COMP 330: Imperative SQL 1

Chris Jermaine and Kia Teymourian
Rice University
In SQL, Can Write Imperative Code

Why useful?
In SQL, 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
- Can respond to events

Note: we focus on TSQL. Why?
Stored Procedure

Common form of imperative code: stored procedure

- Procedure whose code is stored in the DB
- Can be invoked from the command line, external program
- Or from another stored procedure, trigger, function
CREATE PROCEDURE procName
/* list params */
AS BEGIN
/* code here */
END;
First Stored Procedure Example

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

Ex: Write a stored procedure to get the tallest peak in a region

- But if no region given...
- Return the tallest peak overall
First Stored Procedure Example

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

CREATE PROCEDURE getNumPeaks

/* list params */
@whichRegion VARCHAR (8000) = NULL

AS BEGIN
/* code here */
END;
First Stored Procedure Example

CREATE PROCEDURE getNumPeaks

/* list params */
@whichRegion VARCHAR (8000) = NULL

AS BEGIN
DECLARE @queryString VARCHAR (8000);

SET @queryString = 'SELECT COUNT(*) FROM peak' +
ISNULL(' WHERE region = ''' + @whichRegion + '''', ' ');

EXECUTE (@queryString);
END;

▷ All local vars need a DECLARE
▷ Why the @ symbol?
▷ What’s the deal with ”” and ””?
▷ EXECUTE: common, powerful, dangerous!
First Stored Procedure Example

Then to call:

```sql
EXECUTE getNumPeaks
    @whichRegion = 'Corocoran_to_Whitney';
```

▷ BTW: what’s the deal with GO?
Next Stored Procedure Example

Like before, but now we’ll use:

▷ A call-by-reference parameter
▷ A cursor

```
CREATE PROCEDURE getNumPeaks
@whichRegion VARCHAR (8000),
@result INT OUTPUT
AS BEGIN
/* code here */
END
```

▷ What’s the deal with OUTPUT?
Next Stored Procedure Example

CREATE PROCEDURE getNumPeaks
@whichRegion VARCHAR (8000),
@result INT OUTPUT
AS BEGIN

DECLARE myRes CURSOR FOR
SELECT COUNT(*) FROM peak WHERE region = @foo;

OPEN myRes;
FETCH myRes INTO @result;
CLOSE myRes;
DEALLOCATE myRes;
END
Next Stored Procedure Example

AS BEGIN
DECLARE myRes CURSOR FOR
SELECT COUNT(*) FROM peak WHERE region = @foo;

OPEN myRes;
FETCH myRes INTO @result;
CLOSE myRes;
DEALLOCATE myRes;
END

What’s new here: a “cursor”

▷ Standard abstraction for dealing with record sets
▷ Essentially an iterator
▷ What’s the difference between CLOSE and DEALLOCATE?
Next Stored Procedure Example

Then to call the procedure:

```sql
DECLARE @myResult INT
EXECUTE getNumPeaks
    @whichRegion = 'Kaweas_and_West',
    @result = @myResult output;
PRINT @myResult;
```
Questions?