COSC210
Database Mangement Sytems

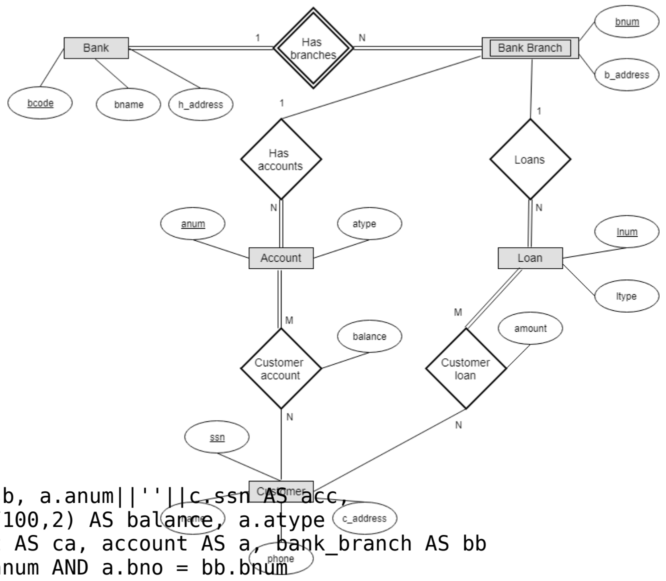
Lecture 8 - SQL Three (Nested Queries)

Dr. Edmund Sadgrove

Summary

* Revision Example. * Unions and Joins

* Nested Queries



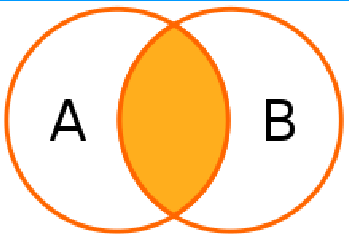
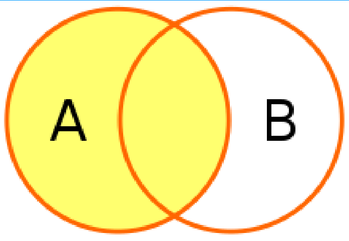
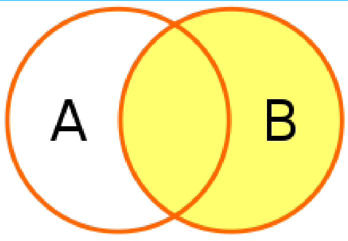
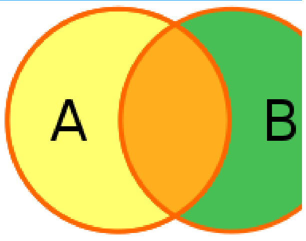
Revision Example - Views

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

```
CREATE VIEW account_details AS
    SELECT c.name, bb.bnum||''||bb.bco AS bsb, a.anum||''||c.ssn AS acc,
    bb.b_address, '$'||''||ROUND(ca.balance/100,2) AS balance, a.atype
        FROM customer AS c, customer_account AS ca, account AS a, bank_branch AS bb
        WHERE c.ssn = ca.cssn AND ca.ano=a.anum AND a.bno = bb.bnum
        ORDER BY c.name ASC, a.atype,bsb,acc,ca.balance;
```

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

Inner Join	Left Outer Join	Right Outer Join	Full Outer Join
			

Example from Last Lecture

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

```
-- Displaying all customers and accounts.
SELECT c.name, a.atype
        FROM customer AS c
```

```
                INNER JOIN customer_account AS ca ON c.ssn = ca.cssn
                FULL OUTER JOIN account AS a ON ca.ano = a.anum AND a.atype ='Student';
```

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 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;
```

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) - multiple OR
    - EXISTS, NOT EXISTS (faster) - single OR

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

* IN checks all results, EXISTS returns true with one match.

Nested Queries - Examples (IN)

- 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 accuount, 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 (IN)

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

```
SELECT c.name FROM customer AS c
        WHERE c.ssn NOT IN
        (SELECT ca.cssn FROM bank_branch AS bb, account AS a, customer_account AS ca
            WHERE ca.ano=a.anum AND a.bno=bb.bnum AND b_address = 'Armidale');
```

- The following will not do the same thing - think about why.

```
SELECT c.name FROM customer AS c, customer_account AS ca, account AS a, bank_branch AS bb
        WHERE c.ssn = ca.cssn AND ca.ano=a.anum
        AND a.bno=bb.bnum AND bb.b_address != 'Armidale';
```

Nested Queries - Examples (EXISTS)

- Customers who have a bank accuount, but no loan.
    - Lets try this with the EXISTS clause.

```
-- Customers with accounts, but not loans.
SELECT DISTINCT c1.cssn FROM customer_account as c1
WHERE NOT EXISTS
    (SELECT * FROM customer_loan c2
     WHERE c2.cssn=c1.cssn);
```

Nested Queries - Examples (EXISTS)

- Display customers with all accounts NOT in Armidale branch.
    - Lets try this with the EXISTS clause.

```
SELECT c.name FROM customer AS c
        WHERE NOT EXISTS
        (SELECT * FROM bank_branch AS bb, account AS a, customer_account AS ca
            WHERE ca.ano=a.anum AND a.bno=bb.bnum AND b_address = 'Armidale' AND c.ssn = ca.cssn);
```

Nested Queries - An Interesting Example

- Return True or False:
    - Determine if more than one House loan exists for two different customers.
    - Without using aggregation.

```
SELECT EXISTS
        (SELECT * FROM customer_loan AS cl1, loan AS l1
         WHERE l1.lnum = cl1.lno AND l1.ltype='House'
         AND EXISTS
            (SELECT * FROM customer_loan AS cl2, loan AS l2
             WHERE l2.lnum = cl2.lno AND l2.ltype='House'
             AND cl1.cssn != cl2.cssn));
```

* This reveals an interesting operational problem, can a customer have more than 1 house loan in our model?

Questions?