PL/SQL

Dr Paolo Guagliardo



THE UNIVERSITY *of* EDINBURGH
informatics

v26.0

Changelog

v26.0 Initial version

Using SQL from applications

Embedded SQL

The use of SQL commands within a **host language** program

All major programming languages provide APIs and drivers to

- establish a **connection** with a database
- execute SQL commands/transactions
- convert SQL data types to appropriate ones

Simple rule: if you know SQL and the host programming language then you know embedded SQL

Cursors

Allow a programming language to operate on collections of rows

Cursors are defined in the SQL standard

Cursor declaration

DECLARE *cursorname* **CURSOR FOR** *query*

A cursor is essentially **pointer** to a row in a table

OPEN : position the cursor just before the first row

FETCH : retrieve the next row

CLOSE : dispose of the cursor

A simple Python program

Interaction with PostgreSQL is handled by the psycopg2 module

```
import psycopg2
# create a connection to the database
conn = psycopg2.connect('dbname=idbcourse \
                         host=pgteach')
# get a cursor
cur = conn.cursor()
# execute an SQL query
q = 'SELECT COUNT(DISTINCT uuid) \
FROM cw0_results'
cur.execute(q)
# print result
print(cur.fetchone()[0])
# close cursor and connection
cur.close()
conn.close()
```

Transactions

For a psycopg2 connection:

- A transaction is started before executing the first command: if `commit()` is not called, changes will be lost
- `rollback()` reverts to the start of any pending transaction
- Closing without committing causes an **implicit rollback**

Examples on laptop

Similar commands are available in other programming languages

Dynamic SQL

Statements where some values are known only at runtime
(e.g., because they are provided by the user)

Example program from laptop

SQL injection

User input can contain malicious SQL statements

- (set appropriate permissions on the database)
- use **read-only** transactions for queries
- always **validate** user input within the application