



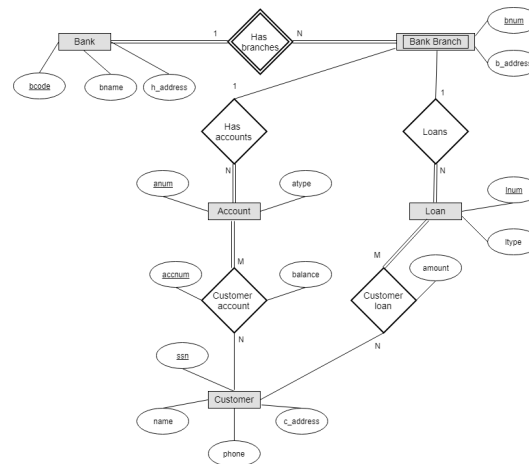
# COSC210 Database Management Systems

## Lecture 10 - SQL Three (Nested Queries)

Dr. Edmund Sadgrove

## Summary

- Revision Example.
- Unions and Joins
- Nested Queries
- Universal Quantifiers
- What to use?
- Rob the Bank



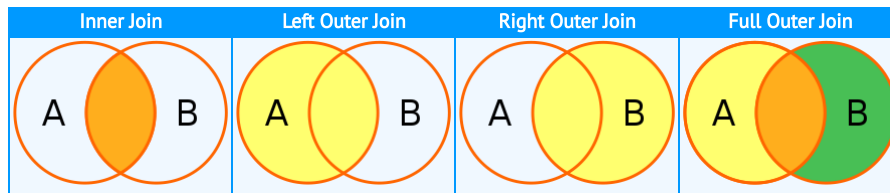
## Revision Example - Views

- A view of cccount details that lists:
  - Customer name - sort ascending.
  - BSB & ACC.
  - Branch loaction.
  - Balance.
  - Account type.

```
CREATE VIEW account_details AS
SELECT c.cname, a.bco||' '||a.bno AS bsb,
       ca.accnum AS acc,bb.b_address, ca.balance, a.atype
FROM customer AS c, bank_branch AS bb,
     customer_account AS ca, account AS a
WHERE c.ssn=ca.csn
      AND ca.ano=a.anum
      AND a.bno=bb.bnum
ORDER BY c.cname,a.atype ASC,bsb,acc,ca.balance,bb.b_address;
```

## Outer Joins

- From last lecture: INNER JOINS and WHERE clauses are often interchangeable.
- OUTER JOINS will not produce the same results.
  - RIGHT OUTER JOIN
  - LEFT OUTER JOIN
  - FULL OUTER JOIN



## Example from Last Lecture

- Display customers with a student account in Armidale branch.
- Change to:
  - Display all customers as well (LEFT OUTER).
  - Display all branches as well (RIGHT OUTER).
  - Display all customers and branches (FULL OUTER).

```
-- Displaying all customers and branches.  
SELECT cname, b_address FROM customer AS c  
      INNER JOIN customer_account AS ca ON c.ssn = ca.cssn  
      INNER JOIN account AS a ON ca.ano = a.anum  
      FULL OUTER JOIN bank_branch AS b ON a.bno = b.bnum  
      AND b.b_address LIKE '%Armidale%' AND a.atype = 'Student';
```

## Nested Queries or Subqueries

- Subqueries allow us to return results from two or more queries.
- They form an additional condition on the main query and restrict results.
- Subqueries can also have subqueries.
- Typical OPERATOR:
  - IN, NOT IN (slower)
  - EXISTS, NOT EXISTS (faster)

```
-- Subquery syntax:  
SELECT column_name [, column_name ]  
FROM   table1 [, table2 ]  
WHERE  column_name OPERATOR  
       (SELECT column_name [, column_name ]  
        FROM table1 [, table2 ]  
        [WHERE])
```

- IN and EXISTS do the same thing, but have different syntax.

## Nested Queries - Examples

- Return cssns for customers who have a loan and a bank account.
  - This can also be expressed as a JOIN.
- Change to customers who have a bank account, but no loan.
  - Easier to use a nested query (NOT IN).

```
-- Customers with accounts, but not loans.  
SELECT DISTINCT cssn FROM customer_account  
WHERE cssn NOT IN  
  (SELECT cssn FROM customer_loan);
```

## Nested Queries - Examples

- Display customers with a student account NOT in Armidale branch.
- This uses a subquery instead of a join.

```
SELECT cname, b_address FROM customer AS c, customer_account AS ca, account AS a, bank_branch as b
WHERE c.ssn = ca.ssn AND ca.ano = a.anum AND a.bno = b.bnum AND
      b.bnum NOT IN
      (SELECT b2.bnum FROM bank_branch AS b2
       WHERE b.b_address LIKE '%Armidale%');
```

## Example Union

- Unions can be used like joins, but require a matching number of columns.
- Lets use a UNION to display customer accounts and loans in one table.
- Will Include Tables:
  - customer
  - customer\_account
  - customer\_loan

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

## Universal and Existential Quantifiers

- SQL can express two types of quantifiers.
  - **Existential quantifiers ( $\exists$ ):**
    - Conditions include:
      - "For some", "there exists", "there is a" or "for at least one".
    - Formally:  $\exists b \in B, R(b)$ ;
      - There exists  $b$  element of  $B$  with condition ( $R$ ).
  - **Universal quantifiers ( $\forall$ ):**
    - Conditions include:
      - "For all", "given any", "for each" or "for every".
    - Formally:  $\forall b \in B, R(b)$ ;
      - For all there exists  $b$  element of  $B$  with condition ( $R$ ).
- Existential requires WHERE clauses, while universal requires nested queries.

## Universal Quantifier #1

- Get Customers whos loan types are all general.
- Will Include Tables:
  - customer
  - customer\_loan
  - loan

```
-- This requires a Universal Quantifier
SELECT cname FROM customer c1, customer_loan cl1, loan l1
WHERE c1.ssn = cl1.cssn AND cl1.lno = l1.lnum
AND c1.ssn NOT IN
  (SELECT c2.ssn FROM customer c2, customer_loan cl2, loan l2
   WHERE c2.ssn = cl2.cssn AND cl2.lno = l2.lnum
   AND l2.lnum NOT IN
    (SELECT l3.lnum FROM loan AS l3
     WHERE l3.ltype = 'General'));
```

## Universal Quantifier #2

- Return a sole customer with general loan types and if another customer with loan type general exists return no result.
- Will Include Tables:
  - customer
  - customer\_loan
  - loan

```
-- This requires a Universal Quantifier
SELECT DISTINCT c1.cname FROM customer c1
WHERE NOT EXISTS
  (SELECT c2.cssn FROM customer_loan c2, loan l2
   WHERE c2.lno = l2.lnum AND l2.ltype = 'General'
   AND c2.cssn NOT IN
    (SELECT c3.cssn FROM customer_loan c3
     WHERE c3.cssn = c1.ssn));
```

## What to use?

- 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 cname,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 cname FROM customer WHERE cname='Randolph Oliver'  
                AND customer_account.cssn=customer.ssn);
```

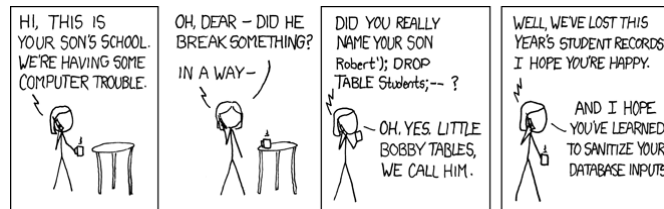
```
--These lines minus from accounts and add to randolf  
UPDATE customer_account SET balance=balance-10  
  WHERE balance>100;  
UPDATE customer_account SET balance=balance+  
  (SELECT COUNT(balance) FROM customer_account WHERE balance>100)*10
```

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

<< WHERE EXISTS (SELECT cname FROM customer WHERE cname='Randolph Oliver' AND  
customer\_account.cssn=customer.ssn);

See: [banking.zip](#)

## Little Bobby Tables



Source: <https://xkcd.com>

# Questions?