SQL Book Notes 2/19/21
6.1 - 6.5

Simple Queries:

* SELECT, FROM, WHERE

ex.

SELECT * FROM tableName WHERE condition(s);

* * - all

Projection in SQL:

* get specific table values from the select clause w/ condition

ex.

SELECT title FROM tableName WHERE condition(s);

* If we want the column headers to be different from the attributes of relation mentioned in the FROM clause
  => SELECT title AS name . . . .

- The columns of the resulting table will be named differently

☆ You can also substitute for expression

ex.

SELECT length*0.1667 AS lengthInHours ....

- This will modify the values in the column that are returned

☆☆ SQL is case insensitive but case sensitivity is common practice

☆ You can also place constants in place of attribute names
=) for style purposes

Selection in SQL:

☆ Comparison operators =, <>, <, >, <=, >=, != (!=)

☆ || - concatenation operator

☆ Can apply arithmetic operators to expressions

☆ Strings are denoted ' '

☆ AND, OR, NOT

Comparison of Strings:

- ☆ Strings are equal if they are the same sequence of chars
- ☆ Can compare between CHAR and VARCHAR
- ☆ using arithmetic operators compares in lexicographical order

Pattern Matching in SQL:

ex.

s LIKE p

- s is a string
- p is a pattern

☆ can use special char %, _

☆ Ordinary chars match only themselves in p but % matches 0 or more and _ matches exactly one char in s.

Dates and Times:

☆ DATE 'yyyy-mm-dd'

☆ TIME 'hh:mm:ss.ss'

☆ TIMESTAMP 'date␣time'

Null values and Comparisons involving Null:

Meanings

1. Value unknown
2. Value inapplicable
3. Value withheld

Operating on NULL

1. using arithmetic operators => output NULL
2. Comparing => output UNKNOWN

☆ Cannot use NULL explicitly as an operand

☆ To look for NULL => var IS NULL
... IS NOT NULL

Truth-value UNKNOWN:

1. AND => UNKNOWN if neither is FALSE but at least one is UNKNOWN
2. OR => UNKNOWN is neither is TRUE but at least one is UNKNOWN

3. NOT => NOT UNKNOWN = UNKNOWN

☆Only values of TRUE will show up in a table

Ordering the Output!

ex.

ORDER BY <list of attributes>

☆Ascending order default (ASC)

☆sort descending order by appending DESC to an attribute(s)

Queries Involving More than One Relation:

☆we can use set operations directly

☆can refer to multiple tables at once

ex.

SELECT ---- FROM tablename, more tables...

Disambiguating Attributes:

☆Can refer to attributes with similar

names across multiple tables using . operator

ex. tableName.attributeName

Tuple Variables:

- can create instances of variables

ex.

```
SELECT Star1.name, Star2.name FROM
  MovieStar Star1, MovieStar Star2 WHERE
    Star1.address = Star2.address;
```

Interpreting Multirelation Queries:

- See documentation for FOR loops
- Parallel assignment
- Conversion to relational algebra

Union, Intersection, and Difference of Queries:

- UNION, INTERSECT, EXCEPT

ex. (query1)
UNION
(query2)

Subqueries

- A query that is part of another query

Subqueries can:

1. return a single constant that can be compared with another value
2. return relations that can be used
3. appear in FROM clause

Subqueries that produce Scalar Values:

ex.

```
. . . .
WHERE cert #=( SELECT
. . . . .
);
```

Conditions Involving Relations:

1. EXISTS R is a condition that is true iff R is not empty

2. s IN R is true iff s = one of the values in R

3. s > ALL R is true iff s > all values in R

☆ Any other conditional operator is analogous to this definition

4. s > ANY R is true iff s > at least one value in R

☆ ALL can be negated using NOT

Conditions Involving Tuples:

☆ Double nested subqueries

Correlated Subqueries:

- when subqueries need to be evaluated many times and a tuple is a term in the subquery that comes from outside the subquery

☆ pay attention to scope

Subqueries in FROM Clause:

☆ place query after FROM clause and give it a name

SQL Join Expressions:

- CROSS JOIN - cartesian product of two tables

☆ Just place the tables side by side

ex.

tableName CROSS JOIN tableName

- theta join - join tables ON a condition

ex.

tableName JOIN tableName
ON condition

Natural Joins:

Differs from theta-join in that:

1. The join condition is that all pairs of attributes from the two relations having a common name are equated, and there are no other conditions
2. One of each pair of equated attributes is projected out

ex.

tableName NATURAL JOIN tableName

Outer Joins:

* Pads dangling tuples from both of it's args

ex.

tableName NATURAL FULL OUTER JOIN tableName

* can also add LEFT or RIGHT
* Can do theta-outer-join
  => discontinue NATURAL and use ON

Full-Relation Operations:

Eliminating Duplicates:

* use "DISTINCT attributeName" to get only one return of multiple matching tuples

Duplicates in Set Operations:

* put ALL after any operation to not exclude duplicates

Grouping and Aggregation in SQL:

Aggregation Operators:

- SUM, MIN, MAX, AVG, COUNT "(*)"

ex.

..... AVG(attributeName) .....

Grouping:

- GROUP BY
- follows WHERE clause
- SELECT can have aggregates or attributes

Grouping, Aggregation, and Nulls

- Null is ignored in any aggregation
- Null treated as ordinary value when forming groups
- Count of empty bag is 0, all other aggregates will produce Null

★★★ Clause order:

SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY

Having Clauses:

Rules: ★Follows GROUP BY

1. Aggregation only applies to the tuples of the group being tested
2. Any attribute of relations in FROM clause may be aggregated in the HAVING clause, but only those attributes in the GROUP BY list may appear unaggregated in the HAVING clause

Database Modifications:

Modifications:

1. Insert
2. Delete

3. Update

Insertion:

ex.

```
INSERT INTO R(A1,...,An)
        VALUES (v1,....,vn);
```

*Order matters

Deletion:

ex.

```
DELETE FROM tablename
    WHERE condition
```

Updates:

ex.

```
UPDATE R SET <new val assignments>
        WHERE condition
```