PROG8080
SELECT Part 2

Glenn Paulley

Fall 2015

Examples

- Previously, you did a ‘dry run’ for assignment 1
  - We created an sql*nnn*e1.sql file
  - We added header comments
  - We identified each example SQL statement with a comment:

```
-- 1.
PRINT ' *** QUESTION 1 *** '
PRINT ' '
(one blank line)
SELECT…
```

Case Sensitivity

- Case sensitivity is determined at the server level when Microsoft SQL Server is installed
  - By default MS SQL Server is case insensitive (like Visual Basic)
  - The installer can opt for case sensitivity (like C/C++/C# and Java)
- Our installation is case insensitive

NULL

- The NULL keyword means “undefined”
- NULL is a value distinct from 0 or 0.0, an empty string (‘’), or a blank string (‘ ‘)
- Predicates (conditions) involving NULL evaluate to UNKNOWN
- SQL uses three-valued logic:
  - Anything compared to NULL evaluates to UNKNOWN
  - NOT UNKNOWN yields UNKNOWN
  - TRUE OR UNKNOWN yields TRUE
  - TRUE AND UNKNOWN yields UNKNOWN
  - FALSE OR UNKNOWN yields UNKNOWN
  - FALSE AND UNKNOWN yields FALSE

CONESTOGA
Connect Life and Learning

NULL

- Try this SQL statement:

  SELECT campusCode, reportsTo, schoolCode

  FROM Employee

  WHERE number = 2117745

- Now try this SQL statement:

  SELECT campusCode, reportsTo, schoolCode

  FROM Employee

  WHERE number = 5512736

IS NULL predicate

- Use the IS NULL predicate in the query's WHERE clause to select rows with NULL values for particular attributes:

```
SELECT *
FROM Employee
WHERE schoolCode IS NULL
```

IS NOT NULL predicate

- Use IS NOT NULL in a search condition to retrieve rows with non-NULL values:

```
SELECT *
FROM Employee
WHERE schoolCode IS NOT NULL
```

NULL AND =, !=

- You *CANNOT* use:

  = NULL instead of IS NULL

  != NULL instead of IS NOT NULL

- Try it:

  … WHERE schoolCode = NULL (wrong!)

  … WHERE schoolCode != NULL (wrong!)

IN predicate

- Instead of OR…

```
SELECT *
FROM Person
WHERE firstName = 'John'  OR firstName = 'Jon‘;
```

- …you can use an IN predicate :

```
SELECT *
FROM Person
WHERE firstName IN ('John' , 'Jon' );
```

- IN can be negated using NOT, as in NOT IN (‘John’, ‘Jon’)

IN predicate

- For readability, IN is preferred when you are working with more than two values:

  ...WHERE state IN (‘CA‘ , ‘CO’, ‘NV’)

- Rather than:

  … WHERE state = ‘CA’ OR state = ‘CO’ OR state = ‘NV’

- But the two constructions are equivalent

BETWEEN predicate

- Instead of >= AND <= …

```
SELECT id, lastName, firstName
FROM Person
WHERE number >= 1110000 AND number <= 1200000
```

- … you can use a BETWEEN predicate:

```
SELECT id, lastname, firstname
FROM Person
WHERE number BETWEEN 1110000 AND 1200000;
```

BETWEEN predicate

- The comparison is inclusive

  …WHERE id BETWEEN 9000 AND 9200

- A row with id = 9000 or a row with id = 9200 would be included in the result set

LIKE predicate

- Use a LIKE predicate to perform basic pattern matching
  - Is part of the ISO SQL Standard
  - There are some specific behaviours with the LIKE predicate in SQL Server due to legacy semantics
  - Syntax: <expression> LIKE <pattern>
- Literal characters must be present in the given position

```
SELECT firstName, lastName
FROM Person
WHERE firstName LIKE 'John'
ORDER BY firstName, lastName
```

- As with string comparisons, LIKE uses case-insensitive character comparisons with a case-insensitive database

LIKE predicate – wildcard characters

- An underscore character (_) matches one arbitrary character in the given position

```
SELECT firstName, lastName
FROM Person
WHERE firstName LIKE 'Joh_'
ORDER BY firstName, lastName
```

LIKE predicate – wildcard characters

- A percent character (%) matches zero, one or more characters starting with the given position

```
SELECT firstName, lastName
FROM Person
WHERE firstName LIKE 'Mar%'
ORDER BY firstName, lastName
```

LIKE patterns

- You can repeat and combine % and _ as needed in a LIKE predicate pattern

```
SELECT firstName, lastName
FROM Person
WHERE firstName LIKE '%Mar%'
ORDER BY firstName, lastName
```

NOT (again!)

- Remember that you can use NOT with the predicates just described:
  - ...NOT IN
  - ...NOT BETWEEN
  - ...NOT LIKE
  - …IS NOT NULL