREATE PROCEDURE ProcTransacExemplo ( IN codigo INT, IN farmaco VARCHAR(45), IN desc VARCHAR(150), OUT res VARCHAR(100)
BEGIN
DECLARE erro INT DEFAULT 0;
DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro=1;
START TRANSACTION;
INSERT INTO Farmacos(id_farmaco,nome,descricao) VALUES(codigo,farmaco,desc);
SELECT COUNT(*) FROM Farmacos WHERE id_farmaco = codigo;
IF erro = 1 THEN
    ROLLBACK;
    SET res = 'Transação abortada.';
    LEAVE InserirFarmaco;
END IF;
(...)
END $$
DELIMITER ;
```

SQL AVANÇADA

EVENTOS

Um evento é uma tarefa que é executada num horário específico. Os eventos podem ser criados para execução única ou para determinados intervalos. Para executar o evento de forma repetitiva, a cláusula EVERY pode ser usada.

Síntaxe:

```
CREATE EVENT <event_name>
  ON SCHEDULE <time_stamp> | EVERY <quantitiy> <unit>
  DO <event_body>;
```

NOTA pode ser necessário ativar a variável:

```
SET GLOBAL event_scheduler = ON;
```

SQL AVANÇADA

EVENTOS

Exemplos:

```
CREATE EVENT one_time_log
ON SCHEDULE AT CURRENT_TIMESTAMP
DO
INSERT INTO messages(message) VALUES('One-time event');
```

```
CREATE EVENT recurrent_time_log
ON SCHEDULE EVERY 1 HOUR
STARTS CURRENT_TIMESTAMP
ENDS CURRENT_TIMESTAMP + INTERVAL 1 MONTH
DO
INSERT INTO messages(message) VALUES(CONCAT('Event at'), NOW());
```

FASE 6: Exploração

Resolução de Exercícios

Ficha de Excercícios PL11