Dr Greg Wadley
Dr Renata Borovica-Gajic
David Eccles

INFO90002 Database Systems & Information Modelling

11. SQL Part 3

Query Nesting, Comparison Operators, EXISTS, NOT EXISTS, VIEWS

Relational Algebra Homework

Optional - Solutions

Homework solutions

1. Find the name of all sailors whose rating is above 9

$$\pi_{sname}(\sigma_{rating>9}(Sailors))$$

2. Find all sailors who reserved a boat prior to November 1, 1996

$$\pi_{sname}(Sailors \rhd\lhd \sigma_{day<"11/01/96"}(Reserves))$$

3. Find (the names of) all boats that have been reserved at least once

$$\pi_{bname}(Boats \rhd\lhd Reserves)$$

Homework solutions

4. Find all pairs of sailors with the same rating

$$\rho(S1(1 \to sid1, 2 \to sname1, 3 \to rating1, 4 \to age1), Sailors)$$

$$\rho(S2(1 \to sid2, 2 \to sname2, 3 \to rating2, 4 \to age2), Sailors)$$

$$\pi_{sname1,sname2}\left(S1 \bowtie_{rating1=rating2 \wedge sid1!=sid2} S2\right)$$

NEWS

- Assignment 1 is due 9 a.m Wednesday 9th September 2020
- Discussion forum will be closed 9 a.m. Monday 7th
  - To ensure that there is no confusing or contradictory messages on the forum by accident or design.
- Assignment 2 is released this week

Query Nesting

- SQL provides the ability to *nest* subqueries
- A nested query is simply another select query you write to produce a table set
  - Remember that all select queries return a table set of data
- A common use of subqueries is to perform set tests
  - Set membership, set comparisons

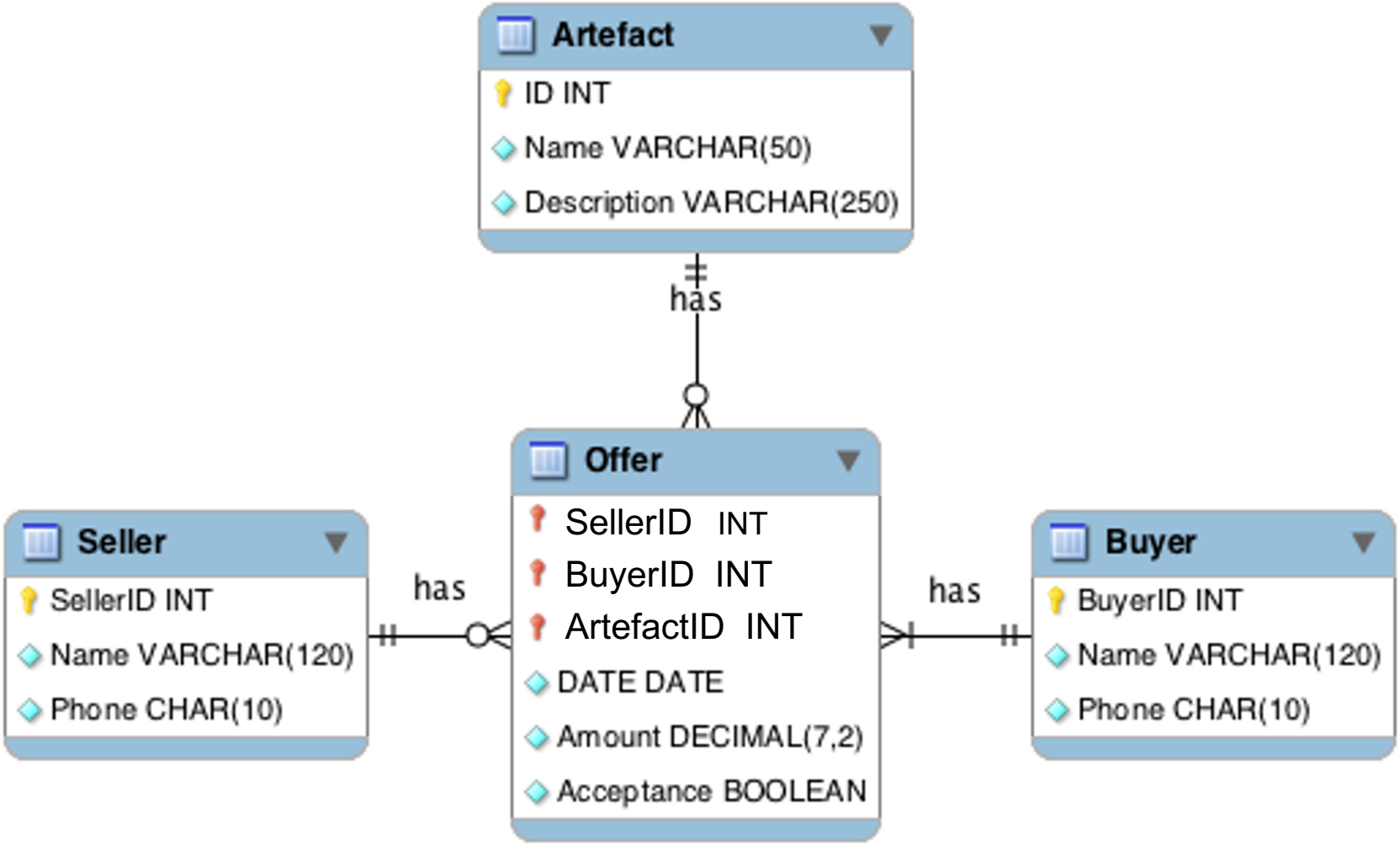
Sub-Query Comparison Operators

- IN / NOT IN
  - Used to test whether the attribute is IN/NOT IN the subquery list
- ANY
  - True if any value returned meets the condition
- ALL
  - True if all values returned meet the condition
- EXISTS
  - True if the subquery returns one or more records

- For more info:
- https://www.w3schools.com/sql/sql_any_all.asp
- https://www.w3schools.com/sql/sql_exists.asp

- General help with SQL: https://www.w3schools.com/sql/ (great tutorial)

Auction Bids – Physical Model



Tables

Artefact

ID	Name	Description
1	Vase	Ming Vase H50cm W30cm
2	Sketch	Early Modern Dutch School
3	Pot	CopperUS 18th Century

Seller

SellerID	Name	Phone
1	Ann	0509 123 321
2	Bill	0518 234 432
3	Carol	02 8344 4777

Buyer

BuyerID	Name	Phone
1	Maggie	0539 335 577
2	Nigel	0519 434 389
3	Olga	13 24 35

Offer

SellerID	BuyerID	ArtefactID	OfferDate	Ammount	Acceptance
1	1	1	2012-06-20	81223.23	0
1	1	2	2012-06-20	82223.23	0
2	2	1	2012-06-20	1995.50	0
2	2	2	2012-06-20	2300.15	0

Example: Subquery

- List the BuyerID, Name and Phone number for all bidders on artefact 1

```
1  SELECT *
2  FROM Buyer
3  WHERE buyerID IN
4        (SELECT BuyerID
5         FROM Offer
6         WHERE artefactid = 1);
```

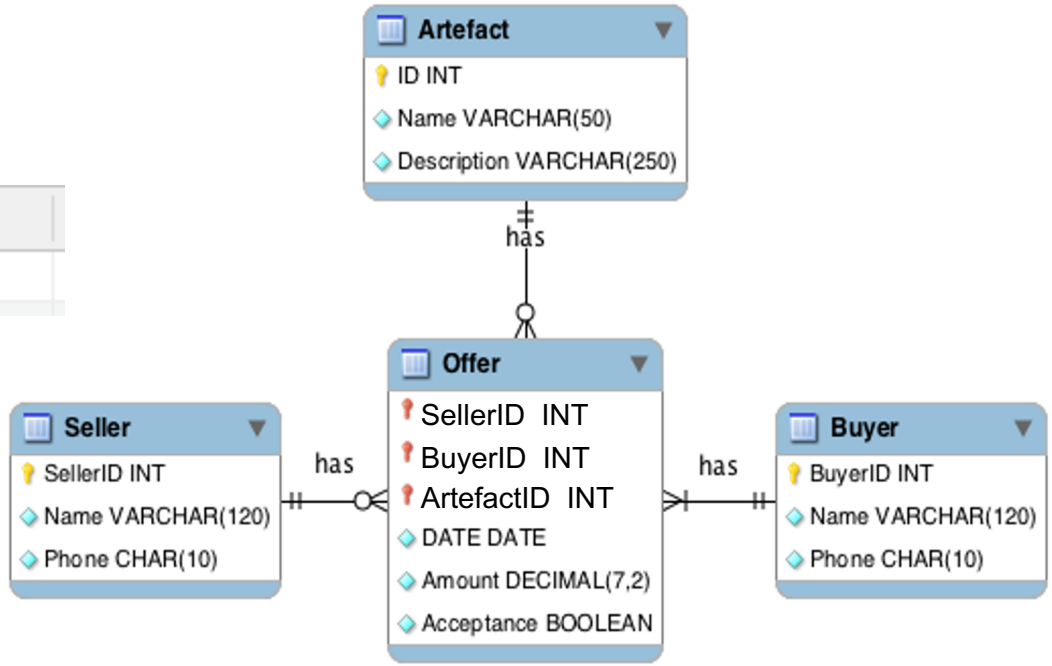
BuyerID	Name	Phone
1	Maggie	0539 335 577
2	Nigel	0519 434 389

More examples using subqueries

- Which Artefacts don't have any offers made on them

```
SELECT *
FROM artefact
WHERE ID NOT IN
     (SELECT artefactid
      FROM offer);
```

ID	Name	Description
3	Pot	CopperUS 18th Century



You try …

Which Buyers haven't made a bid for the "Pot" Artefact ?

Do we need to use IN? Is there another way…

- List the BuyerID, Name and Phone number for all bidders on artefact 1

```
SELECT *
FROM Buyer
WHERE BuyerID IN
    (SELECT BuyerID
    FROM Offer
    WHERE ArtefactID = 1)
```

- Is equal to:

```
SELECT BuyerID, Name, Phone
FROM Buyer NATURAL JOIN Offer
WHERE ArtefactID = 1
```

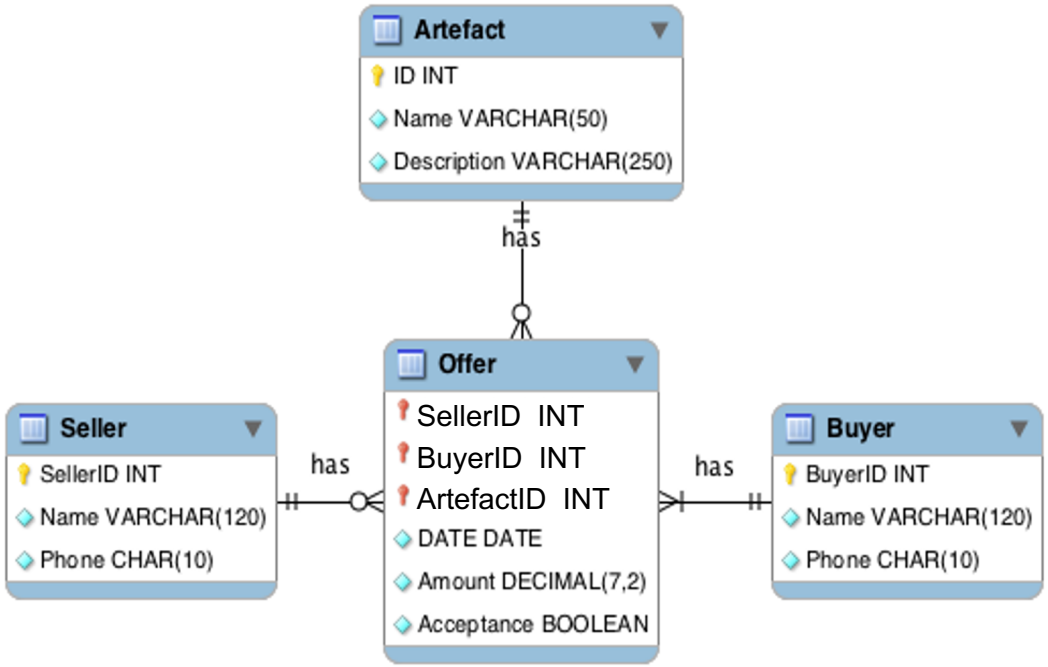
This is a more efficient way

Exists example

- Returns true if the subquery returns one or more records
- **Example**: *List the BuyerID, Name and Phone number for all bidders on artefact 1*

```
SELECT * FROM Buyer
WHERE EXISTS
      (SELECT * FROM Offer
       WHERE Buyer.BuyerID = Offer.BuyerID
       AND ArtefactID = 1)
```

BuyerID	Name	Phone
1	Maggie	0333333333
2	Nicole	0444444444



Views

- Any relation that is not in the physical models, but is made available to the "user" as a virtual relation is called a view.
- Views are good because:
  - They help hide the query complexity from users
  - They help hide data from users
    - Different users use different views
      - Prevents someone from accessing the employee tables to see salaries for instance
    - One way of improving database security
- Create view statement:

  CREATE VIEW nameofview **AS** validsqlstatement

- Once a view is defined
  - Its definition is stored in the database (not the data, but metadata – schema information)
  - Can be used just like any other table

Create View Example

```
CREATE VIEW EmpPay AS
SELECT   Employee.ID, Employee.Name, DateHired,
         EmployeeType, HourlyRate AS Pay
         FROM Employee INNER JOIN Hourly
         ON   Employee.ID = Hourly.ID
UNION
SELECT   Employee.ID, Employee.Name, DateHired,
         EmployeeType, AnnualSalary  AS Pay
         FROM Employee INNER JOIN Salaried
         ON   Employee.ID = Salaried.ID
UNION
SELECT   Employee.ID, Employee.Name, DateHired,
         EmployeeType, BillingRate  AS Pay
         FROM Employee INNER JOIN Consultant
         ON     Employee.ID = Consultant.ID;
```

Create VIEW example

```
CREATE VIEW v_DEPT_SALARY AS
SELECT department.departmentid, department.name,
sum(employee.salary) AS DEPTSAL
FROM department INNER JOIN employee ON
department.departmentid = employee.departmentid
GROUP BY department.departmentid, department.name;
```

Using a View

```
SELECT * FROM EmpPay;
```

Output | Snippets | Query 1 Result | Lecture7.sql R

ID	Name	DateHired	EmployeeType	Pay
3	Alice	2012-12-02	H	23.43
4	Alan	2010-01-22	H	29.43
1	Sean	2012-02-02	S	92000.00
2	Linda	2011-06-12	S	92300.00
5	Peter	2010-09-07	C	210.00
6	Rich	2012-05-19	C	420.00

```
SELECT * FROM EmpPay
    WHERE EmployeeType = "H" OR EmployeeType = "C"
```

Output | Snippets | Query 1 Result | Lecture7.sql Result

Fetched 4 records. Duration: 0.000 sec, fetc

ID	Name	DateHired	EmployeeType	Pay
3	Alice	2012-12-02	H	23.43
4	Alan	2010-01-22	H	29.43
5	Peter	2010-09-07	C	210.00
6	Rich	2012-05-19	C	420.00

More DDL Commands

- There are more than CREATE!
- ALTER
  - Allows us to add or remove attributes (columns) from a relation (table)
    - ALTER TABLE TableName ADD AttributeName AttributeType
    - ALTER TABLE TableName DROP AttributeName
- RENAME
  - Allows the renaming of tables (relations)
    - RENAME TABLE CurrentTableName TO NewTableName

More DDL Commands

- TRUNCATE
  - Same as DELETE * FROM table;
  - Faster but cannot ROLL BACK a TRUNCATE command
    - Have to get data back from backup…
- DROP
  - Potentially DANGEROUS
    - Kills a relation – removes the data, removes the relation
      - There is NO UNDO COMMAND! (have to restore from backup)
    - DROP TABLE TableName
- CTAS (CREATE TABLE as SELECT)

```
CREATE TABLE New_BankHQ
AS
SELECT * FROM
BankHQ
```

```
CREATE TABLE New_BankHQ
AS
SELECT * FROM
BankHQ
WHERE 1=0;
-- Handy to create table structure with no rows
```

Data Control Language / Other Commands

- DCL
  - Users and permissions
    - **CREATE USER, DROP USER**
    - **GRANT, REVOKE**
    - **SET PASSWORD**
- Other Commands
  - Database administration
    - **BACKUP TABLE, RESTORE TABLE**
    - **ANALYZE TABLE**
  - Miscellaneous
    - **DESCRIBE tablename**
    - **USE db_name**
- They are typically called ‘Database Administration Statements’

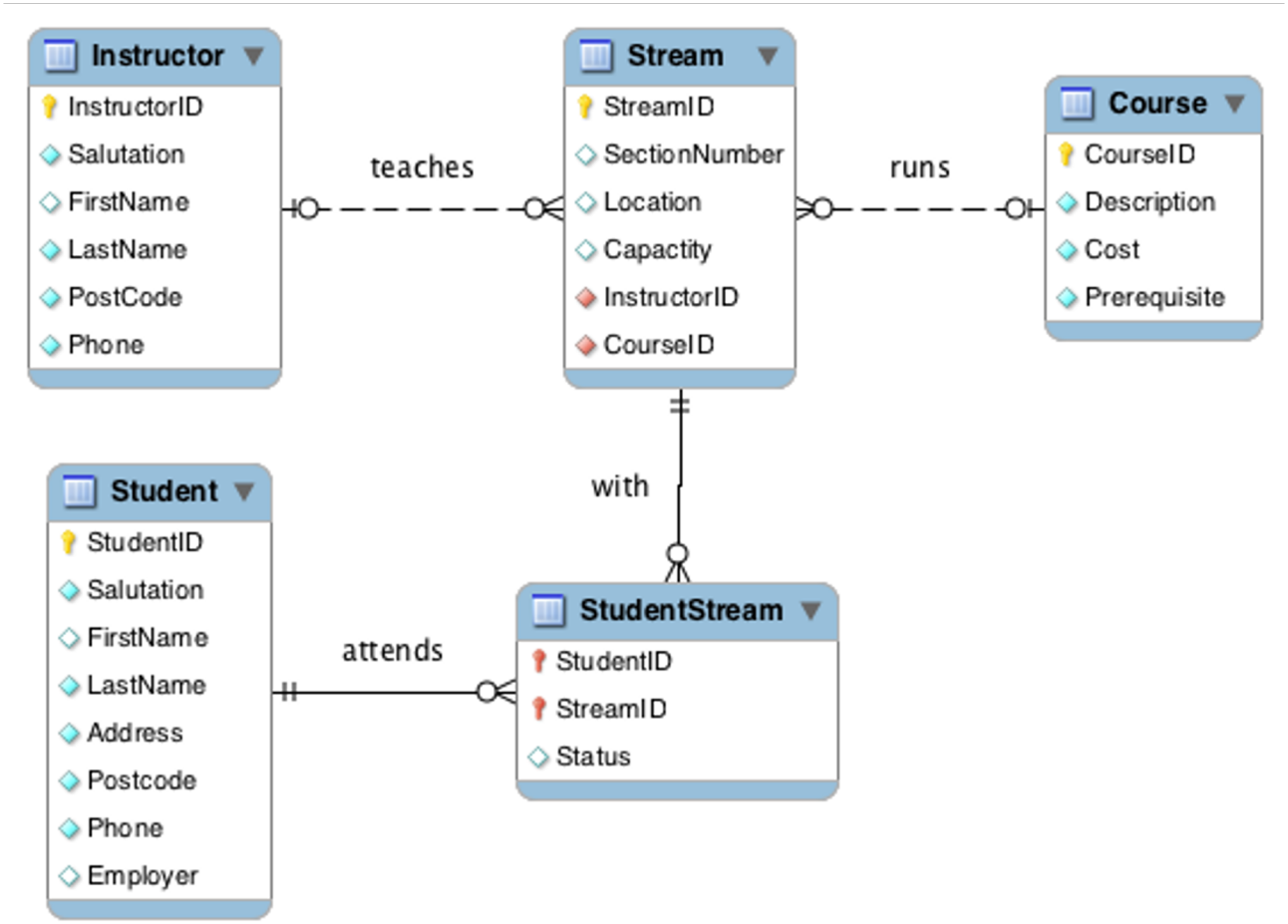
How to Think like SQL

- It's going to be critical for you to think like SQL to handle the queries you will need to write…

- Hopefully the following discussion will help you in this endeavour:
  1. USE the database design as a MAP to help you when you are formulating queries
  2. USE the structure of the SELECT statement as a template
  3. FILL out parts of the SELECT structure and BUILD the query

- Let's try it!

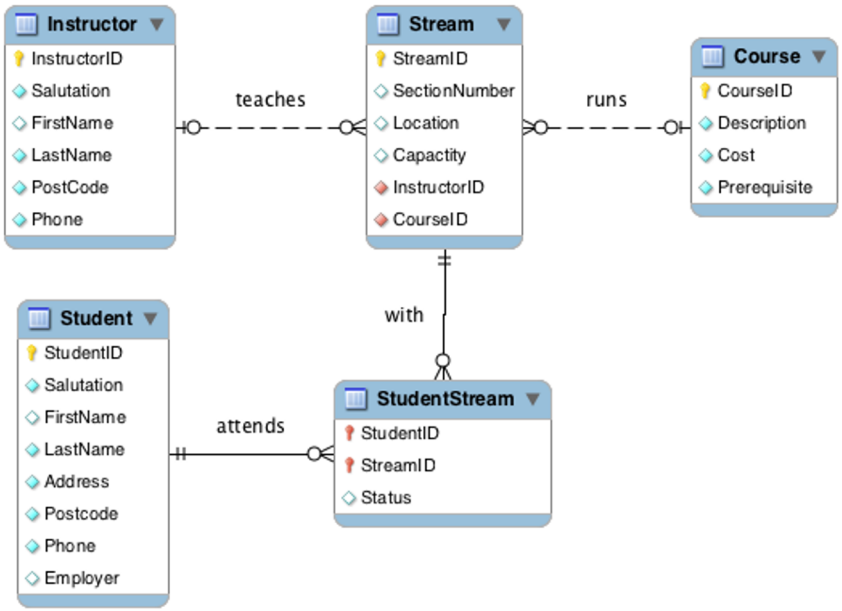
Example



Example: *Which employers employ students who are doing a course in locations where the capacity is greater than 20 persons, and what are those locations?*

How to approach writing queries

Which employers employ students who are doing a course in locations where the capacity is greater than 20 persons, and what are those locations?



- What is the query asking for:
  - Which fields & tables:
    - F: Employer, Location, Capacity
    - T: Student, Stream, StudentStream
  - But only if the capacity > 20 (condition)

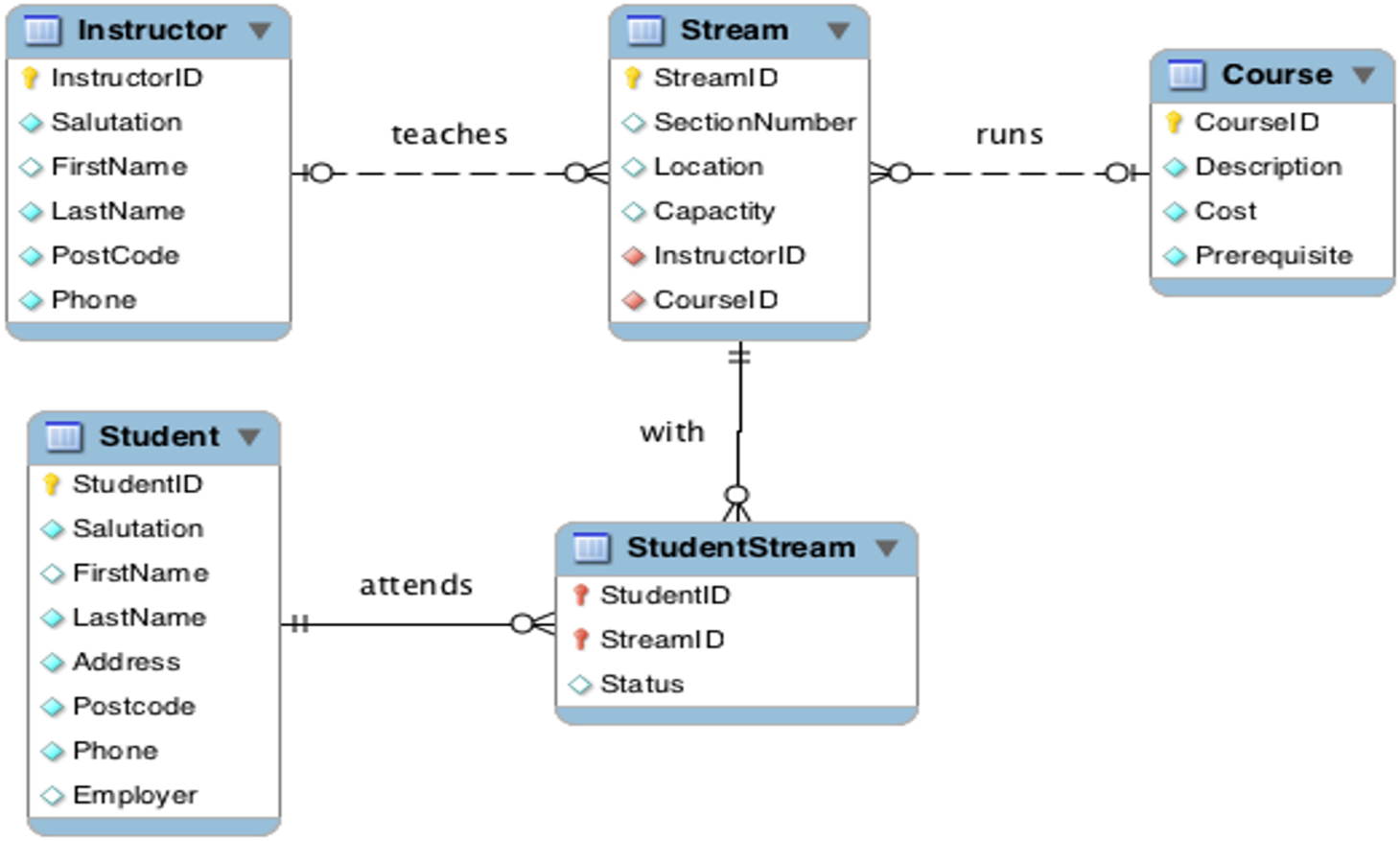
- Lets try to use the structure of the SELECT statement now:

```
SELECT Employer, Location, Capacity
FROM Student INNER JOIN StudentStream
ON Student.StudentID = StudentStream.StudentID
INNER JOIN Stream
ON StudentStream.StreamID = Stream.StreamID
WHERE Capacity > 20;
```

```
SELECT Employer, Location, Capacity
FROM Student
NATURAL JOIN StudentStream
NATURAL JOIN Stream
WHERE Capacity > 20;
```

You try it…

What is the phone number of the instructor who teaches a course that costs over 10000$ attended by studentID 202.



SQL Video

- A very good overview:

https://www.youtube.com/watch?v=uRdIdd-UkTc&index=7&list=PLdQddgMBv5zHcEN9RrhADq3CBColhY2hl

(It runs for 90+ minutes so for later viewing)

What's Examinable

- You need to know how to write SQL
  - DDL
  - DML

Next week

- Transactions and Concurrency
  - Concurrency
  - CRUD anomolies
  - Locking
  - Deadlocks
- Storage and indexing
  - Learn how data is stored and accessed within a DBMS
  - Alternative types of indexes

11. SQL Part 3

Query Nesting, Comparison Operators,
EXISTS, NOT EXISTS, VIEWS