COMP 330/543: Imperative SQL 1

Sinan Kockara

Luis Guzman

Chris Jermaine

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
  Dynamic queries

Note: we focus on TSQL. Why?

Example – Imperative SQL

Data about mountain peaks

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

NAME	ELEV	DIFF	MAP	REGION
Pilot Knob S	6200	2	Onyx	Southern Sierra
Owens Peak	8453	2	Owens Peak	Southern Sierra
Spanish Needle	7841	3	Lament Peak	Southern Sierra
Lamont Peak	7429	2	Lament Peak	Southern Sierra
Sawtooth Peak S	8000	2	Ninemile Canyon	Southern Sierra
…	…	…	…	…

Example – Imperative SQL

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

Write a function to get the peak count in a given region

If no region given, return the count overall

Can we do this in declarative SQL ?

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

Basic Form

```
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 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 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!

Injection example

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

If attacker provided this input:

```
'Southern Sierra'; DROP TABLE PEAK; --
```

It would result in this query:

```
SELECT COUNT(*) FROM PEAK WHERE region =
'Southern Sierra';
DROP TABLE PEAK; --';
```

First Stored Procedure Example

Then to call:

```
EXECUTE getNumPeaks
        @whichRegion = 'Corocoran to Whitney';
```

- BTW: what's the deal with GO?

```
CREATE VIEW MyView AS SELECT * FROM Employees
WHERE Department = 'Sales';
SELECT * FROM MyView;

vs.

CREATE VIEW MyView AS SELECT * FROM Employees
WHERE Department = 'Sales';
GO
SELECT * FROM MyView;
```

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 = @whichRegion;

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 = @whichRegion;

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:

```
DECLARE @myResult INT
EXECUTE getNumPeaks
        @whichRegion = 'Kaweahs and West',
        @result = @myResult output;
PRINT @myResult;
```

Don’t do this!

```
CREATE PROCEDURE getNumPeaks_silly
@whichRegion    VARCHAR    (8000),
@result INT OUTPUT
AS BEGIN
SET @result = 0;
DECLARE myRes CURSOR FOR
SELECT 1 FROM peak WHERE region = @whichRegion;
OPEN myRes;
DECLARE @d INT;
FETCH NEXT FROM myRes INTO @d;
WHILE @@FETCH_STATUS = 0
    BEGIN
        SET @result += 1;
        FETCH NEXT FROM myRes INTO @d;
    END
CLOSE myRes;
DEALLOCATE myRes;
END
```

Questions?