Chapter 3. Introduction to SQL

3.1) Overview of the SQL Query Language

i) Data-definition language (DDL)
- Define/Delete/Modify relational schemas
- Specify integrity constraints
- View definitions
- Specify access rights (authorization)

ii) Data-manipulation language (DML)
- Query, insert, modify, delete tuples

iii) Transaction Control (Beginning & End of transactions)

iv) Embedded/Dynamic SQL (Using SQL with general purpose languages.)

3.2) SQL Data Definition

DDL allows specifications of relations and metainfo on each of them.

CREATE TABLE, DELETE FROM / DROP TABLE, ALTER TABLE
(& delete rows) (& delete both rows and schema)

i) Schema for each relation
- PRIMARY KEY: has to be nonnull & unique
- FOREIGN KEY: has to be another relation's primary key

ii) Types of values associated with each attribute
- null: indicates absentness. (unknown or may not exist)
- Several built-in types

iii) Integrity Constraints
- NOT NULL construct.

iv) Information on the set of indices to be maintained for each relation.

v) Security and authorization info for each relation

vi) Physical storage structure of each relation.

3.3 Basic Structure of SQL Queries.

Produce a new relation with
the relations listed in the FROM clause,
operates and specifies in the WHERE and SELECT clause.

i) DISTINCT vs. (implicit) ALL
Duplicate tuples are not allowed in relations,
but elimination is time consuming, so SQL allows duplicates.

ii) Logical connectives and comparison operators
& AND, OR, NOT
& <, <=, >, >=, <> → Not equal

iii) SELECT
List the attributes desired in the result of query.

iv) FROM
List of the relations to be accessed.
By itself defines a cartesian product, most general case of join.

v) WHERE
Predicates involving attributes of the relation in the FROM query.

vi) NATURAL JOIN USING
Joins tuples of only those with the same values
on those attributes that appears on both relations.
So, attributes are not repeated in the output relation.

(34) Additional Basic Operators

i) AS

Renames the name of attributes or relations.

Solves the problem of i) same name attributes among different relations
ii) renaming results of arithmetic expressions in SELECT.
iii) Comparing tuples in the same relation

Remark. Correlation name (Table alias, correlation variable, tuple variable)
Alternative names used for the same relation.

ii) String operators

- LOWER, UPPER, TRIM (or trims trailing whitespaces), || (Concatenate two strings)
- LIKE: compares two SQL pattern strings.
  - % (percent): matches any substring,
  - _ (underscore): matches any character.
  - \ (backslash): escape character.
- SIMILAR TO: Enables regular expressions.

iii) * (asterisk)

Used in select clause to denote "all" attributes

iv) ORDER BY

controls over the order tuples are displayed.
Specify sort order by ASC (ascending) or DESC (descending)

v) BETWEEN / NOT BETWEEN

Simplifies comparison operators used inside predicates in WHERE.

vi) Tuple notations

notation $(V_1, V_2, \cdots V_n)$ to denote tuples of arity n to use on comparison.

3.5 Set operations

i) UNION / INTERSECT / EXCEPT
Equivalent to mathematical set theory operation $\cup$ / $\cap$ / $-$

ii) UNION ALL / INTERSECT ALL / EXCEPT ALL
Retains all duplicates in set operations.

3.6 Null Values

- The result of all arithmetic expression containing null is null.
- SQL treats null as unknown,
  so every boolean expression containing null is null
  except (true OR null) = true, (false AND null) = false.
- Note that (null = null) = true only on duplication removal (namely, DISTINCT)

3.7 Aggregate Functions

- Basic aggregation functions
  take a collection (a set or multiset) of values and return a single value.
  AVG, MIN, MAX, SUM, COUNT, SOME, EVERY
  ↳ ignores null values. ↳ only on boolean
- DISTINCT
  Eliminates duplicates while aggregating
- GROUP BY
  Form groups for aggregation. Entire relation is one group if omitted.
  Only attributes present in group clause can appear in SELECT
  without being aggregated.
- HAVING
  State a condition (predicate) that applies to groups rather than to tuples.

3.8) Nested Subqueries

i) Set membership : IN / NOT IN connective.

Tests for set membership / absence in set membership.

ii) Set comparison : ALL / SOME (synonymous to ANY)

Comparisons for every tuple in set. ex) >= ALL

iii) Empty relations : EXISTS construct

Returns true if the argument subquery is nonempty.

iv) Absence of duplicate tuples : UNIQUE construct

Returns true if the argument subquery contains no duplicate tuples.

v) Subqueries inside FROM clause

select-from-where returns a new relation, so it can be inserted anywhere relations can appear.

Note that LATERAL keyword enables access to attributes of preceding Tables or subqueries in the FROM clause.

vi) WITH clause (Common Table Expressions, or CTE)

Defines a temporary relation whose definition is available only to the query in which the WITH clause appears.

Remark Compared to nested queries, using CTEs makes i) query logic clearer, ii) permites a view definition to be used in multiple places in query.

vii) Scalar subqueries

Returns only one tuple containing a single attribute.

This can be used wherever an expression returning a value is permitted.

e.g. SELECT, WHERE, HAVING clauses.

Def. A correlated subquery is
a subquery that uses a correlation name from an outer query.
It is legal to use only the names defined in itself or contains itself.
If a name is defined both locally and globally, it follows the scope rule.

Remark It is possible to write the same query in several ways in SQL.
This flexibility allows user to think query in an intuitive sense.
So there is a substantial amount of redundancy in SQL.

3.9 Modification of the Database

i) DELETE FROM
Finds all tuple which given predicates are true, then delete.
Operates only on one relation.

ii) INSERT INTO
Inserts data into relation.
SQL allows attributes to be specified as part of the INSERT statement.

iii) UPDATE SET
Changes a value in tuple without changing all the values.
CASE construct can be used to handle multiple updates in a sigle query.

Remark. Every modification is performed after all the tests are performed
as modification can result in changes of tests.
ex) INSERT INTO r SELECT * FROM r.