SECTION 1: INTRODUCTION

CSE 444

January 6th, 2011

Today

- Intro
- IISQLSRV and Management Studio
- SQL practice
- Project 1

About section and the TAs

- Section in EEB 025 on Thursdays
  - AA: 8:30-9:20, AB: 9:30-10:20
  - Feel free to come to either section

- Me: Rita Sodt
  - rsodt@cs.washington.edu
  - Office hours: Thurs. 10:30-noon in CSE 220 (might switch to 006)
- Liem Dinh
  - liemdinh@cs.washington.edu
  - Office hours: Tues. 1:30-3 in CSE 220

Let’s introduce ourselves

Your name

Something interesting about you

Connecting to SQL Server Management Studio

IISQLSRV connection settings

- Server: iisqlsrv.cs.washington.edu
- Use *SQL Server Authentication*
- Username: your UW email @u.washington
- Password: tmp_PASS
  - Also in the first mail to class list
  - You'll have to change it on first login

IMDB database

Actor (*id*, fname, lname, gender)

Movie (*id*, name, year, rank*)

Directors (*id*, fname, lname)

Casts (pid, mid, role)

Movie_Directors (did, mid)

Genre (genre, mid)

* *currently unused, always null*

Basic SQL

SELECT	column(s) (optional DISTINCT)
FROM	table(s)
WHERE	condition(s)
GROUP BY	column(s)
HAVING	special_condition(s)
ORDER BY	column(s)

Aggregate functions (count, sum, avg, min, max)

A simple query

```
SELECT *
FROM Movie
WHERE name = 'Star Wars: Episode V - The Empire
 Strikes Back';
```

More examples

1. Names of all Star Wars movies
2. All Star Wars movies made in 2000 or later
3. Names and production years of all Star Wars movies from earliest to latest

“Star Wars movie” = movie with “Star Wars” in the name

Can use LIKE condition for pattern matching.

% matches any string of any length

_ matches any single character

Something a little harder…

Who directed The Empire Strikes Back?

Hint: joins!

Who directed The Empire Strikes Back?

Movie (*id*, name, year, rank)

Directors (*id*, fname, lname)

Movie_Directors (did, mid)

Need to *join* (combine) the data from these tables!

Director of Empire Strikes Back

Director of Empire Strikes Back

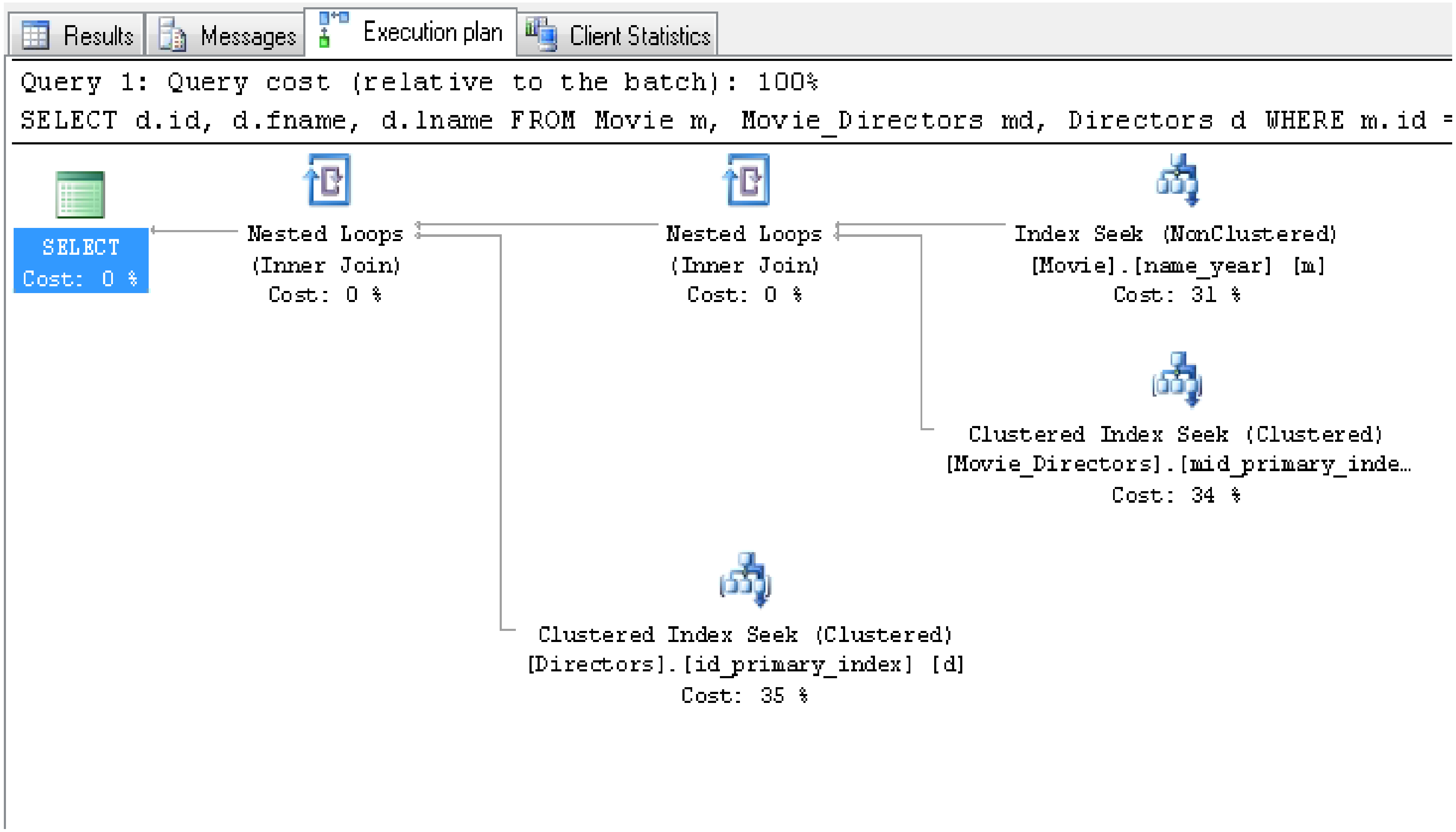
```
SELECT d.id, d.fname, d.lname
FROM Movie m, Movie_Directors md, Directors d
WHERE m.id = md.mid AND
 md.did = d.id AND
 m.name = 'Star Wars: Episode V - The Empire
 Strikes Back';
```

Director of Empire Strikes Back

```
SELECT d.id, d.fname, d.lname
FROM Movie m, Movie_Directors md, Directors d
WHERE m.id = md.mid AND
 md.did = d.id AND
 m.name = 'Star Wars: Episode V - The Empire
 Strikes Back';
```

$\left.\begin{array}{l}\texttt{m.id = md.mid AND}\\ \texttt{md.did = d.id AND}\end{array}\right\}$ Join conditions

SQL server: examine query plan



```
Query 1: Query cost (relative to the batch): 100%
SELECT d.id, d.fname, d.lname FROM Movie m, Movie_Directors md, Directors d WHERE m.id =
```

SELECT
Cost: 0 %
Nested Loops
(Inner Join)
Cost: 0 %
Nested Loops
(Inner Join)
Cost: 0 %
Index Seek (NonClustered)
[Movie].[name_year] [m]
Cost: 31 %
Clustered Index Seek (Clustered)
[Movie_Directors].[mid_primary_inde…
Cost: 34 %
Clustered Index Seek (Clustered)
[Directors].[id_primary_index] [d]
Cost: 35 %

Aggregates

Sometimes we just want summary or extreme-case data.

Examples:

- All Star Wars movies → number of Star Wars movies
  - SELECT * FROM Movie … → SELECT COUNT(*) FROM Movie…
- Dates of all movies → date of earliest movie
  - SELECT year FROM Movie … → SELECT MIN(year) FROM Movie …

Aggregates and grouping

- List actors’ first names and their frequencies, from most to least popular:

Aggregates and grouping

List actors’ first names and their frequencies, from most to least popular:

```
SELECT fname, COUNT(*) AS freq
FROM Actor
GROUP BY fname ← grouping by first name
ORDER BY freq DESC;
```

Aggregates and grouping

List actors' first names and their frequencies, if they occur more than 1000 times, from most to least popular:

```
SELECT fname, COUNT(*) AS freq
FROM Actor
GROUP BY fname
HAVING COUNT(*)>1000 ← more than 1000 times
ORDER BY freq DESC;
```

* note: aggregate can't occur in WHERE clause

Project 1

More fun with the IMDB database!

Some queries need more advanced SQL

Posted now, due January 21st

This weekend: log in to IISQLSRV! If you can't, email me: rsodt@cs

Project 1

- **Read the questions carefully**
- Don’t put SELECT statement inside an aggregate function, only attributes and *
    - Bad: SELECT AVG (SELECT … FROM …)
    - Good: SELECT AVG(attribute) or SELECT AVG(*)
- TOP(1) is disapproved of when we ask for “largest/highest/etc.” result because we should allow to return multiple rows in the case of a tie.

Project 1

- Compare number of rows in your results to the numbers listed on the assignment (some variance may be acceptable)
- All queries should run within 2 minutes (BUT sometimes SQL server is slow if a lot of people are using it and executing inefficient queries)
- If queries are too slow try restructuring/simplifying them and/or examine the query plan in SQL server