CSC 355 Database Systems
Lecture 4

Eric J. Schwabe

School of Computing, DePaul University

Spring 2020

Topic:

- Basic SQL Queries

Displaying a Table’s Contents

SELECT * FROM *TABLE* ;

- This is an example of the simplest form of a *query* – retrieval of information from one or more tables
  - SELECT * : “Show all attributes…”
  - FROM *TABLE* : “…from the indicated table”

SQL Queries

- General form of a query:

SELECT *list of expressions*
    FROM *set of rows*
    [WHERE *condition on rows*]
    [GROUP BY *grouping attributes*]
    [HAVING *condition on groups*]
    [ORDER BY *ordering attributes*] ;

- Result is an ordered set of ordered tuples

FROM

… **FROM** ***set of rows*** **…**

- FROM indicates the set of rows from which information will be retrieved
  - a single table (that’s all we’ll use for now…)
  - a list or other combination (“join”) of tables
  - the result of a “subquery”

SELECT

SELECT *list of expressions* …

- SELECT indicates what information will be displayed
  - values of attributes
  - expressions computed from attributes
  - functions applied to attributes

SELECT

- Displaying attributes:
  - * lists all attributes
  - separate attributes in the list with commas
  - attributes can be renamed in result using AS
  - DISTINCT will remove duplicate rows
- Displaying expressions:
  - Can combine attributes with +, -, *, /, ||

SELECT

- Displaying functions of attributes:
  - Numbers: mod(*a*,*b*), power(*m*,*n*), round(*x*,*i*)
  - Strings: upper(*s*), lower(*s*), substr(*s*,*p*,*l*)
  - Dates: sysdate, to_char(*d*, *field*) with field being, e.g., ‘YYYY’, ‘YY’, ‘YEAR’, ‘MM’, ‘MON’, ‘DD’, ‘DY’… or a combination …
- SQL has many built-in functions…

WHERE

… **WHERE** ***condition*** **…**

- Each row is tested against the condition, and only those that satisfy it are returned by the query
- Condition expression can contain:
  - comparisons
  - expressions with wildcards (for strings)
  - logical operators

Comparisons

- Put numerical or string or date value on each side, comparison returns true or false

=	is equal to
!= or <>	is not equal to
>	is greater than
>=	is greater than or equal to
<	is less than
<=	is less than or equal to

Comparisons

- Numbers and dates are compared in the usual way (smaller < larger, earlier < later)
- String values are compared according to *lexicographic (dictionary)* order
  - Compare strings character by character until they differ
  - The string with the earlier character (by ASCII order) where they first differ is smaller

Wildcards

- Using LIKE, we can compare character strings to strings that include wildcard characters that match anything:

_	matches any single character
%	matches any consecutive set of characters

- For example:
  - ‘b_d’ will match ‘bad’, ‘bed’, but not ‘band’
  - ‘bat%’ will match ‘bat’, ‘bath’, ‘battery’…

Logical Operators

- Simple conditions can be combined into more complicated conditions
  - *X* AND *Y* is satisfied by a tuple if and only if both *X* and *Y* are satisfied by it
  - *X* OR *Y* is satisfied by a tuple if and only if at least one of *X* and *Y* is satisfied by it
  - NOT *X* is satisfied by a tuple if and only if *X* is not satisfied by it

Dealing With NULLs

- Any arithmetic expression involving a NULL will yield NULL (as will most functions)
- To replace NULLs in output, use the function NVL(*expr1*, *expr2*)
  - If *expr1* is not NULL, will display *expr1*
  - If *expr1* is NULL, will display *expr2* instead
  - e.g., SELECT NVL(phone, ‘no phone given’) …

Dealing With NULLs

- Any comparison involving a NULL will yield UNKNOWN
  - Use IS NULL (not =) to check if a value is NULL
  - There are extended definitions of AND, OR, and NOT that include UNKNOWN
  - UNKNOWN will <u>not</u> satisfy a WHERE test
  - UNKNOWN <u>will</u> satisfy a CHECK condition

ORDER BY

… ORDER BY *list of ordering attributes*

- Tuples are sorted by the first attribute in the list
  - Ascending order is the default, DESC after attribute indicates descending order instead
- Ties are broken by the second attribute (if any), then the third attribute (if any), et cetera

Writing a Query

1. FROM: What table should I use?
2. WHERE: How do I indicate which rows to include in the result?
3. ORDER BY: How should I sort the rows in the output?
4. SELECT: What values do I have to compute and display?

Solving a Query Problem

1. Before you write the query:
   - Read the problem carefully to be sure you understand it, and clarify where necessary
   - Look at the data and work it out by hand, then think about how you did it
2. Write the query:

First FROM (with SELECT *), then WHERE, then ORDER BY, then SELECT

Solving a Query Problem

3. Test the query:
   - If there are syntax errors, go back to 2. to correct them
   - Look at the result against what you did by hand
   - If the result is not correct, go back to 2. and re-examine the query against your result and your interpretation of the problem – describe as clearly as you can precisely <u>how</u> it is incorrect

Next:

- More SQL Queries