Tools & Models for Data Science

Imperative SQL

Chris Jermaine & Risa Myers

Rice University

In SQL, We Can Write Imperative Code

? Why is this useful?

In SQL, We Can Write Imperative Code

- Why useful?
  - Encapsulation — make it easy for the programmer
  - Safety — protect the database from the programmer
  - Performance — fewer end-to-end trips
  - Reuse
  - Can process records sequentially
  - Can respond to events

Stored Procedure / Function

- Common form of imperative code: stored procedure and / or function
  - Code stored in the DB
  - Can be invoked from
    - A query
    - An external program
    - Another stored procedure / function
    - A trigger

Basic Form

```
CREATE FUNCTION myFunc(/* list params */)
   RETURNS <datatype> AS
$$
DECLARE
/* declarations here */
BEGIN
/* code here */
END;
$$
LANGUAGE plpgsql ;
```

First Stored Procedure Example

```
PEAK (NAME, ELEV, DIFF, MAP, REGION)
```

? Write a stored procedure to get the peak count in a region

- But if no region given...
- Return the count overall

First Stored Procedure Example

```
PEAK (NAME, ELEV, DIFF, MAP, REGION)

CREATE OR REPLACE FUNCTION getNumPeaks(whichRegion CHARACTER VARYING)
  RETURNS INTEGER AS
$$
DECLARE
        /* declarations here */
BEGIN
        /* code here */
END;
$$
LANGUAGE plpgsql;
```

First Stored Procedure Example

```
CREATE OR REPLACE FUNCTION
   getNumPeaks(whichRegion character varying)
   RETURNS integer AS
$$
DECLARE
  queryString TEXT;
  numPeaks INTEGER;
BEGIN
  numPeaks = 0;
  -- build the query
  queryString = 'SELECT␣COUNT(*)␣FROM␣peak␣' ||
      COALESCE ('␣WHERE␣Region=' ||
         QUOTE_LITERAL(whichRegion) ,'');
  -- run the query
  EXECUTE queryString INTO numPeaks;
  RETURN numPeaks;
END;
$$
LANGUAGE plpgsql;
```

Thoughts on First Stored Procedure Example

- All local variables need to be declared
- Code is bounded by **$$**
- Language is specified at the end
- ? What's the deal with `QUOTE_LITERAL`?
- ? What's `COALESCE`?
- ? `EXECUTE`: common, powerful, dangerous! Why?

First Stored Procedure Example

Then to call

```
SELECT * FROM getNumPeaks('Corocoran to Whitney');
```

and

```
SELECT * FROM getNumPeaks(NULL);
```

Next Stored Procedure Example

- Like before, but now we'll use a cursor

Next Stored Procedure Example

```
CREATE OR REPLACE FUNCTION
   getNumPeaks(whichRegion CHARACTER VARYING) RETURNS INTEGER AS
$$
DECLARE
  myCursor REFCURSOR;
  queryString TEXT;
  numPeaks INTEGER;
BEGIN
  numPeaks = 0;
  -- build the query
  queryString = 'SELECT␣COUNT(*)␣FROM␣peak␣' || COALESCE ('␣WHERE␣Region=' ||
       QUOTE_LITERAL(whichRegion) ,'');
  -- run the query
  OPEN myCursor FOR EXECUTE queryString;
  FETCH  myCursor INTO numPeaks;
  CLOSE myCursor;
  RETURN numPeaks;
END;
$$
LANGUAGE plpgsql;
```

Next Stored Procedure Example

```
BEGIN
  numPeaks = 0;
  -- build the query
  queryString = 'SELECT␣COUNT(*)␣FROM␣peak␣' || COALESCE ('␣WHERE␣Region=' ||
       QUOTE_LITERAL(whichRegion) ,'');
  -- run the query
  OPEN myCursor FOR EXECUTE queryString;
  FETCH  myCursor INTO numPeaks;
  CLOSE myCursor;
  RETURN numPeaks;
END;
```

- What's new here: a "cursor"
  - Standard abstraction for dealing with record sets
  - Essentially an iterator

Steps for Using a Cursor

1. Declare
2. Open
3. Loop
   - 3.1 Fetch
   - 3.2 Check for done
   - 3.3 Repeat from 3.1
4. Close

Cursor Options / Variations

- `FETCH / MOVE` options
  - `NEXT / PRIOR`
  - `FIRST / LAST`
  - `ABSOLUTE / RELATIVE` n
  - `FORWARD / BACKWARD` n
  - ...
- Explicit vs. Implicit

  `FOR` rowData `IN SELECT ... LOOP ... END LOOP`

  `SELECT INTO`...

How to Debug Imperative SQL

- It's primitive
- Use `RAISE NOTICE` 'Step 1';
- Use `RAISE NOTICE` '%', <var>;
- e.g. `RAISE NOTICE` '%', queryString;
- Start simple
- Consider writing data to an external (temporary table)

Wrap up

? How can we use what we learned today?

? What do we know now that we didn't know before?