DATABASE METHODOLOGY

Structured Query Language SQL

Aggregate Functions (SQL-DML)
SET Operators (SQL-DML)
Views (Formally SQL-DDL)

SQL - DML

- In this module you will learn about some advanced SQL-SELECT commands
  - How to use *aggregate functions* to retrieve aggregated data (e.g. summations or counts)
  - How to work with *set operators*
    - specifically, the *UNION* operator
  - How to use *views*
    - Formally part of SQL-DDL
      - but based on using the SELECT command

Aggregate Functions

- We often want to do more with data than just look at what’s directly in rows and columns
  - We want to do *aggregations* of (column) data
    - We use SQLs **aggregate functions**

Function:	**Returns as the result:**
COUNT(*)	**The number of rows in a table (or groups of columns)**
COUNT(*column*)	**The number of non-NULL values in the column**
SUM(*column*)	**The *sum* of the non-NULL values in the column**
AVG(*column*)	**The *average* of the non-NULL values in the column**
MIN(*column*)	**The *smallest* non-NULL value in the column**
MAX(*column*)	**The *highest* non-NULL value in the column**

An SQL Query With COUNT()

Stockholm University

Employee

<u>name</u>	salary	manager
Berg	20000	Flod
Flod	16000	Kvist
Bundy	19000	Kvist
Kvist	17000	Kvist
Rot	18000	Flod
Sten	18000	Kvist

How many employees have Kvist as manager?
(For simplicity: including himself.)

No Of Employees
4

```
SELECT COUNT(name) AS [No Of Employees]
FROM Employee
WHERE manager = 'Kvist'
```

OR

```
SELECT COUNT(*) AS [No Of Employees]
FROM Employee
WHERE manager = 'Kvist'
```

An SQL Query With Grouping

Stockholm University

- **GROUP BY** is used for grouping rows
    - Often used with aggregate functions

Find out how many employees there are in each department! Show department name and the number.

```
SELECT dept AS Department,
       COUNT(*) AS Employees
FROM Employee
GROUP BY dept
```

Employee

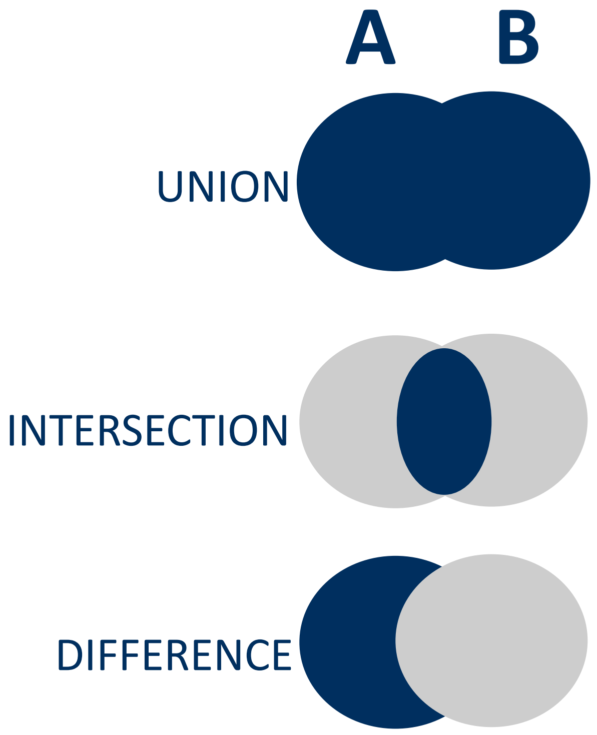
name	dept
Berg	Perfume
Flod	Perfume
Bundy	Shoes
Kvist	Toys
Rot	Sports
Sten	Perfume

name	dept
Berg	Perfume
Flod	Perfume
Sten	Perfume
Bundy	Shoes
Rot	Sports
Kvist	Toys

Department	Employees
Perfume	3
Shoes	1
Sports	1
Toys	1

UNION & Other SET Operations

- **SET operators**
  - analyze two sets of rows, the **operands**
  - return a single set of rows, the result set
  - require that the operands be **union compatible:**
    - they must have the *same number* of columns
    - and the matching columns the *same domains*
- We will look closer at the most common:
  - **UNION** – returns the rows in A and the rows in B, *but discards duplicate rows*.
- Other set operators. (Not part of this course!)
  - **UNION ALL** – same as UNION, but *keeps* duplicate rows
  - **INTERSECTION** - the rows that are in both A *and* B
  - **DIFFERENCE** - the rows that are in A *but not* in B



UNION

- **UNION** – returns the rows in A and the rows in B, *but discards duplicate rows*.

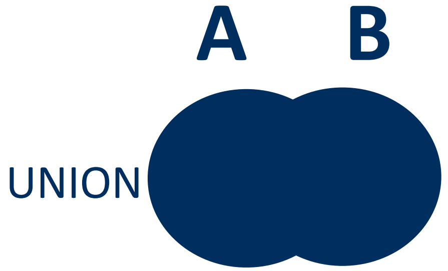
Stockholm University

Cat and Dog are made union compatible with each other:

```
Cat(name:String, age:int, ownedBy:String)
Dog(name:String, age:int, owner:String)
```

Pig is not union compatible with any of the other two:

```
Pig(name:String, age:int, price:int)
```



```
SELECT * FROM Cat
WHERE age >= 5
UNION
SELECT * FROM Dog
WHERE owner = 'Björn Borg'
```

← This UNION works

This does not →

```
SELECT * FROM Cat
WHERE age >= 5
UNION
SELECT * FROM Pig
WHERE age = 3
```

VIEWS – Formally Part Of SQL-DDL

- A view is basically a **named** SQL query
  - Actually defined by an SQL query
  - Can be called by name over and over
  - Only code stored (once), not result
    - Always showing updated data when run

```
CREATE VIEW V_Manager AS
SELECT manager AS BigShot,
          COUNT(name) AS Emps
FROM Employee
GROUP BY manager
```

Employee

<u>name</u>	salary	manager	department
Berg	20000	Kvist	Perfume
Kvist	16000	Kvist	Perfume
Bundy	19000	Flod	Shoes
Flod	17000	Flod	Shoes
Rot	18000	Kvist	Groceries
Sten	18000	Kvist	Perfume

V_Manager

BigShot	Emps
Kvist	4
Flod	2

Usage example

```
SELECT BigShot
FROM V_Manager
WHERE Emps > 2
```

BigShot
Kvist

SQL – DML Summary

- So, now you’ve learnt basics about
  - **Aggregate functions**
    - Also with grouping (GROUP BY)
  - **SET operations**
    - Specifically UNION
  - **VIEWS**
    - which are, in essence, named queries

Medverkande

Anders Thelemyr – Lärare

Lars In de Betou – Mediepedagog

Inspelat 2015-09-02
Institutionen för data- och systemvetenskap, DSV

Stockholms universitet