



COMP1140: Database and Information Management

Lecture Note – Week 08

Dr Suhuai Luo

School of EEC

University of Newcastle



Notice

- Assignment 2 is due this Friday (Sept 21) at 4 pm.
- SQL test
 - will be taken in week 9 (1st week after the break) in your own lab
 - No lecture in week 9. Use it as review & consulting. Any questions on the SQL test, pls come to see me in my office during the lecture time of week 9.
 - Details will be discuss at the end of this lecture
- A3 specification will be available on BB from 1pm today (Sept 18)



Last week

- Structured Query Language (SQL)
 - Historical Perspective
 - DDL
 - Integrity constraints and SQL
 - CREATE, ALTER and DROP
 - DML
 - INSERT
 - UPDATE
 - DELETE
 - SELECT Basics



Summary example

- For each branch **with more than 1 member of staff**, find number of staff in each branch and sum of their salaries.

```
SELECT  branchNo,  
        COUNT(staffNo) AS myCount,  
        SUM(salary) AS mySum  
FROM Staff  
GROUP BY branchNo  
HAVING COUNT(staffNo) > 1  
ORDER BY branchNo;
```



This week

- Continue SELECT statement
 - Joins
 - Subqueries
 - Set operations
- Reference – chapter 6
- Discussion on A3
- Discussion on SQL test



Multi-Table Queries

- If resultant 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).



Multi-Table Queries (contd.)

- Also possible to use an alias for a table named in FROM clause.
- Alias is separated from table name with a space.
- Alias can be used to qualify column names when there is ambiguity.



Example: Simple Join

- List names of all clients who have viewed a property along with any comment supplied.

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

Alias
name *c*
qualifies
that
clientNo
is from
client
table

Aliases

Join
condition



Example: Simple Join (contd.)

- Only those rows from both tables that have identical values in the clientNo columns ($c.\text{clientNo} = v.\text{clientNo}$) are included in result.
- Equivalent to equi-join in relational algebra.

Table 5.24 Result table for Example 5.24.

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



Aliases and Renaming

- Aliases for tables can be specified in the FROM clause using the optional AS keyword
 - Format: FROM Table [AS] Alias
 - E.g. `FROM Staff AS s` or `FROM Staff s`
- Column names in the result of SELECT can be renamed using the optional AS keyword
 - Format: SELECT column [AS] new-column-name
 - E.g. `SELECT staffNo AS StaffNumber` or `SELECT staffNo StaffNumber`



Alternative JOIN Constructs

- SQL provides alternative ways to specify joins:
 - `FROM Client c JOIN Viewing v ON c.clientNo = v.clientNo`
- In each case, FROM replaces original FROM and WHERE. However, this alternative produces table with two identical clientNo columns.
- Joins can be used with other clauses – such as GROUP BY, ORDER BY



Computing a Join - Procedure for generating results of a SELECT with a join:

1. Form Cartesian product of the tables named in FROM clause.
2. If there is a WHERE clause, apply the search condition to each row of the product table, retaining those rows that satisfy the condition.
3. Apply any GROUP BY - HAVING clauses, if specified, to the remaining rows.
4. For each row in result, determine value for each item in SELECT list.
5. If DISTINCT has been specified, eliminate any duplicate rows from the result.
6. If there is an ORDER BY clause, sort result table as required.



Example: Sorting with a join

- For each branch, list:
 - branch number, s.staffNo & fName & IName of staff who manage properties, and propertyNo of properties they manage
 - in ascending order of branchNo, staffNo and propertyNo.

```
SELECT s.branchNo, s.staffNo, fName, IName,  
       propertyNo  
FROM Staff s, PropertyForRent p  
WHERE s.staffNo = p.staffNo  
ORDER BY s.branchNo, s.staffNo, propertyNo;
```



Example: Sorting with a join

Table 5.25 Result table for Example 5.25.

branchNo	staffNo	fName	lName	propertyNo
B003	SG14	David	Ford	PG16
B003	SG37	Ann	Beech	PG21
B003	SG37	Ann	Beech	PG36
B005	SL41	Julie	Lee	PL94
B007	SA9	Mary	Howe	PA14




Example: Three Table Join

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

Multiple
join
conditions

```
SELECT b.branchNo, b.city, s.staffNo, fName,  
       IName, propertyNo  
FROM   Branch b, Staff s, PropertyForRent p  
WHERE  b.branchNo = s.branchNo AND  
       s.staffNo = p.staffNo  
ORDER BY b.branchNo, s.staffNo, propertyNo;
```





Example: Three Table Join (contd.)

Table 5.26 Result table for Example 5.26.

branchNo	city	staffNo	fName	lName	propertyNo
B003	Glasgow	SG14	David	Ford	PG16
B003	Glasgow	SG37	Ann	Beech	PG21
B003	Glasgow	SG37	Ann	Beech	PG36
B005	London	SL41	Julie	Lee	PL94
B007	Aberdeen	SA9	Mary	Howe	PA14

- Alternative formulation for FROM and WHERE:

FROM (Branch b JOIN Staff s USING branchNo)
JOIN PropertyForRent p USING staffNo



Example: Multiple Grouping Columns

- Find number of properties handled by each staff member. Display branchNo, staffNo and the number

```
SELECT p.branchNo, p.staffNo, COUNT(*) AS myCount
FROM PropertyForRent p
GROUP BY p.branchNo, p.staffNo
ORDER BY p.branchNo, p.staffNo;
```

branchNo	staffNo	myCount
B003	SG14	1
B003	SG37	2
B005	SL41	1
B007	SA9	1

Example: Multiple Grouping Columns

Find number of properties handled by each staff member. Display branchNo, staffNo, *fName* and the number

```
SELECT s.branchNo, s.staffNo, s.fName, COUNT(*) AS myCount
FROM Staff s, PropertyForRent p
WHERE s.staffNo = p.staffNo
GROUP BY s.branchNo, s.staffNo, s.fName
ORDER BY s.branchNo, s.staffNo;
```

branchNo	staffNo	fName	myCount
B003	SG14	David	1
B003	SG37	Ann	2
B005	SL41	Julie	1
B007	SA9	Mary	1



Cross Joins

- In SQL, you can specify CROSS joins in multiple ways:

- Specify tables in FROM clause without a join condition in the WHERE clause,

```
SELECT    [DISTINCT | ALL]          {*} | columnList}  
FROM Table1, Table2
```

or

- Use CROSS JOIN keyword:

```
SELECT    [DISTINCT | ALL]          {*} | columnList}  
FROM Table1 CROSS JOIN Table2
```



Outer Joins

- If one row of a joined table is unmatched, row is omitted from result table.
- Outer join operations *also* retain rows that do not satisfy the join condition.
- Consider following tables:

Branch1

branchNo	bCity
B003	Glasgow
B004	Bristol
B002	London

PropertyForRent1

propertyNo	pCity
PA14	Aberdeen
PL94	London
PG4	Glasgow



Outer Joins

~~The (inner) join of these two tables:~~

```
SELECT b.*, p.*  
FROM Branch1 b, PropertyForRent1 p  
WHERE b.bCity = p.pCity;
```

Table 5.27(b) Result table for inner join of Branch1 and PropertyForRent1 tables.

branchNo	bCity	propertyNo	pCity
B003	Glasgow	PG4	Glasgow
B002	London	PL94	London



Outer Joins (contd.)

- Result table has two rows where cities are same.
- There are no rows corresponding to branch in Bristol, and property in Aberdeen.
- To include unmatched rows in result table, use an Outer join.



Example: Left Outer Join

- List branches and properties that are in same city along with any unmatched branches.

```
SELECT b.*, p.*  
FROM Branch1 b LEFT JOIN PropertyForRent1 p  
    ON b.bCity = p.pCity;
```

Example: Left Outer Join (contd.)

- Includes those rows of first (left) table unmatched with rows from second (right) table.
- Unmatched columns from second table are filled with NULLs.

Table 5.28 Result table for Example 5.28.

branchNo	bCity	propertyNo	pCity
B003	Glasgow	PG4	Glasgow
B004	Bristol	NULL	NULL
B002	London	PL94	London



Example: Right Outer Join

- List branches and properties in same city and any unmatched properties.

```
SELECT b.*, p.*
```

```
FROM Branch1 b RIGHT JOIN PropertyForRent1 p  
ON b.bCity = p.pCity;
```



Example: Right Outer Join (contd.)

- Right Outer join includes those rows of second (right) table that are unmatched with rows from first (left) table.
- Unmatched columns from first table are filled with NULLs.

Table 5.29 Result table for Example 5.29.

branchNo	bCity	propertyNo	pCity
NULL	NULL	PA14	Aberdeen
B003	Glasgow	PG4	Glasgow
B002	London	PL94	London



Example: Full Outer Join

- List branches and properties in same city and any unmatched branches or properties.

```
SELECT b.*, p.*
```

```
FROM Branch1 b FULL JOIN PropertyForRent1 p  
    ON b.bCity = p.pCity;
```

Example: Full Outer Join (contd.)

- Includes rows that are unmatched in both tables.
- Unmatched columns are filled with NULLs.

Table 5.30 Result table for Example 5.30.

branchNo	bCity	propertyNo	pCity
NULL	NULL	PA14	Aberdeen
B003	Glasgow	PG4	Glasgow
B004	Bristol	NULL	NULL
B002	London	PL94	London



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*.
- Subselects may also appear in INSERT, UPDATE, and DELETE statements.



Example: Subquery with Equality

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

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



Example: Subquery with Equality (contd.)

- 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.
- Outer SELECT then becomes:

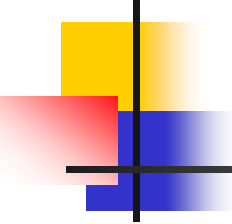
```
SELECT staffNo, fName, lName, position  
FROM Staff  
WHERE branchNo = 'B003';
```



Example: Subquery with Equality (contd.)

Table 5.19 Result table for Example 5.19.

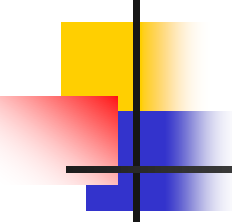
staffNo	fName	lName	position
SG37	Ann	Beech	Assistant
SG14	David	Ford	Supervisor
SG5	Susan	Brand	Manager



Example: Subquery with Aggregate

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

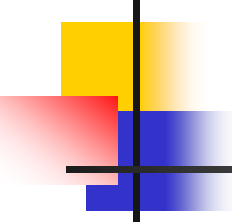
```
SELECT staffNo, fName, lName, position,  
       salary – (SELECT AVG(salary) FROM Staff) As SalDiff  
FROM Staff  
WHERE salary >  
       (SELECT AVG(salary)  
        FROM Staff);
```



Example: Subquery with Aggregate (contd.)

- Cannot write 'WHERE salary > AVG(salary)'
- Instead, use subquery to find average salary (17000), and then use outer SELECT to find those staff with salary greater than this:

```
SELECT staffNo, fName, lName, position,  
       salary - 17000 As salDiff  
FROM Staff  
WHERE salary > 17000;
```



Example: Subquery with Aggregate (contd.)

Table 5.20 Result table for Example 5.20.

staffNo	fName	lName	position	salDiff
SL21	John	White	Manager	13000.00
SG14	David	Ford	Supervisor	1000.00
SG5	Susan	Brand	Manager	7000.00



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 in subquery 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.



Subqueries returning multiple rows

- When subqueries return multiple rows, the comparison operator must be followed by an ANY/SOME or ALL
- E.g. $> ANY$
 $> ALL$
- $=ANY$ or $=SOME$ is equivalent to IN



Example: Nested subquery: use of IN

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

```
SELECT propertyNo, street, city, postcode, type, rooms,  
       rent  
FROM PropertyForRent  
WHERE staffNo IN  
       (SELECT staffNo  
        FROM Staff  
        WHERE branchNo =  
              (SELECT branchNo  
               FROM Branch  
               WHERE street = '163 Main St'));
```

Example: Nested subquery: use of IN

PropertyForRent

propertyNo	street	city	postcode	type	rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40		B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003

Table 5.21 Result table for Example 5.21.

propertyNo	street	city	postcode	type	rooms	rent
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375
PG21	18 Dale Rd	Glasgow	G12	House	5	600



ANY and ALL

- 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* (1 or more) values produced by subquery.
- If subquery is empty, ALL returns true, ANY returns false.
- SOME may be used in place of ANY.



Example: Use of ANY/SOME

- Find staff whose salary is larger than salary of at least one member of staff at branch B003.

```
SELECT staffNo, fName, lName, position, salary
FROM Staff
WHERE salary > SOME
    (SELECT salary
     FROM Staff
     WHERE branchNo = 'B003');
```



Example: Use of ANY/SOME (contd.)

- Inner query produces set {12000, 18000, 24000} and outer query selects those staff whose salaries are greater than any of the values in this set.

Table 5.22 Result table for Example 5.22.

staffNo	fName	lName	position	salary
SL21	John	White	Manager	30000.00
SG14	David	Ford	Supervisor	18000.00
SG5	Susan	Brand	Manager	24000.00



Example: Use of ALL

- Find staff whose salary is larger than salary of every member of staff at branch B003.

```
SELECT staffNo, fName, lName, position, salary
FROM Staff
WHERE salary > ALL
    (SELECT salary
     FROM Staff
     WHERE branchNo = 'B003');
```



Example: Use of ALL

Table 5.23 Result table for Example 5.23.

staffNo	fName	lName	position	salary
SL21	John	White	Manager	30000.00



Correlated nested queries

- Correlated nested queries are subqueries that join with tables from the outer query
- Commonly used with EXISTS and NOT EXISTS keywords



EXISTS and NOT EXISTS

- EXISTS and NOT EXISTS are for use only with subqueries.
- Produce a simple true/false result.
- EXISTS is True if and only if there exists at least one row in result table returned by subquery.
- EXISTS is False if subquery returns an empty result table.
- NOT EXISTS is the opposite of EXISTS.



EXISTS and NOT EXISTS

- As (NOT) EXISTS check only for existence or non-existence of rows in subquery result table, subquery can contain any number of columns.
- It's common for subqueries following (NOT) EXISTS to be of form:

(SELECT * ...)

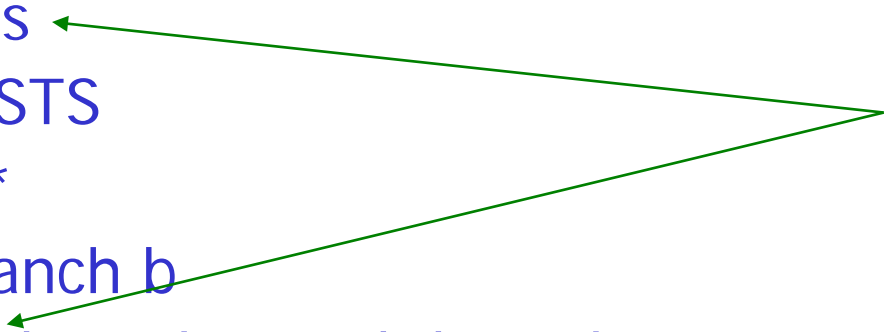


Example: Query using EXISTS

- Find all staff who work in a London branch.

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

Correlated
nested query





Example: Query using EXISTS (contd.)

- Note, search condition `s.branchNo = b.branchNo` is necessary to consider correct branch record for each member of staff.
- If omitted, would get all staff records listed out because subquery:

```
SELECT * FROM Branch WHERE city='London'
```

would always be true and query would be:

```
SELECT staffNo, fName, IName, position FROM Staff  
WHERE true;
```



Example: Query using EXISTS

Table 5.31 Result table for Example 5.31.

staffNo	fName	lName	position
SL21	John	White	Manager
SL41	Julie	Lee	Assistant



Example: Query using EXISTS (contd.)

- Could also write this query using join construct:

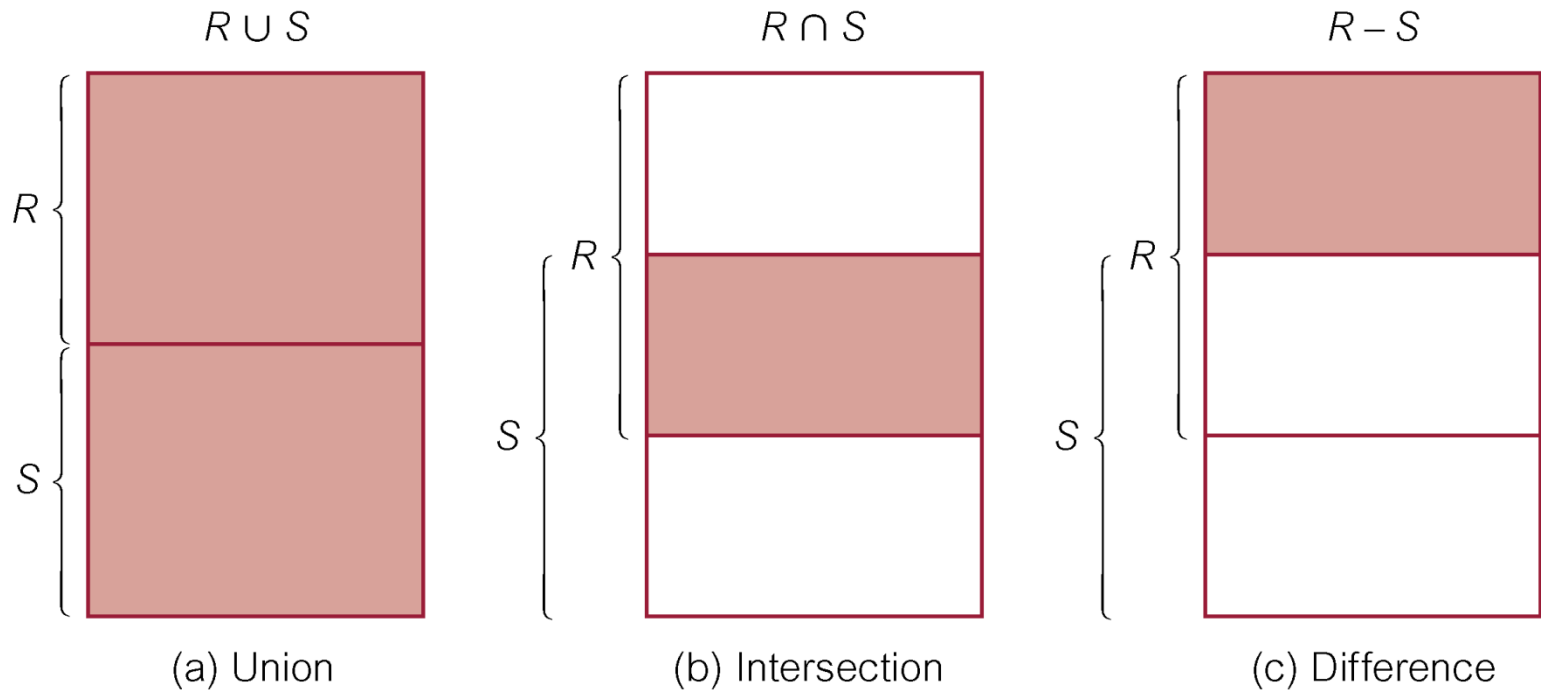
```
SELECT staffNo, fName, lName, position  
FROM Staff s, Branch b  
WHERE s.branchNo = b.branchNo AND  
       city = 'London';
```



Union, Intersect, and Difference (Except)

- Can use normal set operations of Union, Intersection, and Difference to combine results of two or more queries into a single result table.
- Union of two tables, A and B, is table containing all rows in either A or B or both.
- Intersection is table containing all rows common to both A and B.
- Difference is table containing all rows in A but not in B.
- Two tables must be *union compatible*.

Union, Intersect, and Difference (Except) (contd.)





Example: Use of UNION

- List all cities where there is either a branch office or a property.

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



Example: Use of UNION (contd.)

- Produces result tables from both queries and merges both tables together.

Table 5.32 Result table for Example 5.32.

city
London
Glasgow
Aberdeen
Bristol



Example: Use of ALL

- UNION ALL – allows duplicate rows in the result

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




Example: Use of INTERSECT

- List all cities where there is both a branch office and a property.

(SELECT city FROM Branch)

INTERSECT

(SELECT city FROM PropertyForRent);



Example: Use of INTERSECT (contd.)

- Could rewrite this query without INTERSECT operator:

```
SELECT b.city  
FROM Branch b PropertyForRent p  
WHERE b.city = p.city;
```

- Or:

```
SELECT DISTINCT city FROM Branch b  
WHERE EXISTS  
(SELECT * FROM PropertyForRent p  
WHERE p.city = b.city);
```



Example: Use of EXCEPT

- List of all cities where there is a branch office but no properties.

```
(SELECT city FROM Branch)  
EXCEPT  
(SELECT city FROM PropertyForRent);
```

Table 5.34 Result table for Example 5.34.

city
Bristol



Example: Use of EXCEPT (contd.)

- Could rewrite this query without EXCEPT:

```
SELECT DISTINCT city FROM Branch  
WHERE city NOT IN  
  (SELECT city FROM PropertyForRent);
```

- Or

```
SELECT DISTINCT city FROM Branch b  
WHERE NOT EXISTS  
  (SELECT * FROM PropertyForRent p  
   WHERE p.city = b.city);
```



Summary

- Joins
 - Inner, Outer
- Subqueries (nested queries)
 - Non-correlated
 - Correlated
- Set Operations
 - UNION, INTERSECT, EXCEPT



Lab This Week

- Continue SQL query practice on two cases:
Hotel and Registration



Issues and discussions on Assignment 3

- [A3 specification](#)
- [A3marking guide](#)



Discussion - InLab test week 9

- Weighting: 10% of total course score
- Content
 - practical T-SQL test; manage and query on a database
- Style
 - *Close book*, 1hr in lab test
 - Note: you can bring in one page with the tables' content only
- Arrangement
 - **You need to attend your lab session**
 - The database and related setup script will be available from October 1 from
BB->Content->Assessment->DatabaseCreatingCodeForSQLtest
 - You have to work on the virtual SQL from lab PC
- How to prepare
 - All SQL-related contents in lectures & labs (mainly weeks 7 & 8) & more practice
- Q