Introduction
Data Retrieval
Subqueries
Joins
Union Operation
Update
Insert
Delete
Update
Conclusion
Suggested Readings

Data Manipulation Language

Natalia Criado
Email: natalia.criado@kcl.ac.uk

Session Objectives

Data Manipulation Language

Introduction

Data Retrieval
- Subqueries
- Joins
- Union Operation

Update
- Insert
- Delete
- Update

Conclusion
- Suggested Readings

In this session, you will learn:

- More advanced queries
- Data update queries

Review

Data Manipulation Language

Introduction

Data Retrieval
Subqueries
Joins
Union Operation

Update
Insert
Delete
Update

Conclusion
Suggested Readings

Which of the following is not true about SQL statements?

A SQL statements are not case sensitive.

B SQL statements can be written on one or more lines.

C Keywords cannot be split across lines.

D Clauses must be written on separate lines.

Review

Consider the following schema

```
STUDENTS(student_code, first_name, last_name, email,
         phone_no, date_of_birth, honours_subject);
```

Which of the following query would display all the students where the second letter in the first name is 'i'?

A select first_name from students where first_name like '_i%';

B select first_name from students where first_name like '%i_';

C select first_name from students where first_name like '%i%';

D select first_name from students where first_name like '_i_';

Data Manipulation Language

Introduction

Data Retrieval
Subqueries
Joins
Union Operation

Update
Insert
Delete
Update

Conclusion
Suggested Readings

Data Manipulation Language

DML allows to retrieve and update data:

- `SELECT` statement retrieves data
- `INSERT`, `UPDATE`, `DELETE` statements update data

Data Manipulation Language

Introduction

Data Retrieval
- Subqueries
- Joins
- Union Operation

Update
- Insert
- Delete
- Update

Conclusion
- Suggested Readings

Subqueries

- Some SQL statements can have a `SELECT` embedded within them
- A subselect can be used in `WHERE` and `HAVING` clauses of an outer `SELECT`, where it is called a subquery or nested query:
  - Subquery produce a temporary table with results that can be accessed by the outer statement
  - Subqueries can be used following a relational operator $(=, <, >, <=, >=, <>)$ in `WHERE` and `HAVING` clauses
  - Subqueries are always enclosed by parentheses
- Subselects may also appear in `INSERT`, `UPDATE`, and `DELETE` statements

Data Manipulation Language

Introduction

Data Retrieval

Subqueries

Joins

Union Operation

Update

Insert

Delete

Update

Conclusion

Suggested Readings

Subqueries: Example

Branch (branchNo, street, city, postcode)
Staff (staffNo, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

List staff who work in branch at '163 Main St'.

```
SELECT staffNo, fName, lName, position
FROM Staff
WHERE branchNo =
      (SELECT branchNo
       FROM Branch
       WHERE street = '163 Main St');
```

Subqueries: Example

```
SELECT staffNo, fName, lName, position
FROM Staff
WHERE branchNo =
      (SELECT branchNo
       FROM Branch
       WHERE street = '163 Main St');
```

- Inner SELECT finds branch number for branch at '163 Main St' ('B003').
- Outer SELECT then retrieves details of all staff who work at this branch.

staffNo	fName	lName	position
SG37	Ann	Beech	Assistant
SG14	David	Ford	Supervisor
SG5	Susan	Brand	Manager

Subquery Rules

- `ORDER BY` clause may not be used in a subquery (although it may be used in outermost `SELECT`)
- Subquery `SELECT` list must consist of a single column name or expression, except for subqueries that use `EXISTS`
- By default, column names refer to table name in `FROM` clause of subquery. Can refer to a table in `FROM` using an alias
- When subquery is an operand in a comparison, subquery must appear on right-hand side
- A subquery may not be used as an operand in an expression

Data Manipulation Language

Introduction

Data Retrieval
Subqueries
Joins
Union Operation

Update
Insert
Delete
Update

Conclusion
Suggested Readings

Table Alias

- Alias can be used to qualify column names when there is ambiguity
- To perform join, include more than one table in FROM clause
- Use comma as separator and typically include WHERE clause to specify join column(s)
- To use an alias for a table in FROM clause:

  `FROM` *tableName* `[AS]` *newName*

Subqueries & `IN`

Data Manipulation Language

Introduction

Data Retrieval
- Subqueries
- Joins
- Union Operation

Update
- Insert
- Delete
- Update

Conclusion
- Suggested Readings

- You can use `IN` to say that the value in the expression must be among the values returned by the subquery
- You can also use the `IN` keyword with the NOT keyword in order to select rows when the value is not among the values returned by the subquery

Data Manipulation Language

Introduction

Data Retrieval

Subqueries

Joins

Union Operation

Update

Insert

Delete

Update

Conclusion

Suggested Readings

Subqueries: Example

Branch (branchNo, street, city, postcode)
Staff (staffNo, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

```
SELECT propertyNo, street, city, postcode, type,
    rooms, rent
FROM PropertyForRent
WHERE staffNo IN
    (SELECT staffNo
    FROM Staff
    WHERE branchNo ='B02')
```

- What does this query?

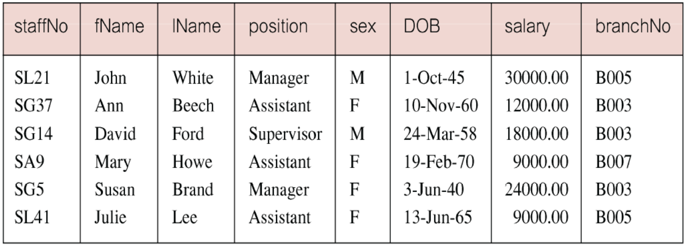
Introduction

Data Retrieval
Subqueries
Joins
Union Operation

Update
Insert
Delete
Update

Conclusion
Suggested Readings

Subqueries: Exercise I



staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000.00	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000.00	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000.00	B005

- List all staff whose salary is greater than the average salary, and show by how much

Subqueries: Exercise I

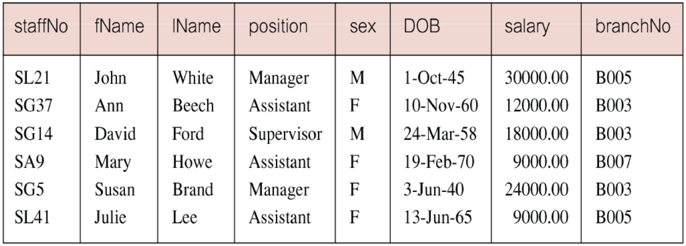
Data Manipulation Language

Introduction

Data Retrieval
Subqueries
Joins
Union Operation

Update
Insert
Delete
Update

Conclusion
Suggested Readings



staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000.00	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000.00	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000.00	B005

- List all staff whose salary is greater than the average salary, and show by how much

Subqueries: Exercise II

Data Manipulation Language

Introduction

Data Retrieval
Subqueries
Joins
Union Operation

Update
Insert
Delete
Update

Conclusion
Suggested Readings

Branch (branchNo, street, city, postcode)
Staff (staffNo, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

- List properties handled by staff at '163 Main St'.

ALL/ANY

Data Manipulation Language

Introduction

Data Retrieval

Subqueries

Joins

Union Operation

Update

Insert

Delete

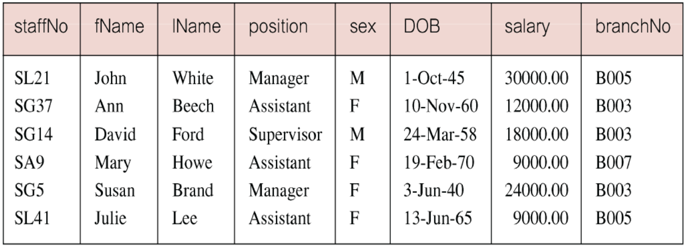
Update

Conclusion

Suggested Readings

- `ANY` and `ALL` may be used with subqueries that produce a single column of numbers
- With `ALL`, condition will only be true if it is satisfied by all values produced by subquery
- With `ANY`, condition will be true if it is satisfied by any values produced by subquery
- If subquery is empty, `ALL` returns true, `ANY` returns false

ALL/ANY: Example



staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000.00	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000.00	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000.00	B005

```
SELECT staffno, fname, lname, position, salary
FROM staff
WHERE salary > ALL (SELECT salary
   FROM staff
   WHERE branchno = 'B003')
```

- What does this query?

Data Manipulation Language

Introduction

Data Retrieval
Subqueries
Joins
Union Operation

Update
Insert
Delete
Update

Conclusion
Suggested Readings

`EXISTS`

- `EXISTS` and `NOT EXISTS` are for use only with subqueries
- Produce a simple true/false result
- True if and only if there exists at least one row in result table returned by subquery
- False if subquery returns an empty result table
- `NOT EXISTS` is the opposite of `EXISTS`
- As (NOT) EXISTS check only for existence or non-existence of rows in subquery result table, the subquery can contain any number of columns

EXISTS / NOT EXISTS: Example

Data Manipulation Language
Introduction
Data Retrieval
Subqueries
Joins
Union Operation
Update
Insert
Delete
Update
Conclusion
Suggested Readings

```
SELECT staffNo, fName, lName, position
FROM Staff AS s
WHERE EXISTS
          (SELECT *
          FROM Branch AS b
          WHERE s.branchNo = b.branchNo AND
                    city = 'London')
```

- What does this query?

Introduction
Data Retrieval
Subqueries
Joins
Union Operation
Update
Insert
Delete
Update
Conclusion
Suggested Readings

Subqueries: Exercise III

Branch (<u>branchNo</u>, street, city, postcode)
Staff (<u>staffNo</u>, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent (<u>propertyNo</u>, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

- List the name of members of the staff who do not manage any property

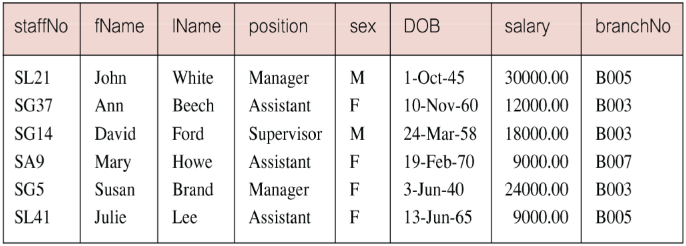
Subqueries in FROM

Subqueries are legal in a SELECT statement's FROM clause. The actual syntax is:

```
SELECT ... FROM (subquery) [AS] name ...
```

- The [AS] name clause is mandatory, because every table in a FROM clause must have a name
- Any columns in the subquery select list must have unique names

Subqueries in FROM: Example



staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000.00	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000.00	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000.00	B005

```
SELECT MAX(staffCount),branchNo
FROM (SELECT COUNT(staffNo) AS staffCount,branchNo
     FROM staff GROUP BY branchNo) AS staffInBranch;
```

What does this query?

Multi-Table Queries

Data Manipulation Language

Introduction

Data Retrieval
Subqueries
Joins
Union Operation

Update
Insert
Delete
Update

Conclusion
Suggested Readings

- If result columns come from more than one table must use a join
- To perform join, include more than one table in `FROM` clause
- Use comma as separator and typically include `WHERE` clause to specify join column(s)

Data Manipulation Language

Introduction

Data Retrieval

Subqueries

Joins

Union Operation

Update

Insert

Delete

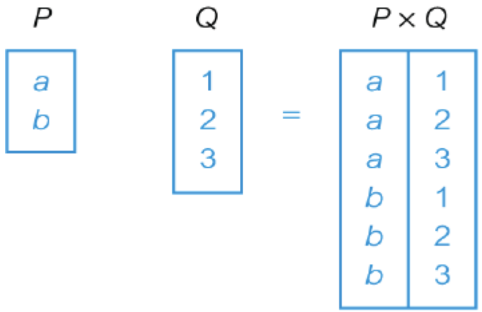
Update

Conclusion

Suggested Readings

Multi-Table Queries: Simple Example

```
SELECT *
FROM P, Q;
```



Data Manipulation Language

Introduction

Data Retrieval
- Subqueries
- Joins
- Union Operation

Update
- Insert
- Delete
- Update

Conclusion
- Suggested Readings

Multi-Table Queries: Example

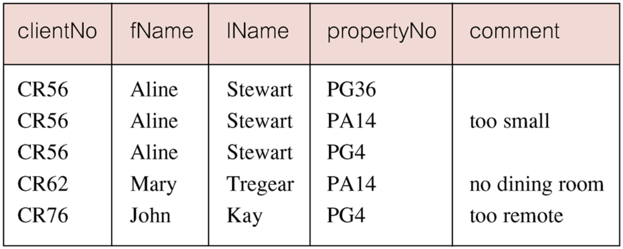
```
SELECT c.clientNo, fName, lName, propertyNo,
            comment
FROM Client c, Viewing v
WHERE c.clientNo = v.clientNo;
```

- What does this query?

Multi-Table Queries: Example

```
SELECT c.clientNo, fName, lName, propertyNo,
            comment
FROM Client c, Viewing v
WHERE c.clientNo = v.clientNo;
```

- Only those rows from both tables that have identical values in the `clientNo` columns (`c.clientNo = v.clientNo`) are included in result.

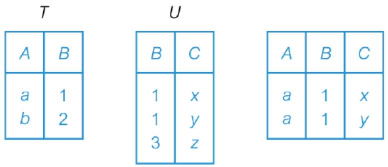


clientNo	fName	lName	propertyNo	comment
CR56	Aline	Stewart	PG36	
CR56	Aline	Stewart	PA14	too small
CR56	Aline	Stewart	PG4	
CR62	Mary	Tregear	PA14	no dining room
CR76	John	Kay	PG4	too remote

Join

Join

A Join operation is used to combine rows from two or more tables, based on a common field between them



T

A	*B*
a	1
b	2

U

B	*C*
1	*x*
1	*y*
3	*z*

A	*B*	*C*
a	1	*x*
a	1	*y*

Data Manipulation Language

Introduction

Data Retrieval
Subqueries
Joins
Union Operation

Update
Insert
Delete
Update

Conclusion
Suggested Readings

Joins

- SQL provides alternative ways to specify joins between tables:
  - `FROM Client c, Viewing v`
    `WHERE c.clientNo = v.clientNo`
  - `FROM Client c JOIN Viewing v ON`
    `    c.clientNo = v.clientNo`
  - `FROM Client JOIN Viewing USING clientNo`
  - `FROM Client NATURAL JOIN Viewing`
- In each case, FROM replaces original FROM and WHERE. However, first produces table with two identical clientNo columns

JOIN ON

Data Manipulation Language

Introduction

Data Retrieval
- Subqueries
- Joins
- Union Operation

Update
- Insert
- Delete
- Update

Conclusion
- Suggested Readings

```
FROM Client c JOIN Viewing v
 ON c.clientNo = v.clientNo
```

The `ON` clause determines the condition for making the join

Data Manipulation Language

Introduction

Data Retrieval
Subqueries
Joins
Union Operation

Update
Insert
Delete
Update

Conclusion
Suggested Readings

JOIN USING

```
FROM Client JOIN Viewing USING(clientNo)
```

The `USING`(*column_list*) clause names a list of columns that must exist in both tables

NATURAL JOIN

Data Manipulation Language

Introduction

Data Retrieval
Subqueries
Joins
Union Operation

Update
Insert
Delete
Update

Conclusion
Suggested Readings

```
FROM Client NATURAL JOIN Viewing
```

The `NATURAL JOIN` of two tables is defined to be semantically equivalent to an `JOIN` with a `USING` clause that names all columns that exist in both tables

Join: Exercise I

Branch (<u>branchNo</u>, street, city, postcode)
Staff (<u>staffNo</u>, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent (<u>propertyNo</u>, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

- For each branch, list numbers and names of staff who manage properties, and properties they manage

Join: Exercise II

Branch (<u>branchNo</u>, street, city, postcode)
Staff (<u>staffNo</u>, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent (<u>propertyNo</u>, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

- For each branch, list staff who manage properties, including city in which branch is located and properties they manage

Join: Exercise III

Branch (branchNo, street, city, postcode)
Staff (staffNo, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

- List the staff names and surnames together with the number of properties handled by each staff member

Data Manipulation Language

Introduction

Data Retrieval
Subqueries
Joins
Union Operation

Update
Insert
Delete
Update

Conclusion
Suggested Readings

Outer Joins

Often in joining two tables, a row in one table does not have a matching row in the other table; in other words, there is no matching value in the join columns.

Outer Join

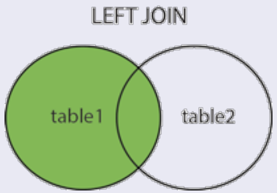
Return all rows from at least one of the tables even when there are no matching values in the other table

Types of Outer Joins

Left Join

Returns all rows from the left table (table1), with the matching rows in the right table (table2). The result is NULL in the right side when there is no match.

```
SELECT column_name(s)
FROM table1 LEFT JOIN table2
     ON table1.column_name=table2.column_name;
```



Left Join: Example

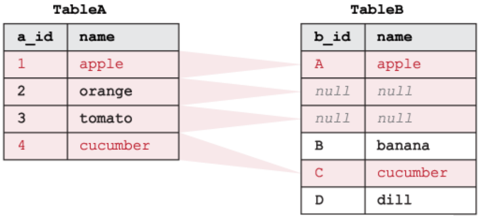
TableA

a_id	name
1	apple
2	orange
3	tomato
4	cucumber

TableB

b_id	name
A	apple
B	banana
C	cucumber
D	dill

```
SELECT *
FROM TableA
LEFT OUTER JOIN TableB
  ON tableA.name = tableB.name;
```



TableA

a_id	name
1	apple
2	orange
3	tomato
4	cucumber

TableB

b_id	name
A	apple
null	*null*
null	*null*
B	banana
C	cucumber
D	dill

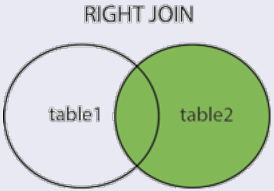
a_id	TableA.name	b_id	TableB.name
1	apple	A	apple
2	orange	*null*	*null*
3	tomato	*null*	*null*
4	cucumber	C	cucumber

Types of Outer Joins

Right Join

Returns all rows from the right table (table2), with the matching rows in the left table (table1). The result is NULL in the left side when there is no match

```
SELECT column_name(s)
FROM table1 RIGHT JOIN table2
     ON table1.column_name=table2.column_name;
```



Right Join: Example

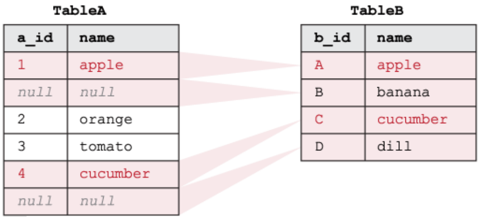
TableA

a_id	name
1	apple
2	orange
3	tomato
4	cucumber

TableB

b_id	name
A	apple
B	banana
C	cucumber
D	dill

```
SELECT *
FROM TableA
RIGHT OUTER JOIN TableB
  ON tableA.name = tableB.name;
```



TableA

a_id	name
1	apple
null	*null*
2	orange
3	tomato
4	cucumber
null	*null*

TableB

b_id	name
A	apple
B	banana
C	cucumber
D	dill

a_id	TableA.name	b_id	TableB.name
1	apple	A	apple
null	*null*	B	banana
4	cucumber	C	cucumber
null	*null*	D	dill

Types of Outer Joins

Data Manipulation Language

Introduction

Data Retrieval
Subqueries
Joins
Union Operation

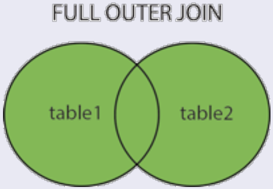
Update
Insert
Delete
Update

Conclusion
Suggested Readings

Full Join

The FULL OUTER JOIN keyword returns all rows from the left table (table1) and from the right table (table2).

```
SELECT column_name(s)
FROM table1 RIGHT JOIN table2
     ON table1.column_name=table2.column_name;
```



Full Join: Example

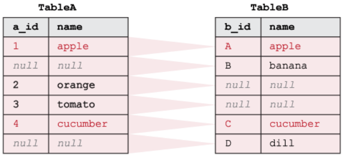
TableA

a_id	name
1	apple
2	orange
3	tomato
4	cucumber

TableB

b_id	name
A	apple
B	banana
C	cucumber
D	dill

```
SELECT *
FROM TableA
FULL OUTER JOIN TableB
  ON tableA.name = tableB.name;
```



TableA

a_id	name
1	apple
null	*null*
2	orange
3	tomato
4	cucumber
null	*null*

TableB

b_id	name
A	apple
B	banana
null	*null*
null	*null*
C	cucumber
D	dill

a_id	TableA.name	b_id	TableB.name
1	apple	A	apple
null	*null*	B	banana
2	orange	*null*	*null*
3	tomato	*null*	*null*
4	cucumber	C	cucumber
null	*null*	D	dill

Outer Joins in SQL

Data Manipulation Language

Introduction

Data Retrieval
Subqueries
Joins
Union Operation

Update
Insert
Delete
Update

Conclusion
Suggested Readings

- SQL provides ways to specify outer joins:

```
FROM Client c LEFT JOIN Viewing v ON
          c.clientNo = v.clientNo
FROM Client c RIGTH JOIN Viewing v ON
          c.clientNo = v.clientNo
FROM Client c FULL JOIN Viewing v ON
          c.clientNo = v.clientNo
```

Data Manipulation Language
Introduction
Data Retrieval
Subqueries
Joins
Union Operation
Update
Insert
Delete
Update
Conclusion
Suggested Readings

Outer Join: Exercise I

Branch (<u>branchNo</u>, street, city, postcode)
Staff (<u>staffNo</u>, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent (<u>propertyNo</u>, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

- List the staff names and surnames together with the number of properties handled by each staff member

Data Manipulation Language

Introduction

Data Retrieval
Subqueries
Joins
Union Operation

Update
Insert
Delete
Update

Conclusion
Suggested Readings

Union Operation

```
SELECT ...
UNION [ALL | DISTINCT]
SELECT ...
```

- UNION is used to combine the result from multiple SELECT statements into a single result set
- The column names from the first SELECT statement are used as the column names for the results returned
- Selected columns listed in corresponding positions of each SELECT statement should have the same data type.
- If `ALL` specified, result can include duplicate rows

UNION: Example

Data Manipulation Language

Introduction

Data Retrieval
- Subqueries
- Joins
- Union Operation

Update
- Insert
- Delete
- Update

Conclusion
- Suggested Readings

```
(SELECT city
FROM Branch
WHERE city IS NOT NULL)
UNION
(SELECT city
FROM PropertyForRent
WHERE city IS NOT NULL)
```

- What does this query?

Introduction
Data Retrieval
Subqueries
Joins
Union Operation
Update
Insert
Delete
Update
Conclusion
Suggested Readings

Data Updates

SQL can be used for modifying the data in the database:

- `INSERT` adds new rows of data to a table
- `UPDATE` modifies existing data in a table
- `DELETE` removes rows of data from a table

Data Manipulation Language

Introduction

Data Retrieval
- Subqueries
- Joins
- Union Operation

Update
- Insert
- Delete
- Update

Conclusion
- Suggested Readings

Insert Rows

`INSERT INTO` *table_name* [(columnList)] `VALUES` (dataValueList)

- columnList is optional; if omitted, SQL assumes a list of all columns in their original `CREATE TABLE` order
- Any columns omitted must have been declared as `NULL` when table was created, unless `DEFAULT` was specified when creating column
- dataValueList must match columnList

Data Manipulation Language

Introduction

Data Retrieval
- Subqueries
- Joins
- Union Operation

Update
- **Insert**
- Delete
- Update

Conclusion
- Suggested Readings

Insert Rows: Example

Branch (branchNo, street, city, postcode)
Staff (staffNo, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

Insert a new row into Staff table supplying data for all mandatory columns.

```
INSERT INTO Staff (staffNo, fName, lName,position, salary,
 branchNo)
VALUES ('SG44', 'Anne', 'Jones','Assistant', 8100, 'B003')
```

or

```
INSERT INTO Staff
VALUES ('SG44', 'Anne', 'Jones', 'Assistant', NULL, NULL,
 8100, 'B003');
```

Data Manipulation Language

Introduction

Data Retrieval
- Subqueries
- Joins
- Union Operation

Update
- **Insert**
- Delete
- Update

Conclusion
- Suggested Readings

Coping Rows

`INSERT INTO` *table_name* [(columnList)] selectStatement

- Allows multiple rows to be copied from one or more tables to another

Delete Rows

Data Manipulation Language

Introduction

Data Retrieval
Subqueries
Joins
Union Operation

Update
Insert
Delete
Update

Conclusion
Suggested Readings

```
DELETE FROM tableName [WHERE searchCondition]
```

- tableName can be name of a base table or an updatable view.
- searchCondition is optional; if omitted, all rows are deleted from table. This does not delete table. If searchCondition is specified, only those rows that satisfy condition are deleted.

Update Rows: Example

Data Manipulation Language

Introduction

Data Retrieval

Subqueries

Joins

Union Operation

Update

Insert

Delete

Update

Conclusion

Suggested Readings

Branch (<u>branchNo</u>, street, city, postcode)
Staff (<u>staffNo</u>, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent (<u>propertyNo</u>, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

Delete all properties that relate to branch B003.

```
DELETE FROM PropertyForRent
 WHERE branchNo = 'B003';
```

Data Manipulation Language

Introduction

Data Retrieval
- Subqueries
- Joins
- Union Operation

Update
- Insert
- Delete
- Update

Conclusion
- Suggested Readings

Update Rows

UPDATE tableName
SET columnName1 = dataValue1 [, columnName2 = dataValue2...] [WHERE searchCondition]

- tableName can be name of a base table or an updatable view.
- SET clause specifies names of one or more columns that are to be updated.

Update Rows: Example

Data Manipulation Language

Introduction

Data Retrieval
- Subqueries
- Joins
- Union Operation

Update
- Insert
- Delete
- **Update**

Conclusion
- Suggested Readings

Branch (branchNo, street, city, postcode)
Staff (staffNo, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

Give all staff a 3% pay increase.

```
UPDATE Staff SET salary = salary*1.03;
```

Update: Exercise I

Data Manipulation Language

Introduction

Data Retrieval

Subqueries

Joins

Union Operation

Update

Insert

Delete

Update

Conclusion

Suggested Readings

Branch (<u>branchNo</u>, street, city, postcode)
Staff (<u>staffNo</u>, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent (<u>propertyNo</u>, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

Assume there is a table StaffPropCount that contains names of staff and number of properties they manage:

StaffPropCount(<u>staffNo</u>, fName, lName, propCnt)

Populate StaffPropCount using Staff and PropertyForRent tables

Data Manipulation Language

Introduction

Data Retrieval
Subqueries
Joins
Union Operation

Update
Insert
Delete
Update

Conclusion
Suggested Readings

Update: Exercise II

Branch (branchNo, street, city, postcode)
Staff (staffNo, fName, lName, position, sex, DOB, salary, branchNo)
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

Give all Managers a 5% pay increase

Data Manipulation Language

Introduction

Data Retrieval
- Subqueries
- Joins
- Union Operation

Update
- Insert
- Delete
- Update

Conclusion
- Suggested Readings

Conclusion

In this session we have covered:

- SELECT
  - Subqueries, Joins, Union
- Data Update
  - Insert, Delete, Update

Lab Session

This week lab session more about performing queries and database updates

Introduction
Data Retrieval
Subqueries
Joins
Union Operation
Update
Insert
Delete
Update
Conclusion
Suggested Readings

Suggested Readings

- Chapters 4 and 5 of Fundamentals of Database Systems. Elmasri & Navathe.
- Chapters 5 and 6 of Database systems: a practical approach to design, implementation, and management. Connolly, Thomas M; Begg, Carolyn