SQL: Part (I)

Qiang Yin

Spring, 2024

Quick review

Relational algebra is a query language for relational data.

- Selection $\sigma_p(R)$
- Projection $\Pi_{A_1,\ldots,A_k}(R)$
- Product $R \times S$
- Union $R \cup S$
- Difference $R - S$
- Renaming $\rho_{S(A_1,\ldots,A_k)}(R)$, $\rho_S(R)$

Simple: (1) a small set of core operations; (2) semantics are easy to grasp.

Declarative: each operation only defines what data are needed.

SQL

SQL is the standard query language supported by most DBMS.

- SQL: Structured Query Lanuage
- Pronounced “S-Q-L” or “sequel”

A brief history

- IBM system R, early 1970s
- ANSI/ISO SQL-86 (SQL1)
- ANSI SQL-89
- ANSI/ISO SQL-92 (SQL2)
- ANSI/ISO SQL:1999 (SQL3)
- SQL:2003, SQL:2005, SQL:2008, SQL:2011, SQL:2016, SQL:2019
- SQL:2023, adds data type JSON, add SQL/PGQ for property graph queries

Relational Database language

- Procedural: specify what data are needed and how to get the data.
- Declarative: specify what data are needed without specifying how to get the data.
- DDL (data definition language): Specification notation for defining the database schema.
- DML (data manipulation language): DML is also known as query language.

SQL DDL

SQL DDL

```
CREATE TABLE R(
    ...,
    attribute_name attribute_type,
    ...,
    [integrity_constraints],
    ...
);
DROP TABLE R;
```

Example

```
create table department      -- sql is insensitive to case
  (dept_name varchar(20),    -- sql is insensitive to white spaces
   building varchar(15),     -- everything from '--' to the end of
   budget numeric(12,2),     -- line is ignored
   primary key(dept_name)); -- primary key constraint
drop table department;
```

Built-in data types in SQL

char(n)	fixed-length string with len=n
varchar(n)	variable-length string with max_len=n
int, smallint	integer, small integer
numeric(p,d)	fixed point number
real, double precision	floating point and double-precision floating point numbers
float(n)	floating-point number, with precision at least n digits

Table: Basic data types in SQL

- Machine dependent types: int, smallint, real, double precision.
- Each type has a special value called NULL.
- NULL means that the value is unknown or not applicable.
  - SQL introduce special rules for dealing with NULL's.

Integrity constraints

```
CREATE TABLE instructor (
  ID varchar(5),
  name varchar(20) not null,
  dept_name   varchar(20),
  salary      numeric(8,2),
  primary key (ID),
  foreign key (dept_name) references department);
```

- primary key $(A_1, \ldots, A_n)$: attributes A_1, ..., A_n form the primary key for the relation.
- foreign key $(A_1, \ldots, A_n)$ references S: the values of attributes $(A_1, \ldots, A_k)$ must correspond to values of the primary key of table S.
- not null: the null value is not allowed for the specified attribute.
- Primary keys are not nullable.

Basic database modification

- Insertion: insert a tuple into table R

  ```
  INSERT INTO R(A_1,..,A_n) VALUES (v_1,...,v_n);
  ```

  Example:

  ```
  INSERT INTO instructor VALUES('10211', 'Turing', 'Comp. Sci.', 95000);
  INSERT INTO instructor(ID, name) VALUES('10222', 'Root');
  ```

- Deletion: purge tuples satisfying a given condition from table R

  ```
  DELETE FROM R WHERE condition
  ```

  Example:

  - `DELETE FROM instructor WHERE name='Turing';`
  - `DELETE FROM student;`

- DBMS will prevent any update to the database that would violate an integrity constraint.

Basic SQL Queries

Basic SQL queries

SELECT A_1, A_2, ..., A_n
FROM R_1, R_2, ..., R_m
WHERE P;

A basic sql query can be expressed by a SELECT-FROM-WHERE statement as shown above.

- A_1, A_2, ..., A_n: a list of desired attributes in the query.
- R_1, R_2, ..., R_m: a list of tables accessed during the query evaluation.
- P: a filtering predicate involving the attributes from R_1, R_2, ..., R_m.

Example

List the ID and name of every instructor from the Computer Science department.

- `SELECT ID, name FROM instructor WHERE dept_name = 'Comp. Sci.';`

More examples

- The WHERE clause is optional.

```
SELECT * from instructor; -- * is a shorthand for all attributes
```

- Use logical connectives AND, OR and NOT in the WHERE clause.

```
SELECT ID, name FROM student
WHERE tot_cred > 30 AND (dept_name = 'Physics' OR
                         dept_name = 'Music');
```

- SELECT list can contain expressions

```
SELECT ID, name, salary/12 FROM instructor;
```

- Use a relation name prefix to distinguish attributes with the same name.

```
SELECT student.name, instructor.name
FROM student, advisor, instructor
WHERE student.ID = advisor.S_ID
   AND advisor.i_ID = instructor.ID;
```

Semantics of SFW statements

for each tuple $t_1 \in R_1$ do
    ...
        for each tuple $t_m \in R_m$ do
            if P is true w.r.t. t_1,..., t_m then
                evaluate A_1, ..., A_n according to
                t_1, ..., t_m to produce a tuple in the result

Table: SELECT A_1, A_2, ..., A_n FROM R_1, R_2, ..., R_m WHERE P

Question. Is the above SQL query equivalent to the following relational algebra query?

$$\Pi_{A_1,\ldots,A_n}(\sigma_P(R_1 \times \cdots \times R_m)).$$

Bag semantics vs. set semantics

- SQL adopts bag (i.e., multiset) semantics by default.
  - That is, duplicates are allowed in query results.
- Use keyword DISTINCT to eliminate duplicates explicitly.

dept_name
Finance
History
Comp. Sci.
Physics
History
Comp. Sci.

```
SELECT dept_name from instructor;
```

dept_name
Finance
History
Comp. Sci.
Physics

```
SELECT DISTINCT dept_name
from instructor;
```

String operations

- Strings literals (case sensitive) are quoted by single quotes.

```
SELECT ID, name FROM instructor WHERE dept_name = 'Comp. Sci';
```

- Comparison: $\mathtt{str}_1 < \mathtt{str}_2$ w.r.t. the lexicographic order.
  - Similar for $=, \geqslant, <, \leqslant, <>$.

- Pattern matching: LIKE matches a string against a pattern.
  - The percent (%) character matches any string of zero or more characters.

    ```
    SELECT name FROM instructor WHERE name LIKE '%and%';
    ```

  - The underscore (_) character matches any single character.

    ```
    SELECT ID FROM instructor WHERE name LIKE '___';
    ```

  - Use keyword escape to specify an escape character.

    ```
    SELECT ID FROM instructor WHERE name LIKE 'ab\%cd' ESCAPE '\';
    ```

Renaming

- Keyword AS in the SELECT to rename attributes.

```
SELECT ID, salary/12 AS month_salary FROM instructor;
```

- Keyword AS in the FROM clause to rename relations.

```
SELECT DISTINCT name
FROM instructor, advisor AS S, advisor AS T
WHERE instructor.ID=S.i_ID AND
      S.i_ID = T.i_ID AND S.s_ID <> T.s_ID;
```

- The keyword AS is optional.

```
SELECT ID, salary/12  month_salary FROM instructor;
```

Ordering output

```
SELECT ... FROM ... [WHERE ...]
ORDER BY ..., column[ASC|DESC], ...;
```

- Append a ORDER BY clause at the end of a SFW query to sort the query result.
  - DESC = descending, ASC=ascending.
  - ASC is the default option.
- List all instructors, sort them by salary (descending) and name (ascending).

```
SELECT * FROM instructor
ORDER BY salary DESC, name;
```

Limit output

- A LIMIT $\mathbf{n}$ clause can be append to a query to limit the number of tuples in output.
- We can write top-$\mathbf{n}$ queries by combing an ORDER BY clause and a LIMIT $\mathbf{n}$ clause.

Example

- `SELECT * FROM instructor LIMIT 2;`
- `SELECT name FROM instructor ORDER BY salary DESC LIMIT 1;`
- `SELECT ID FROM STUDENT ORDER BY tot_cred LIMIT 3;`

Set operations

```
SELECT ... FROM ... WHERE ...
UNION|INTERSECT|EXCEPT
SELECT ... FROM ... WHERE ...;
```

- SQL supports UNION, INTERSECT and EXCEPT as in RA.
- They all eliminate duplicates by default.
- To retain all duplicates in query results, explicitly use keyword ALL
  - UNION ALL, INTERSECT ALL, EXCEPT ALL

Examples

- Find the courses taught in Fall 2017 or in Spring 2018.

```
SELECT course_id FROM section
WHERE semester = 'Fall' AND year = 2017
UNION
SELECT course_id FROM section
WHERE semester = 'Spring' AND year = 2018;
```

- Find the courses taught in Fall 2017 but not in Spring 2018.

```
SELECT course_id FROM section
WHERE semester = 'Fall' AND year = 2017
EXCEPT
SELECT course_id FROM section
WHERE semester = 'Spring' AND year = 2018;
```

Basic SQL queries recap

- SELECT-FROM-WHERE statements
- SQL uses bag semantics by default
- Use keyword AS for renaming when needed
- ORDER BY clause: ordering output
- LIMIT clause for top-n queries
- Set operations: UNION, INTERSECT, EXCEPT