COMP3311 24T1 Database Systems

week 3 - 1

Outline

- **Announcements**
- Roadmap & Recap
- Data Modification in SQL
- Single Table Queries
- Views

Announcement 1

- **Quiz - 1**
  - **stay updated:** email; web cms notices; lecture slides
  - ~50 students didn't submit ☹️
  - most students get full marks 🥳
    - the average score is a bit higher than before

- **Quiz - 2**
  - started today, ddl: 23:59:59, Friday (1st Mar 2024)

Announcement 2

- **Assignment - 1**
  - **stay updated:** email; web cms notices; lecture slides
  - submission before the end of week 5
  - will be discussing about it on this Thursday (29th Feb)

Outline

- Announcements
- **Roadmap & Recap**
- Data Modification in SQL
- Single Table Queries
- Views

Roadmap & Recap

In the last week, we have covered:

ER-Rel Mapping, ER-Rel-SQL Mapping

- Entities: strong, weak
- Relationships: n-m, 1-n, 1-1, n-ary relationships
- Attributes: composite, multi-valued
- Subclasses: ER-style, OO style, Single-table style

Design a DB:

- Start from ER
- Convert into Rel/SQL

Roadmap & Recap

In the last week, we have covered:

SQL

- Syntax: constants (string, number …), identifiers, keywords
- Operators: arith, string (matching), NULL value (logic, check)
- SQL DDL:
  - define a schema (keys, constraints indicated by “primary key”?, constraints, checks)
  - checks CANNOT have sub-queries (in postgres), constraints CAN have names
- Manage DB: create/drop DB, create/alter/drop Table …

Basic SQL operations
- sub-languages: DDL, data update language, data query language

Outline

- Announcements
- Roadmap & Recap
- **Data Modification in SQL**
- Single Table Queries
- Views

Data Modification in SQL

We have seen statements to modify table meta-data (in DB catalog):

- CREATE TABLE ... add new, initially empty, table to DB
- DROP TABLE ... remove table data (all tuples) and meta-data
- ALTER TABLE ... change meta-data of table (e.g add constraints)

SQL also provides statements for modifying data in tables:

- INSERT ... add a new tuple(s) into a table
- DELETE ... remove tuples from a table (via condition)
- UPDATE ... modify values in existing tuples (via condition)

Constraint checking is applied automatically on any change.
Operation fails (no change to DB) if any constraint check fails

Insertion

Add new tuples via the INSERT operation:

```
INSERT INTO RelationName
VALUES (val1, val2, val3, ...)

INSERT INTO RelationName(Attr1, Attr2, ...)
VALUES (valForAttr1, valForAttr2, ...)

INSERT INTO RelationName
VALUES Tuple1, Tuple2, Tuple3, ...
```

The first two add a single new tuple into RelationName.

The last form adds multiple tuples into RelationName.

Insertion – cont

INSERT INTO R VALUES (v1,v2,...)

- values must be supplied for all attributes of R
- in **same order** as appear in CREATE TABLE statement
- special value **DEFAULT** forces default value or NULL

INSERT INTO R(A1,A2,...) VALUES (v1,v2,...)

- can specify any subset of attributes of R
- values must match attribute specification order
- unspecified attributes are assigned default or null

E

Insertion - example

```
-- create
CREATE TABLE EMPLOYEE (
 zid varchar(8) CONSTRAINT ZidValidity check (zid ~ '^z[0-9]{7}$') PRIMARY KEY,
 name TEXT NOT NULL,
 role TEXT DEFAULT 'Lecturer',
 intro TEXT
);

CREATE TABLE CONTACT (
 employee varchar(8) REFERENCES EMPLOYEE(zid),
 name TEXT NOT NULL,
 relation TEXT
);

-- insert
-- INSERT INTO EMPLOYEE VALUES ('z1234567', 'Clark', 'Lecturer');
-- INSERT INTO EMPLOYEE VALUES ('z2345678', 'Dave', 'Tutor', 'Dave is a tutor');
-- INSERT INTO EMPLOYEE VALUES ('z2345678', 'Bob', 'Bob is a Lecturer');
-- what if we want to add Bob as a lecturer without the 'role' attribute?
-- what if we want to add Clark and Dave together?
-- what if we want to add a column to the table? Say email?

-- fetch
SELECT * FROM EMPLOYEE;
```

Try it: PostgreSQL - OneCompiler - Write, run and share PostgreSQL code online

Bulk Insertion of Data

Tuples may be inserted individually:

- insert into Stuff(x,y,s) values (2,4,'green');
- insert into Stuff(x,y,s) values (4,8,null);
- insert into Stuff(x,y,s) values (8,null,'red');
- ...

but this is tedious if 1000's of tuples are involved.

It is also inefficient

all relevant constraints are checked on insertion of each tuple

So, most DBMSs provide **non-SQL** methods for bulk insertion

Bulk Insertion of Data - Cont

Bulk insertion methods typically ...

- use a compact representation for each tuple
- "load" all tuples without constraint checking
- do all constraint checks at the end
- if any tuples fail checks, none are inserted

```
COPY table_name [ ( column_name [, ...] ) ]
    FROM { 'filename' | PROGRAM 'command' | STDIN }
    [ [ WITH ] ( option [, ...] ) ]
    [ WHERE condition ]
```

https://www.postgresql.org/docs/16/sql-copy.html

K

Deletion

Removing tuples is accomplished via **DELETE** statement:

```
DELETE FROM Relation
WHERE  Condition
```

Removes all tuples from **Relation** that satisfy **Condition**.

Special case: Make relation R empty.

`DELETE FROM R;` or `DELETE FROM R WHERE true;`

E

Deletion - example

```
-- create
CREATE TABLE EMPLOYEE (
 zid varchar(8) CONSTRAINT ZidValidity check (zid ~ '^z[0-9]{7}$') PRIMARY KEY,
 name TEXT NOT NULL,
 role TEXT DEFAULT 'Lecturer',
 intro TEXT
);

CREATE TABLE CONTACT (
 employee varchar(8) REFERENCES EMPLOYEE(zid),
 name TEXT NOT NULL,
 relation TEXT
);

-- insert
INSERT INTO EMPLOYEE VALUES ('z1234567', 'Clark', 'Lecturer');
INSERT INTO EMPLOYEE VALUES ('z2345678', 'Dave', 'Tutor', 'Dave is a tutor');
INSERT INTO EMPLOYEE VALUES ('z1345678', 'Bob', 'Tutor', 'Bob is a tutor');
-- INSERT INTO CONTACT VALUES ('z1345678', 'Bella', 'Mother');

-- How to delete all tutors?
-- How to delete all people whose zid starts with digit 1?
-- How to delete everything? What else command can we use (other than delete)?

-- fetch
SELECT * FROM EMPLOYEE;
```

Try it: PostgreSQL - OneCompiler - Write, run and share PostgreSQL code online

Deletion – Semantic

Method A for DELETE FROM R WHERE Cond :

```
FOR EACH tuple T in R DO
   IF T satisfies Cond THEN
      remove T from relation R
   END
END
```

Method B for DELETE FROM R WHERE Cond :

```
FOR EACH tuple T in R DO
   IF T satisfies Cond THEN
      make a note of this T
   END
END
FOR EACH noted tuple T DO
   remove T from relation R
END
```

K

Deletion – Semantic – Cont

Does it matter which method the DBMS uses?

For most cases, the same tuples would be deleted

But if Cond involves a query on the table R

the result of Cond might change as the deletion progresses
so Method A might delete less tuples than Method B

E.g.

```
DELETE FROM EMPLOYEE
WHERE (SELECT count(*) FROM EMPLOYEE) > 1;
```

Method A deletes employees until there are only 1 left
Method B deletes all employees if there were more than 1 to start with

Try it: PostgreSQL - OneCompiler - Write, run and share PostgreSQL code online

Postgres uses Method B (take note and then delete)

K

Update

The **UPDATE** statement allows you to

- modify values of specified attributes in specified tuples of a relation
-

```
UPDATE R
SET    List of assignments
WHERE  Condition
```

Each tuple in relation R that satisfies Condition is affected

Assignments may:

- assign constant values to attributes,
  - e.g. SET price = 2.00
- use existing values in the tuple to compute new values,
  - e.g. SET price = price * 0.5

E

Update - example

```
-- create
CREATE TABLE EMPLOYEE (
 zid varchar(8) CONSTRAINT ZidValidity check (zid ~ '^z[0-9]{7}$') PRIMARY KEY,
 name TEXT NOT NULL,
 role TEXT DEFAULT 'Lecturer',
 intro TEXT
);

CREATE TABLE CONTACT (
 employee varchar(8) REFERENCES EMPLOYEE(zid),
 name TEXT NOT NULL,
 relation TEXT
);

-- insert
INSERT INTO EMPLOYEE VALUES ('z1234567', 'Clark', 'Lecturer');
INSERT INTO EMPLOYEE VALUES ('z2345678', 'Dave', 'Tutor', 'Dave is a tutor');
INSERT INTO EMPLOYEE VALUES ('z1234568', 'Bob', 'Admin');

-- How to change the intro for Bob?
-- How to change the role and intro for Bob?
-- How to change the role of all employees whose zid starts with 1 into Lecturer?
-- What will happen if we don't have the WHERE clause?

-- fetch
SELECT * FROM EMPLOYEE;
```

Try it: PostgreSQL - OneCompiler - Write, run and share PostgreSQL code online

Outline

- Announcements
- Roadmap & Recap
- Data Modification in SQL
- **Single Table Queries**
- Views

Queries

A **query** is a declarative program that retrieves data from a database.

declarative = say what we want, not method to get it

Queries are used in two ways in RDBMSs:

- interactively (e.g. in psql)
  - the entire result is displayed in tabular format on the output
- by a program (e.g. in a PLpgSQL function)
  - the result tuples are consumed one-at-a-time by the program

SQL is based on the relational algebra (covered after we learn SQL)

SQL Query Language

An **SQL query** consists of a **sequence of clauses**:

```
SELECT   projectionList
FROM     relations/joins
WHERE    condition
GROUP BY groupingAttributes
HAVING   groupCondition
```

WHERE, GROUP BY, HAVING clauses are optional.

Result of query: a relation, typically displayed as a table.

Result could be just one tuple with one attribute (i.e. one value) or even empty

SQL Query Language - Cont

Functionality provided by SQL ...

Filtering: extract attributes from tuples, extract tuples from tables

```
SELECT b,c FROM R(a,b,c,d) WHERE a > 5
```

Combining: merging related tuples from different tables

```
... FROM R(x,y,z) JOIN S(a,b,c) ON R.y = S.a
```

Summarising: aggregating values in a single column

```
SELECT avg(mark) FROM ...
```

Set operations: union, intersection, difference

```
SELECT b,c FROM R(a,b,c,d) WHERE a > 5 UNION SELECT b,c FROM R(a,b,c,d) WHERE d < 4
```

SQL Query Language - Cont

More functionality provided by SQL ...

Grouping: forming subsets of tuples sharing some property

```
... GROUP BY R.a
```

(forms groups of tuples from R sharing the same value of a)

Group Filtering: selecting only groups satisfying a condition

```
... GROUP BY R.a HAVING max(R.a) < 75
```

Renaming: assign a name to a component of a query

```
SELECT a as name
FROM Employee(a,b,c) e WHERE e.b > 50000
```

E

Grouping - example

```
-- Creating a table called 'sales' with columns 'product', 'price' and 'quantity'
CREATE TABLE sales (
 product VARCHAR(20),
 price DECIMAL(10,2),
 quantity INT
);

-- Inserting some sample data into the table
INSERT INTO sales VALUES
('Laptop', 1000.00, 5),
('Mouse', 20.00, 10),
('Keyboard', 30.00, 8),
('Laptop', 900.00, 3),
('Mouse', 25.00, 12);

-- Using GROUP BY clause to aggregate the data by product and calculate the total sales
SELECT * FROM sales;

-- Can we do this?
-- SELECT * FROM sales GROUP BY product;

-- What are the restrictions?
-- SELECT product, SUM(price * quantity) AS total_sales
-- FROM sales
-- GROUP BY product;
```

Try it: [PostgreSQL - OneCompiler - Write, run and share PostgreSQL code online]

E

Query - example

```
-- Create Students table
CREATE TABLE Students (
  StudentID INT PRIMARY KEY,
  FirstName VARCHAR(50),
  LastName VARCHAR(50),
  Email VARCHAR(100),
  Year INT
);


-- Insert sample data into Students table
INSERT INTO Students (StudentID, FirstName, LastName, Email, Year) VALUES
(1, 'John', 'Doe', 'john.doe@example.com', 2),
(2, 'Jane', 'Smith', 'jane.smith@example.com', 1),
(3, 'Mike', 'Brown', 'mike.brown@example.com', 3),
(4, 'Emily', 'Wilson', 'emily.wilson@example.com', 1),
(5, 'David', 'Johnson', 'david.johnson@example.com', 1),
(6, 'Anna', 'Martinez', 'anna.martinez@example.com', 2),
(7, 'Carlos', 'Garcia', 'carlos.garcia@example.com', 1),
(8, 'Sara', 'Lee', 'sara.lee@example.com', 2),
(9, 'Alex', 'Wilson', 'alex.wilson@example.com', 3),
(10, 'Nina', 'Patel', 'nina.patel@example.com', 4),
(11, 'Omar', 'Khan', 'omar.khan@example.com', 1),
(12, 'Tara', 'Sharma', 'tara.sharma@example.com', 2),
(13, 'Liam', 'Wilson', 'liam.wilson@example.com', 2),
(14, 'Ava', 'Patel', 'ava.patel@example.com', 1),
(15, 'Ethan', 'Sharma', 'ethan.sharma@example.com', 3),
(16, 'Mia', 'Wilson', 'mia.wilson@example.com', 4),
(17, 'Aryan', 'Patel', 'aryan.patel@example.com', 2),
(18, 'Isha', 'Sharma', 'isha.sharma@example.com', 1);


-- questions
-- List all students whose ID is larger than 5
-- List all students whose FirstName starts with 'A'
-- Add one year for all students/Reduce one year from all students
-- What's the average year of all students?
-- How many students do we have for each year?
-- What's the average year for the students for each unique family name (with more than one students having that family name)?
```

Try it: PostgreSQL - OneCompiler - Write, run and share PostgreSQL code online

Thank you!