COSC210

Database Mangement Sytems

une University of New England

Lecture 4 - SQL Lecture One - Data Definitions

Dr. Edmund Sadgrove

Summary

- PostgreSQL
- Banking ER Diagram
- Mapping to PostgreSQL
- Simple Queries
- Updating Constrains
- Updating Data

* Droping Tables, Rows and Columns

PostgreSQL

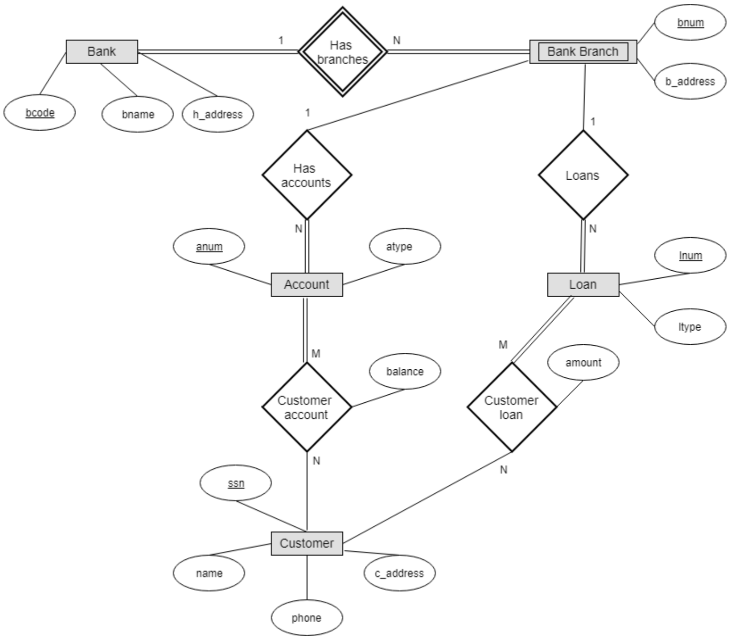
- About PostgreSQL:
  - Open Source.
  - Object-Relational Database System (ORDBMS).
  - Extensive documentation.
  - Active community base.
- More information at postgresql.org

ERD to Relational Mapping

- Mapping Entity Relationship Diagrams (ERD) requires:
  - Mapping of strong entities.
  - Mapping of weak entities.
  - Mapping of cardinalities.
  - Placing constraints on keys.

* More on mapping later.

Banking ER Diagram



Step One - Strong Entities

- Bank
    - bcode (primary key)
    - bname
    - h_address

```
CREATE TABLE bank (
    bcode int,
    bname varchar(20) NOT NULL,
    h_address varchar(50) NOT NULL,
    PRIMARY KEY(bcode)
);
```

Step One - Strong Entities

- Account
    - anum (primary key)
    - atype
    - bno & bco (composite foreign key)

```
CREATE TABLE account (
    anum int,
    bco int,
    bno int,
    atype varchar(20),
    PRIMARY KEY(anum),
    FOREIGN KEY(bno,bco) REFERENCES bank_branch(bnum,bco)
        ON DELETE SET NULL ON UPDATE CASCADE
);
```

Step One - Strong Entities

- Loan
    - lnum (primary key)
    - ltype
    - bno & bco (composite foreign key)

```
CREATE TABLE loan (
    lnum int,
    ltype varchar(20),
    bco int,
    bno int,
    PRIMARY KEY(lnum),
    FOREIGN KEY(bno,bco) REFERENCES bank_branch(bnum,bco)
        ON DELETE SET NULL ON UPDATE CASCADE
);
```

Step One - Strong Entities

- Customer
    - ssn (primary key)
    - name
    - phone
    - c_address

```
CREATE TABLE customer (
    ssn int,
    name varchar(20),
    phone varchar(20),
    c_address varchar(20),
    PRIMARY KEY(ssn)
);
```

Step Two - Weak Entities

- Bank Branch
    - composite primary key (bnum,bco)
    - b_address
    - bco (foreign key)

```
CREATE TABLE bank_branch(
    bnum int,
    bco int,
    b_address varchar(20),
    PRIMARY KEY(bnum,bco),
    FOREIGN KEY(bco) REFERENCES bank(bcode)
        ON DELETE CASCADE ON UPDATE CASCADE
);
```

* Skip step three (no 1:1)

Step Four - 1:N relationships

- This involves adding foreign keys to the N side of the relationship

- In bank_branch:

```
FOREIGN KEY(bco) REFERENCES bank(bcode)
        ON DELETE CASCADE ON UPDATE CASCADE
```

- In Account

```
FOREIGN KEY(bno,bco) REFERENCES bank_branch(bnum,bco)
        ON DELETE SET NULL ON UPDATE CASCADE
```

- In Loan

```
FOREIGN KEY(bno,bco) REFERENCES bank_branch(bnum,bco)
        ON DELETE SET NULL ON UPDATE CASCADE
```

Step Five - M:N Relationships

- Customer Loan
  - composite primary key (lno,cssn)
  - amount
  - lno (foreign key)
  - cssn (foreign key)

```
CREATE TABLE customer_loan (
    lno int,
    cssn int,
    amount int,
    PRIMARY KEY(lno,cssn),
    FOREIGN KEY(lno) REFERENCES loan(lnum)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY(cssn) REFERENCES customer(ssn)
        ON DELETE CASCADE ON UPDATE CASCADE
);
```

Step Five - M:N Relationships

- Customer Account
  - composite primary key (accnum,ano,cssn)
  - balance
  - ano (foreign key)
  - cssn (foreign key)

```
CREATE TABLE customer_account (
    balance int,
    ano int,
    cssn int,
    PRIMARY KEY(ano,cssn),
    FOREIGN KEY(ano) REFERENCES account(anum)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY(cssn) REFERENCES customer(ssn)
        ON DELETE CASCADE ON UPDATE CASCADE
);
```

Inserting Data into our Tables

- Inserting data

```
--Insert into specific columns
INSERT INTO table_name
    column_name1,column_name2...
    VALUES ( value1,value2... )

--Insert into all columns
INSERT INTO table_name
    VALUES ( value1,value2... )

--Insert from data file
\copy table_name FROM '/path/to/my/data/file.csv' CSV;
```

Inserting Data into our Tables

- Single row and multiple row.

```
--Bank:
INSERT INTO bank (bcode,bname,h_address)
    VALUES (555, 'My Bank', 'Sydney');

--Bank Branch:
INSERT INTO bank_branch (bnum,bco,b_address)
    VALUES
        (1001,555,'Armidale'),
        (1002,555,'Guyra'),
        (1003,555,'Sydney'),
        (1004,555,'Melbourne');
```

Inserting Data into our Tables

```
--Account:
INSERT INTO account (anum,bco,bno,atype)
    VALUES
        (1,555,1001,'Saver'),
        (2,555,1001,'Spender'),
        (3,555,1001,'Student'),
        (4,555,1001,'Investor'),
        (5,555,1002,'Saver'),
        (6,555,1002,'Spender'),
        (7,555,1002,'Student'),
        (8,555,1002,'Investor'),
        (9,555,1003,'Saver'),
        (10,555,1003,'Spender'),
        (11,555,1003,'Student'),
        (12,555,1003,'Investor'),
        (13,555,1004,'Saver'),
        (14,555,1004,'Spender'),
        (15,555,1004,'Student'),
        (16,555,1004,'Investor');
```

Inserting Data into our Tables

```
--Loan:
INSERT INTO loan (lnum, bco, bno,ltype)
    VALUES
        (111,555,1001,'House'),
        (112,555,1001,'Car'),
        (113,555,1001,'General'),
        (114,555,1002,'House'),
        (115,555,1002,'Car'),
        (116,555,1002,'General'),
        (117,555,1003,'House'),
        (118,555,1003,'Car'),
        (119,555,1003,'General'),
        (120,555,1004,'House'),
        (121,555,1004,'Car'),
        (122,555,1004,'General');
```

Inserting Data into our Tables

- Using random name and phone number generators

```
--Customer:
INSERT INTO customer (ssn,name,phone,c_address)
    VALUES
        (501,'Sonya Carter','(257) 231-1533','Melbourne, VIC 2222'),
        (502,'Randolph Oliver','(476) 738-4677','Armidale, NSW 2350'),
        (503,'Colleen Hall','(185) 336-3857','Armidale, NSW 2350'),
        (504,'Wilson Peterson','(543) 667-3820','Armidale, NSW 2350'),
        (505,'Bessie Snyder','(683) 431-7713',' Sydney, NSW 9999'),
        (506,'Erik Drake','(771) 495-9114','Armidale, NSW 2350'),
        (507,'Terry Little','(594) 141-5933','Hobart, TAS 1568'),
        (508,'Mindy Hines','(522) 563-9580','Armidale, NSW 2350'),
        (509,'Samantha Bass','(483) 131-6173','Melbourne, VIC 2222'),
        (510,'Wilbert Harrington','(667) 217-7407','Guyra, NSW 2352');
```

Inserting Data into our Tables

```
--customer_account:
INSERT INTO customer_account (balance,ano,cssn)
    VALUES
        (3457,14,501),
        (844,3,502),
        (2184,2,503),
        (4868,1,504),
        (21462,9,505),
        (171139,5,506),
        (6542,16,508),
        (284,3,508),
        (3996,6,510),
        (5,3,506),
        (24228,13,509),
        (6954537,12,507);

--customer_loan:
INSERT INTO customer_loan (lno, cssn, amount)
    VALUES
        (111,506,250000),
        (115,510,10000),
        (122,501,500),
        (117,505,1000000);
```

Testing our Data

- The SQL query structure

```
--Simple query
SELECT column1, column2, ...
```

```
FROM table_name;

--With conditions
SELECT column1, column2, ...
FROM table_name WHERE condition1 AND condition2 OR condition3 ...;

--With more options
SELECT column1, column2, ...
FROM table_name WHERE condition1 UNION
    SELECT... GROUP BY... ORDER BY... ;
```

* We can have SQL Subqueries and even Loops

Testing our Data

- Database checks
- With some simple queries

```
--Check customer table
SELECT * FROM Customer;

        --List student accounts
SELECT * FROM account;

--List customers from armidale
SELECT * FROM Customer WHERE customer.c_address='Armidale, NSW 2350';
```

Updating Data

- We can update values in columns based on conditions.

```
--Syntax
UPDATE table_name
    SET column1 = value1, column2 = value2 ,...
        WHERE condition;

--Update customer account balance
UPDATE customer_account
    SET balance = 50
        WHERE cssn = 503;
```

Updating Data

- When storing data we should store money in cents.
- We previously stored it in dollars.
- We should fix this.

```
--Updating customer_account
UPDATE customer_account
    SET balance = balance*100
        WHERE true;

--Syntax
UPDATE customer_loan
    SET amount=amount*100
        WHERE true;
```

Removing Data

- Deleting rows, dropping columns and tables.

```
--Syntax
DELETE FROM table_name
    WHERE condition;

--Dropping columns
ALTER TABLE table_name
    DROP COLUMN column_name;


--Dropping tables
DROP TABLE table_name;
```

Removing Data

- Deleting Rows, Dropping Columns and Tables.

```
--New table
CREATE TABLE transaction (
    tid int,
    cssn int,
    ption varchar(20),
    PRIMARY KEY(tid),
    FOREIGN KEY(cssn) REFERENCES customer(ssn)
        ON DELECT CASCADE ON UPDATE CASCADE
);
--Syntax
DELETE FROM transaction
    WHERE cssn=501;

--Dropping columns
ALTER TABLE transaction
    DROP COLUMN description;


--Dropping tables
DROP TABLE transaction;
```

What's Next?

- More on query structure

* More advance queries

Questions?