COMP 543: Tools & Models for Data Science

Imperative SQL 2

Chris Jermaine & Risa Myers

Rice University

More Interesting Cursor Example

? Write a stored procedure giving the name and height of the tallest peak in a region.

```
CREATE OR REPLACE FUNCTION
   getTallestPeak(whichRegion TEXT,
      OUT finalBestName TEXT, finalBestHeight INTEGER) AS
$$
BEGIN
...
END;
$$
LANGUAGE plpgsql ;
```

More Interesting Cursor Example

- Body 1: Declare control variables as well as the cursor

```
CREATE OR REPLACE FUNCTION
   getTallestPeak(whichRegion TEXT,
      OUT finalBestName TEXT, finalBestHeight INTEGER) AS
$$
DECLARE
   peakName TEXT;
   bestName TEXT;
   peakHeight INTEGER DEFAULT -1;
   bestHeight INTEGER DEFAULT -1;
   peakCursor CURSOR FOR SELECT name, elev
        FROM Peak WHERE Region = whichRegion;
BEGIN
END;
$$
LANGUAGE plpgsql;
```

More Interesting Cursor Example

- Body 2: Open cursor and loop to find the tallest peak

```
BEGIN
OPEN peakCursor;
LOOP
        FETCH peakCursor INTO peakname, peakHeight;
        EXIT WHEN NOT FOUND;
        IF peakHeight > bestHeight THEN
                bestHeight = peakHeight;
                bestName = peakName;
        END IF;
END LOOP;
CLOSE peakCursor;
```

More Interesting Cursor Example

- Body 3: return the result

```
finalBestName = bestName;
finalBestHeight = bestHeight;
END;
$$
LANGUAGE plpgsql;
```

- To call:

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

or

```
SELECT getTallestPeak('Corocoran to Whitney');
```

Important: Don’t EVER Write Such Code!

- I wrote code that looped to find the tallest
  - Terrible idea!
  - LIMIT k would be shorter, easier, faster
  - Rule: use AS MUCH declarative code as possible
  - Only use loops, etc. when you MUST
  - Sometimes 3+ orders of magnitude speed difference

Implicit Cursor Version

```
CREATE OR REPLACE FUNCTION
   getTallestPeakImplicit(whichRegion TEXT, OUT finalBestName TEXT,
      OUT finalBestHeight INTEGER)  AS
$$
DECLARE
   bestName TEXT;
   bestHeight INTEGER DEFAULT -1;
   row_data RECORD;
BEGIN
  FOR row_data IN SELECT name, elev FROM Peak WHERE Region = whichRegion
  LOOP
      IF row_data.elev > bestHeight THEN
        bestHeight = row_data.elev;
        bestName = row_data.name;
      END IF;
  END LOOP;
  finalBestName = bestName;
  finalBestHeight = bestHeight;
END;
$$
LANGUAGE plpgsql;
```

Implicit or Explicit?

- Implicit cursors are unique(ish?) to Postgres
  - Less portable
- But they are easier to use
- Limitations?
  - One direction only (?)

Returning Relations

- `RETURNS TABLE`(<list of names, type pairs>)

```
CREATE OR REPLACE FUNCTION
   tableValuedFunctionPLSQL(x INTEGER)
   RETURNS TABLE (f1 INTEGER, f2 TEXT ) AS
$$
DECLARE
        queryString TEXT;
BEGIN
        RETURN QUERY SELECT x, CAST(x AS TEXT) || '␣is␣text';
END;
$$
LANGUAGE plpgsql;

SELECT * FROM tableValuedFunctionPLSQL(5)
```

Returning Relations using SQL language

- `RETURNS TABLE`(<list of names, type pairs>)

```
CREATE OR REPLACE FUNCTION
   tableValuedFunction(x INTEGER)
   RETURNS TABLE (f1 INTEGER, f2 TEXT )
AS
$$
  SELECT x, CAST(x AS TEXT) || '␣is␣text';
$$
LANGUAGE sql;


SELECT * FROM tableValuedFunction(5)
```

Triggers

- Stored procedures that fire in response to some event
- Standard options
  - Events: UPDATE, INSERT, DELETE
  - Timing: BEFORE/AFTER: run only once triggering action succeeds
- Some RDBMSs
  - Triggers: Typically not run when TRUNCATE is called on a table

Trigger Special Variables

- old: table containing old versions of records
- new: table containing new versions

	INSERT	
	Before	After
old.	N/A	N/A
new.	new	new

	UPDATE	
	Before	After
old.	old	old
new.	new	new

	DELETE	
	Before	After
old.	old	old
new.	N/A	N/A

Trigger Example

- Write a trigger that catches updates to peak table, prints an error message and does not process
- Write the function first
- Then create a trigger that calls the function

Trigger: Function Framework

- NOTE: Return type of `TRIGGER`
- NOTE: No arguments

```
CREATE OR REPLACE FUNCTION
   ignorePeakUpdate() RETURNS TRIGGER AS
$$
DECLARE
        ...
BEGIN
        ...
END;
$$
LANGUAGE plpgsql;
```

Trigger: Function Body

```
BEGIN
   RAISE NOTICE 'You␣changed␣the␣height␣of␣\%␣', old.name ;
   RAISE NOTICE 'from␣\%␣to␣\%.', old.elev, new.elev ;
   RAISE NOTICE 'I␣am␣ignoring␣it.';

   RETURN OLD;
END;
```

- NOTE: Multiple lines used here only for formatting reasons. Can all be done on one line

Trigger Example

- Write a trigger that catches updates to peak table, prints error message and does not process

```
CREATE TRIGGER checkHeight
BEFORE UPDATE ON peak
FOR EACH ROW
EXECUTE PROCEDURE ignorePeakUpdate()
```

Temporary Tables

```
CREATE TEMPORARY TABLE myMap (
  myKey INTEGER,
  myValue VARCHAR(200),
  PRIMARY KEY(myKey)
);
```

- When to use
  - Passing lots of data
  - Short life span (current session only)
  - Debugging

Questions?