

Lecture 8 - SQL Lecture Two - Query Structure

Dr. Edmund Sadgrove

Summary

* Query Structure * Multiple Table Queries * Aggregation

* Creating Views

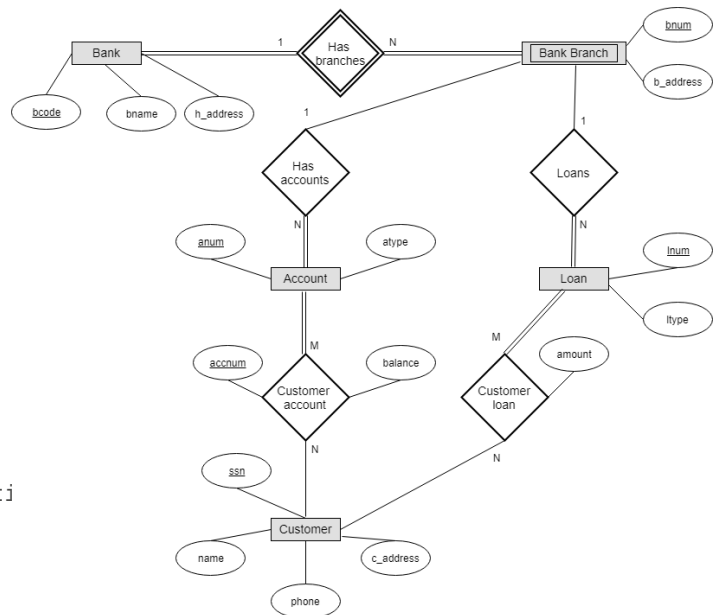
The SQL Query Structure

- Query Structure

```
--Simple query
SELECT column1, column2, ...
FROM table_name;

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

--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 (not covered in this unit).

The 'WHERE' Clause.

- The 'WHERE' clause has all the comparison and logical operators of a regular programming language.

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

One Relation Conditional Queries

- We can use comparison and logical operators to return tuples based on a defined set of conditions.
- Examples:

```
-- Display customer account numbers with balances greater than $500.00
SELECT accnum FROM customer_account WHERE balance > 50000;

-- Display customer loans with amounts less than $1000.00
SELECT * FROM customer_loan WHERE amount < 1000000;

-- Display customer names with ssn's between 502 and 504
SELECT cname, ssn FROM customer WHERE ssn >= 502 AND ssn <= 504;
```

One Relation Conditional Queries

- The LIKE clause can return search results when attribute syntax is unknown.
- String wildcards:
 - % : denotes start and end of string.
 - || : concatenates results.
- Remember to use single quotes (""):

```
-- Display customers that are from Armidale.
SELECT * FROM customer WHERE c_address LIKE '%Armidale%';

-- Display names and phone numbers for customers with an area code starting with 5.
SELECT cname,phone FROM customer WHERE phone LIKE '(5%';

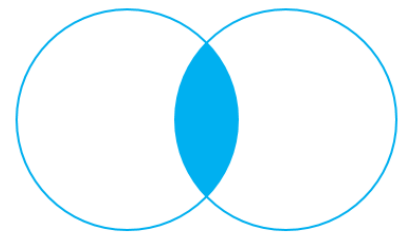
-- Display bank and branch numbers for Student accounts in one cell, preceded by *BSB:*.
SELECT 'BSB:'||bco||bno AS bsb FROM account WHERE atype LIKE '%Student%';
```

Multiple Table Results Using Inner Joins

- An INNER JOIN can be used to join results from multiple tables.
- OUTER JOINS will not produce the same results (more on this later).
 - RIGHT OUTER JOIN
 - LEFT OUTER JOIN
 - FULL OUTER JOIN
- JOIN syntax (inner):

```
--Query structure
SELECT column_1, column_2,..., column_n
FROM table_1 INNER JOIN table_2 ON table1.column_1 = table_2.column

--Compact syntax
SELECT t_1.column_1, t_2.column_2,..., t_n.column_n
FROM table1 AS t_1 JOIN table2 as t_2 ON t_1.column_1 = t_2.column
```



INNER JOIN

Multiple Table Results Using Inner Joins

- Display customers with a student account in Armidale branch.
- Will include tables:
 - customer
 - customer_account
 - account
 - bank_branch

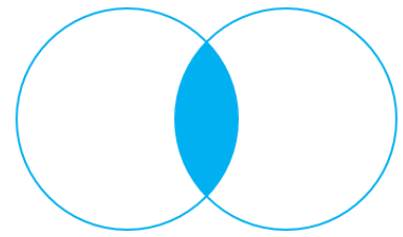
```
SELECT c.cname, a.atype, b.b_address FROM customer AS c
JOIN customer_account AS ca ON c.ssn = ca.cssn
JOIN account AS a ON ca.ano = a.anum
JOIN bank_branch AS b ON a.bno = b.bnum
WHERE b.b_address LIKE '%Armidale%'
AND a.atype LIKE '%Student%';
```

Multiple Table Results Using Inner Joins

- The WHERE clause can be used as an alternative to INNER JOINS.
- This produces the same results, but is more compact.
- But this may be harder to read.
- WHERE syntax (join):

```
--Query structure
SELECT column_1, column_2,..., column_n
FROM table_1, table_2 WHERE table1.column_1 = table_2.column_2;

--Compact syntax
SELECT t_1.column_1, t_2.column_2,..., t_n.column_n
FROM table1 AS t_1, table2 AS t_2 WHERE t_1.column_1 = t_2.column_2;
```



INNER JOIN

Multitable Table Results Using the Where Clause

- Display customers with a student account in Armidale branch.
- Will include tables:
 - customer
 - customer_account
 - account
 - bank_branch

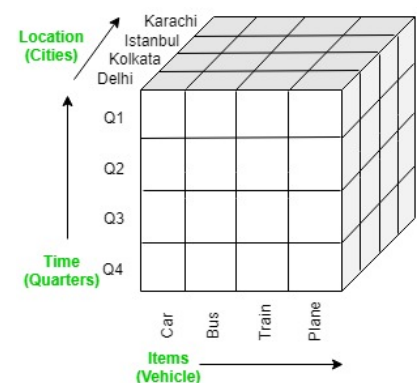
```
SELECT name,atype,b_address FROM customer,customer_account, account,bank_branch
WHERE customer.ssn=customer_account.cssn
AND customer_account.ano=account.anum
AND account.bno=bank_branch.bnum
AND account.atype='Student'
AND bank_branch.b_address LIKE '%Armidale%';
```

*** INNER JOINS and WHERE clauses are often interchangeable.**

Aggregate Functions

- Aggregate functions include:
 - MIN()
 - MAX()
 - AVG()
 - COUNT()
 - SUM()
- A *GROUP BY* is required to pivot around an attribute.
- The HAVING clause is used instead of WHERE to compare results.

```
SELECT t1.column_1, AVG(t2.column_2) AS Average,SUM(t2.column_3) AS total, COUNT(t2.column_4)
FROM table_1 AS t1, table_2 AS t2
WHERE t1.column_5 = t2.column_6 GROUP BY t1.column_1 HAVING AVG(column2) > 100;
```



Aggregate Functions

- Lets display average, total and 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;
```

Aggregate Functions

- List the number of accounts and names of customers who have more than one bank account.
- Will include tables:
 - customer
 - customer_account

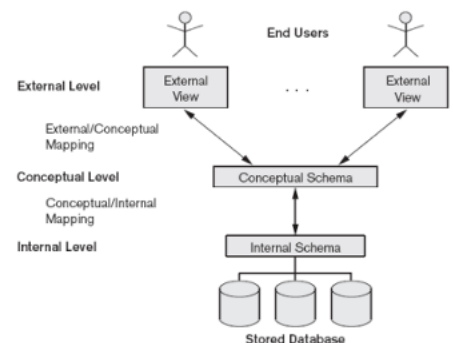
```
SELECT cname, COUNT(ca.ano)
FROM customer AS c, customer_account AS ca
WHERE c.ssn=ca.cssn
GROUP BY ssn
HAVING COUNT(ca.ano) > 1;
```

Creating Views

- PostgreSQL can be used to create different user views.
- Accessing views is similar to accessing tables.
- The \dv command can be used instead of \dt.

```
--Syntax
CREATE VIEW view_name AS
SELECT column1,column2... FROM table_name.
```

Figure 2.2
The three-schema
architecture.



Creating Views

- A view of customer_details that lists:
 - Customer name
 - Account balance
 - Customer address

```
--- '$'||'||'||' converts our balance to a string and adds a dollar sign.
CREATE VIEW customer_details AS
SELECT cname,'$'||'||'||balance as Balance,c_address
FROM customer,customer_account
WHERE customer.ssn=customer_account.cssn;
```

Creating Views

- A view of customer_loans that lists:
 - Customer name

- Loan amount
- Loan type

```
CREATE VIEW loan_details AS
  SELECT name, amount, ltype as loan_type
  FROM customer, loan, customer_loan
  WHERE customer.ssn=customer_loan.cssn
        AND customer_loan.lno=loan.lnum;
```

Questions?