SQL: Part (I)

March 3, 2023

Announcements

- Assignment (I) due: March 5.
- Assignment (II) will be released on next Monday.
- Please make sure that you have set up PostgreSQL properly.
  - You will need it for Assignment (II).
  - Yo may want to read the handy tutorial prepared by the TAs.

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 $\rho_{S(A_1,\ldots,A_k)}(R)$, $\rho_S(R)$
- Joins $R \bowtie_\theta S$, $R \bowtie S$

Relational data languages

- 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.

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:2008, SQL:2011, SQL:2016, SQL:2019

Relational Database language

- 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.

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.

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;`

Remark. DBMS will prevent any update to the database that violates 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 t1 ∈ R1 do
    ...
        for each tuple tm ∈ Rm do
            if P is true for t1,..., tm then
                evaluate A1, ..., An according to
                t1, ..., tm 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: $\mathrm{str}_1 < \mathrm{str}_2$ 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 '___';
    ```

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

Aggregation and Grouping

Aggregate functions

AVG	average value
MIN	minimum value
MAX	maximum value
SUM	sum of values
COUNT	number of values

An aggregate function combines a collection of values into a single value.

Basic aggregation

Aggregate functions can only be used in the SELECT output list.

- Find the average salary of instructors in the CS department

```
SELECT AVG(salary)
FROM instructor
WHERE dept_name= 'Comp. Sci.';
```

- Find the number of tuples in the course relation

```
SELECT COUNT(*) FROM course;
```

- Get the number of students in CS and their average credits.

```
SELECT COUNT(*), AVG(tot_cred)
FROM student
WHERE dept_name = 'Comp. Sci.';
```

Distinct aggregation

- Find the total number of instructors who have taught in the Spring 2010 semester.

```
SELECT COUNT(DISTINCT ID)
FROM teaches
WHERE semester = 'Spring' AND year = 2010;
```

- COUNT, SUM and AVG support keyword DISTINCT.

Question. How about MIN and MAX?

Aggregation with grouping

- We can use a clause

  ```
  GROUP BY list_of_columns
  ```

  to apply aggregate functions to a group of sets of tuples.

- Get the average credit of the students for each department.

```
SELECT dept_name, AVG(tot_cred)
FROM student
GROUP BY dept_name;
```

Semantics of GROUP BY

```
SELECT ...
FROM ...
WHERE ...
GROUP BY A1, ..., Ak
```

1. Evaluate the relation $\mathbf{R}$ expressed by the FROM and WHERE clauses.
2. Group the rows of $\mathbf{R}$ according the GROUP BY attributes A_1, ..., A_k.
3. Evaluate the SELECT clause.

Example of GROUP BY

ID	name	dept_name	salary
22222	Einstein	Physics	95000
10101	Srinivasan	Comp. Sci.	65000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000

```
SELECT dept_name, AVG(salary) FROM instructor GROUP BY dept_name;
```

ID	name	dept_name	salary
22222	Einstein	Physics	95000
10101	Srinivasan	Comp. Sci.	65000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000

1. Group rows according to the values of GROUP BY columns

dept_name	avg_salary
Physics	91000
Comp. Sci.	70000

2. Compute aggregation for each group

Restriction on SELECT

If a query uses aggregate/group by, then every attribute in the SELECT clause must

- either enclosed in an aggregate function, or
- in the GROUP BY list.

Example

The following queries are invalid.

- `SELECT dept_name, ID, AVG(salary) FROM instructor GROUP BY dept_name;`
- `SELECT ID, MAX(salary) FROM instructor;`

Remark. This ensures that any SELECT expression produces only one value for each group.

Aggregation: HAVING clause

HAVING filters groups based on the group properties including

- aggregate values
- GROUP BY column values

Example

List the average salary for each department with more than 10 instructors.

```
SELECT dept_name, AVG(salary)
FROM instructor
GROUP BY dept_name
HAVING COUNT(*) > 10;
```

Question. What attributes can be used in the HAVING clause?

Recap

- SQL DDL
- SELECT-FROM-WHERE statement
- Set operations of SQL
- Aggregation and grouping

Next lecture

- Three-valued logic of SQL
- More joins
- Subqueries
- More integrity constraints