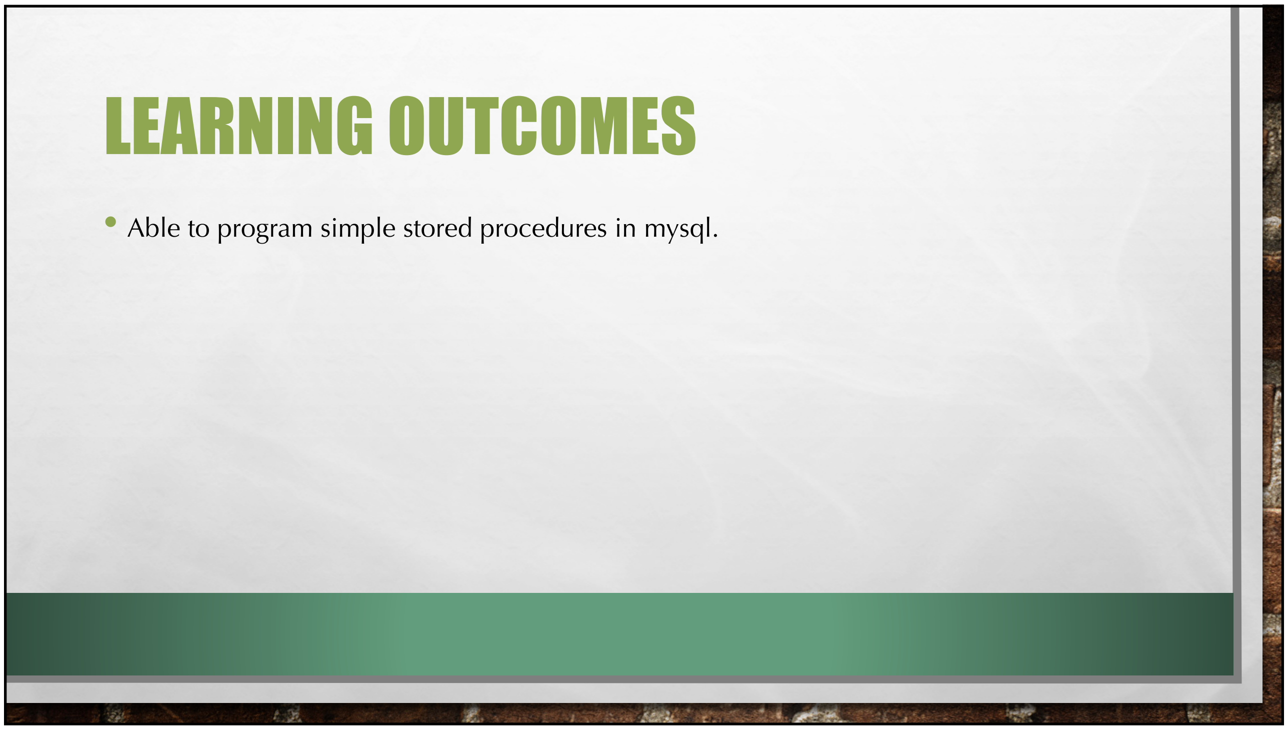
STORED PROCEDURES IN MYSQL

RICK WIGHTMAN, UNB FACULTY OF COMPUTER SCIENCE

1



LEARNING OUTCOMES

- Able to program simple stored procedures in mysql.

2

STORED PROCEDURE LANGUAGE

- MySQL has a procedural language extension
- Provides for declaring variables, assignment statements, decision and loop control structures, functions and procedures, and error handling
- Data types are compatible with MySQL data types

3

STORED PROCEDURES

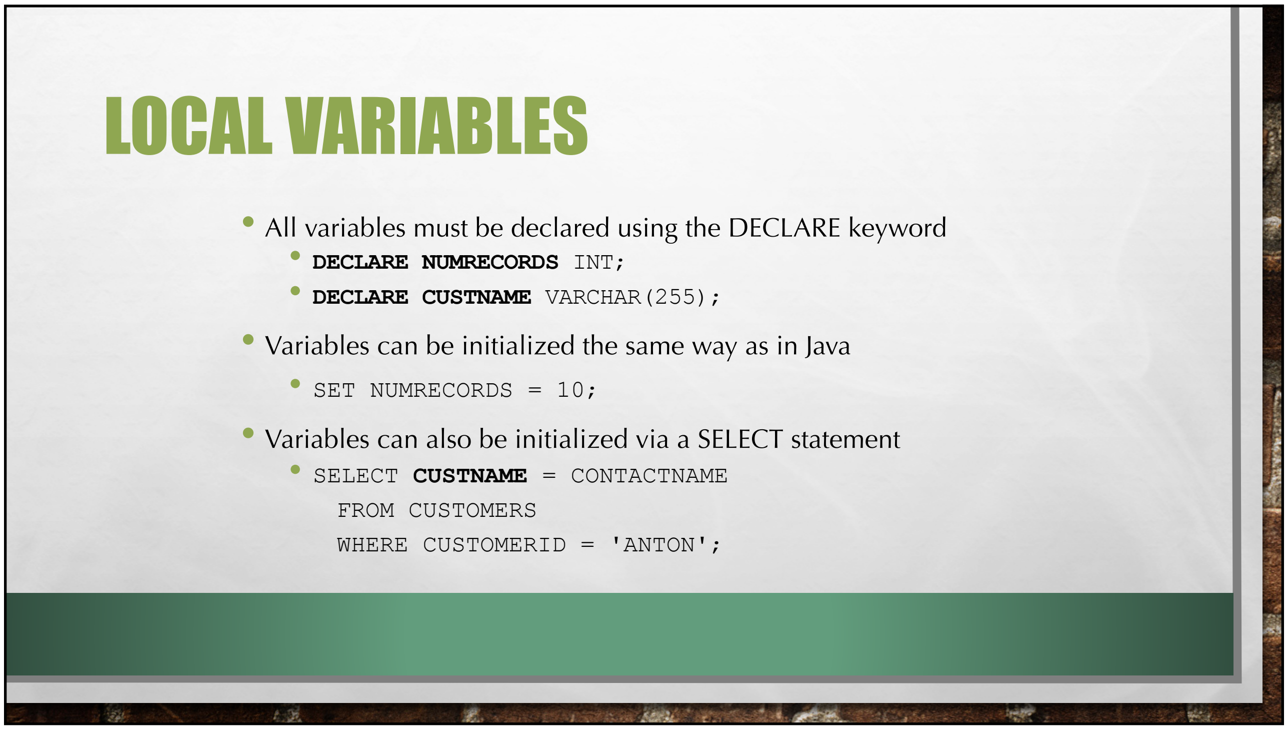
- Segment of code called to complete a task, returning to the point at which it was called. It can have any number of inputs and return any number of values.
- Can be stored in the database and be used by different applications.
- General format of a stored procedure definition is:

```
CREATE PROCEDURE PROCNAME [PARAMETER, … PARAMETER]
BEGIN
     [DECLARE VAR TYPE;]
  -- EXECUTABLE STATEMENTS
END;
```

- parameters are declared as

```
[OUT] VARIABLE DATATYPE
```

4

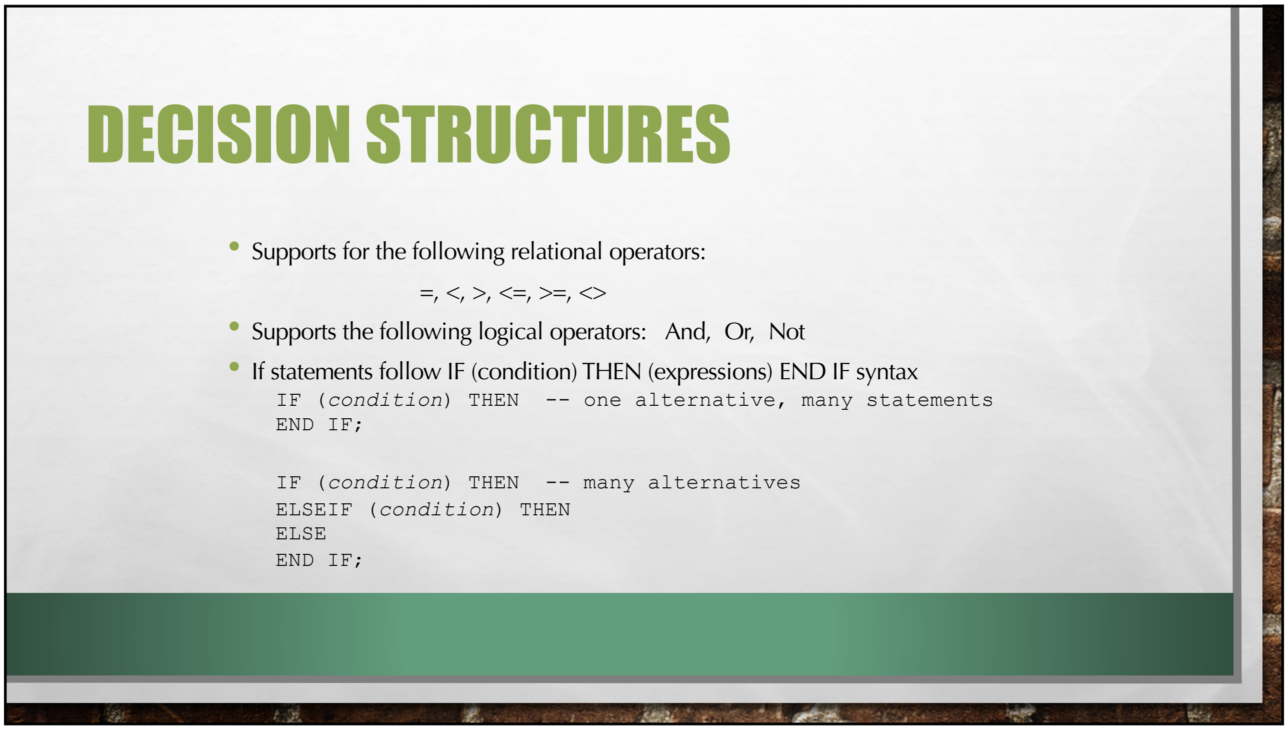


LOCAL VARIABLES

- All variables must be declared using the DECLARE keyword
  - `DECLARE NUMRECORDS INT;`
  - `DECLARE CUSTNAME VARCHAR(255);`
- Variables can be initialized the same way as in Java
  - `SET NUMRECORDS = 10;`
- Variables can also be initialized via a SELECT statement

  -
    ```
    SELECT CUSTNAME = CONTACTNAME
      FROM CUSTOMERS
      WHERE CUSTOMERID = 'ANTON';
    ```

5



DECISION STRUCTURES

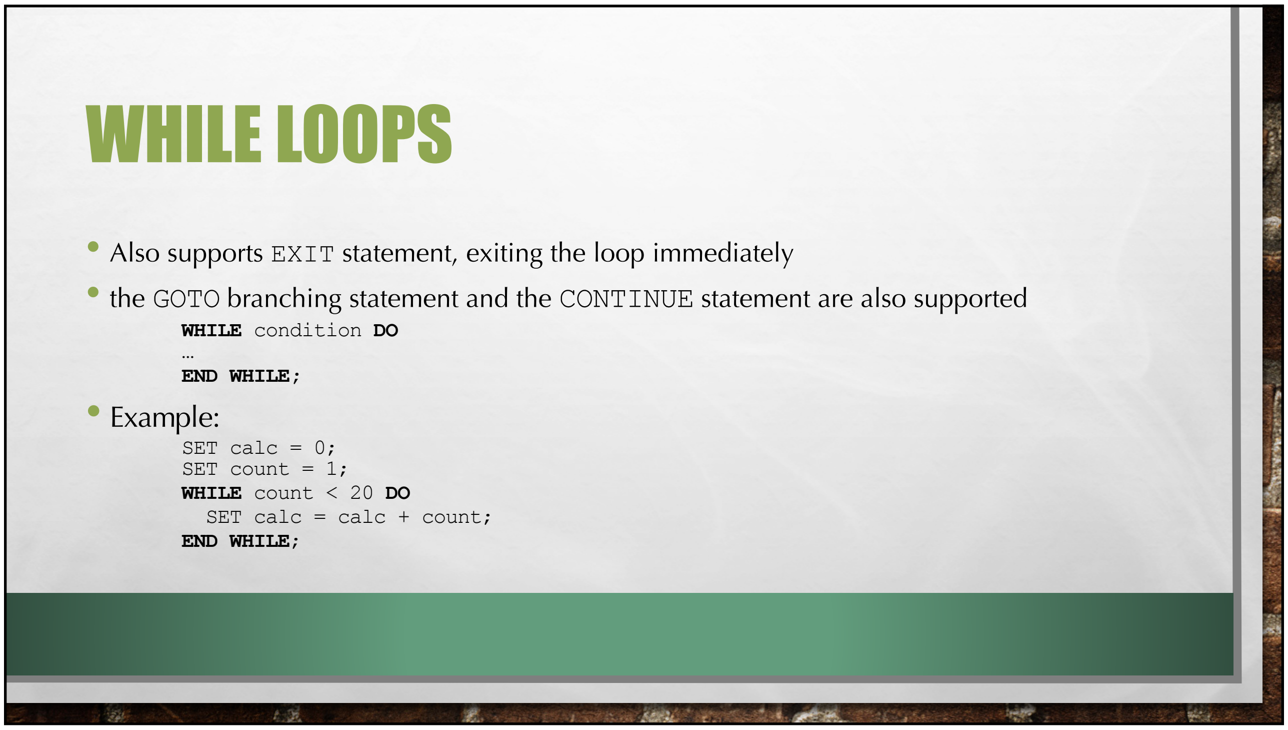
- Supports for the following relational operators:

  =, <, >, <=, >=, <>
- Supports the following logical operators: And, Or, Not
- If statements follow IF (condition) THEN (expressions) END IF syntax

```
IF (condition) THEN  -- one alternative, many statements
END IF;

IF (condition) THEN  -- many alternatives
ELSEIF (condition) THEN
ELSE
END IF;
```

6



WHILE LOOPS

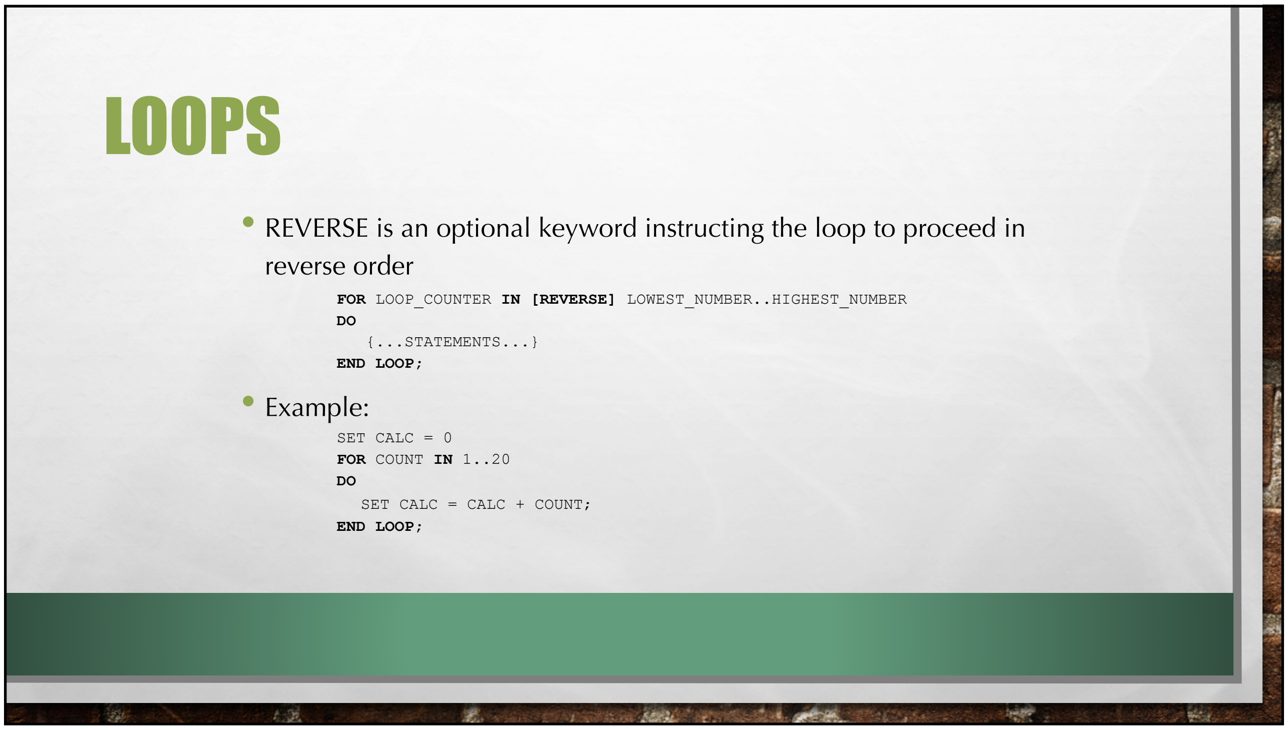
- Also supports `EXIT` statement, exiting the loop immediately
- the `GOTO` branching statement and the `CONTINUE` statement are also supported

```
WHILE condition DO
…
END WHILE;
```

- Example:

```
SET calc = 0;
SET count = 1;
WHILE count < 20 DO
  SET calc = calc + count;
END WHILE;
```

7



LOOPS

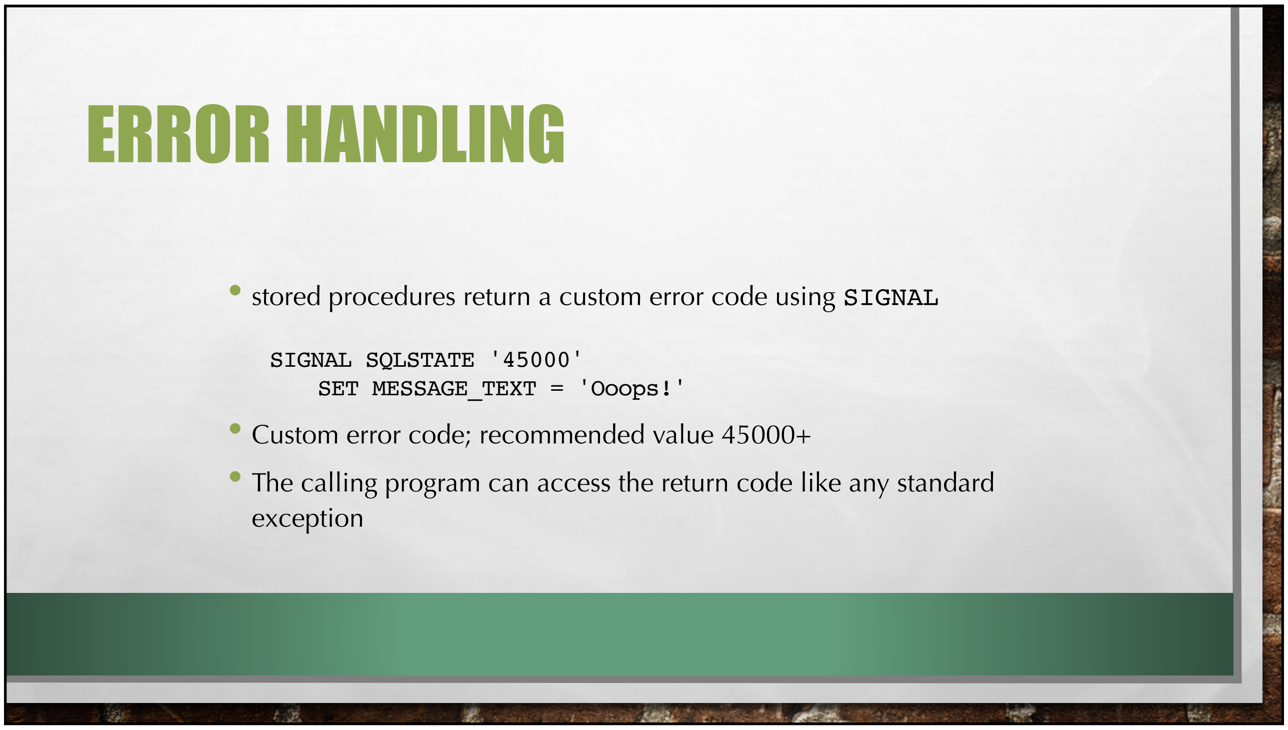
- REVERSE is an optional keyword instructing the loop to proceed in reverse order

```
FOR LOOP_COUNTER IN [REVERSE] LOWEST_NUMBER..HIGHEST_NUMBER
DO
   {...STATEMENTS...}
END LOOP;
```

- Example:

```
SET CALC = 0
FOR COUNT IN 1..20
DO
   SET CALC = CALC + COUNT;
END LOOP;
```

8



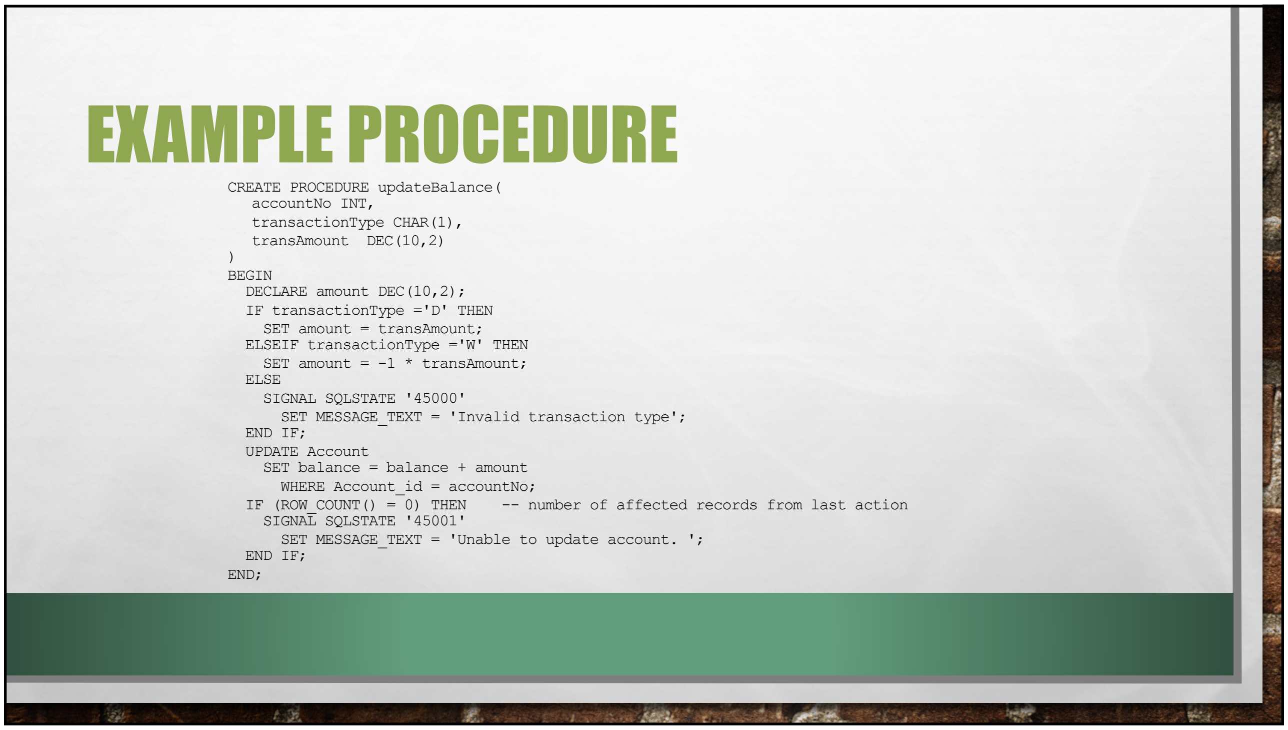
ERROR HANDLING

- stored procedures return a custom error code using `SIGNAL`

```
SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Ooops!'
```

- Custom error code; recommended value 45000+
- The calling program can access the return code like any standard exception

9



EXAMPLE PROCEDURE

```
CREATE PROCEDURE updateBalance(
   accountNo INT,
   transactionType CHAR(1),
   transAmount  DEC(10,2)
)
BEGIN
  DECLARE amount DEC(10,2);
  IF transactionType ='D' THEN
    SET amount = transAmount;
  ELSEIF transactionType ='W' THEN
    SET amount = -1 * transAmount;
  ELSE
    SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Invalid transaction type';
  END IF;
  UPDATE Account
    SET balance = balance + amount
      WHERE Account_id = accountNo;
  IF (ROW_COUNT() = 0) THEN    -- number of affected records from last action
    SIGNAL SQLSTATE '45001'
      SET MESSAGE_TEXT = 'Unable to update account. ';
  END IF;
END;
```

10