

# Relational Databases - Comp2400 / Comp6240

## Lecture 8: Updates and Constraints

### Changing the database, keeping it valid

- Update operations and the constraints they can violate [E&N §5.3]
- Example student database
- UML for this
- SQL examples of constraint violation (see studentEg\*.sql files)

See the Postgress documentation Chapter 5  
(its available on the Labs page)

## Feedback Summary

Thanks for writing down your feedback last week.  
Here is a summary:

	<b>yes</b>	<b>no</b>
overview helped	66	11
know what to expect	52	21
oversimplified	15	60
understood lecture 5	65	9

96 responses in total

## Feedback Summary

- 25 people want more practical exercises, examples and hands-on stuff

## Feedback Summary

- 25 people want more practical exercises, examples and hands-on stuff
- 13 people expect the labs to help, and/or want their databases working

## Feedback Summary

- 25 people want more practical exercises, examples and hands-on stuff
- 13 people expect the labs to help, and/or want their databases working
- 6 people want the slides well in advance of the lectures

## Feedback Summary

- 25 people want more practical exercises, examples and hands-on stuff
- 13 people expect the labs to help, and/or want their databases working
- 6 people want the slides well in advance of the lectures
- 6 people want exam-like example questions, past papers, revision lectures

## Feedback Summary

- 25 people want more practical exercises, examples and hands-on stuff
- 13 people expect the labs to help, and/or want their databases working
- 6 people want the slides well in advance of the lectures
- 6 people want exam-like example questions, past papers, revision lectures
- 5 people want set readings or more references to the text

## Feedback Summary

- 25 people want more practical exercises, examples and hands-on stuff
- 13 people expect the labs to help, and/or want their databases working
- 6 people want the slides well in advance of the lectures
- 6 people want exam-like example questions, past papers, revision lectures
- 5 people want set readings or more references to the text
- 2 people do nice origami (especially the swan)

## Feedback Summary

- 25 people want more practical exercises, examples and hands-on stuff
- 13 people expect the labs to help, and/or want their databases working
- 6 people want the slides well in advance of the lectures
- 6 people want exam-like example questions, past papers, revision lectures
- 5 people want set readings or more references to the text
- 2 people do nice origami (especially the swan)
- 1 person says I suck

## Feedback Summary

- 25 people want more practical exercises, examples and hands-on stuff
- 13 people expect the labs to help, and/or want their databases working
- 6 people want the slides well in advance of the lectures
- 6 people want exam-like example questions, past papers, revision lectures
- 5 people want set readings or more references to the text
- 2 people do nice origami (especially the swan)
- 1 person says I suck
- and 1 person says I look like this guy



# Update Operations

[E&N §5.3]

A database state represents the state of the world.

# Update Operations

[E&N §5.3]

A database state represents the state of the world.

The world changes, and so must our databases. Thus we have update operations.

`insert` a tuple into a relation

# Update Operations

[E&N §5.3]

A database state represents the state of the world.

The world changes, and so must our databases. Thus we have update operations.

**insert** a tuple into a relation

**delete** a tuple from a relation

# Update Operations

[E&N §5.3]

A database state represents the state of the world.

The world changes, and so must our databases. Thus we have update operations.

**insert** a tuple into a relation

**delete** a tuple from a relation

**update** the value of an attribute of a tuple in a relation

## SQL for Update Operations

Read some SQL reference such as Chapter 5 of the Postgres documentation.

```
INSERT INTO student  
(name, studentNumber, class, major)  
VALUES ('MacLane', 200001, 1930, 'maths');
```

## SQL for Update Operations

Read some SQL reference such as Chapter 5 of the Postgres documentation.

```
INSERT INTO student
(name, studentNumber, class, major)
VALUES ('MacLane', 200001, 1930, 'maths');
```

```
DELETE FROM course WHERE courseNumber=1;
```

## SQL for Update Operations

Read some SQL reference such as Chapter 5 of the Postgres documentation.

```
INSERT INTO student
(name, studentNumber, class, major)
VALUES ('MacLane', 200001, 1930, 'maths');
```

```
DELETE FROM course WHERE courseNumber=1;
```

```
UPDATE section
SET courseNumber=2
WHERE courseName='Sheaf Theory';
```

## What could go wrong?

Under what circumstances can these operations take us from a valid database state to an invalid one?

constraints	update operation		
	insert	delete	update
<b>domain</b>			
<b>entity integrity</b>			
<b>primary key integrity</b>			
<b>referential integrity</b>			

## Some Examples

Now some examples that (attempt to) violate these kinds of constraints. This gives us a chance to

- look at a new example database

## Some Examples

Now some examples that (attempt to) violate these kinds of constraints. This gives us a chance to

- look at a new example database
- see some more SQL

## Some Examples

Now some examples that (attempt to) violate these kinds of constraints. This gives us a chance to

- look at a new example database
- see some more SQL
- have a practical look at database integrity

## Some Examples

Now some examples that (attempt to) violate these kinds of constraints. This gives us a chance to

- look at a new example database
- see some more SQL
- have a practical look at database integrity

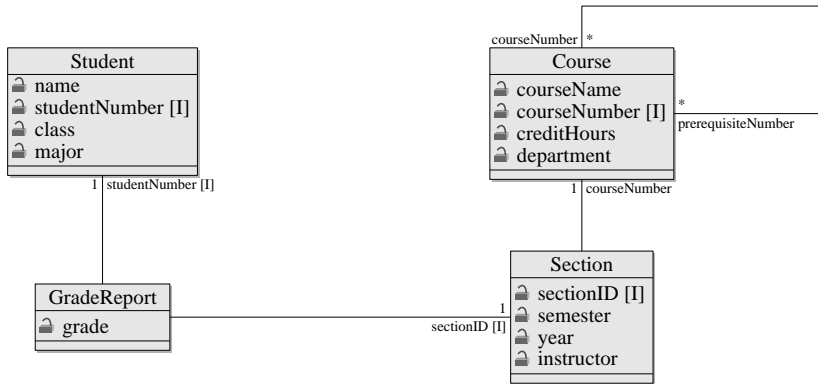
## Some Examples

Now some examples that (attempt to) violate these kinds of constraints. This gives us a chance to

- look at a new example database
- see some more SQL
- have a practical look at database integrity

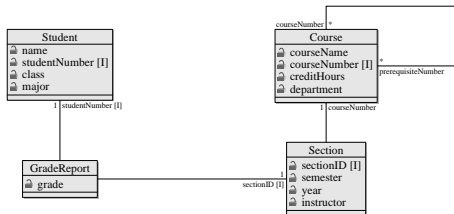
The following slide is a UML class diagram showing the database subject.

# Example Database



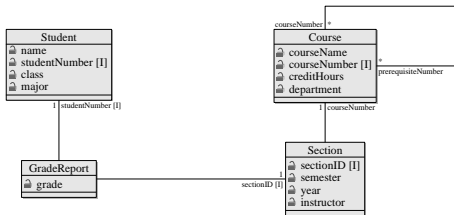
(database schema adapted from Elmasri and Navathe, 2007)

# Remarks on this (ab)use of UML



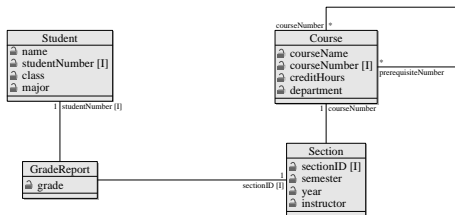
- this is UML from schema

# Remarks on this (ab)use of UML



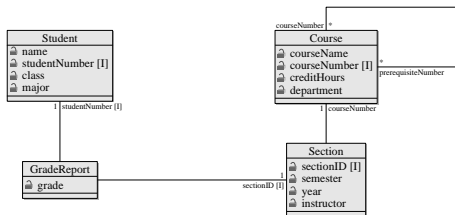
- this is UML from schema
- not schema from UML

# Remarks on this (ab)use of UML



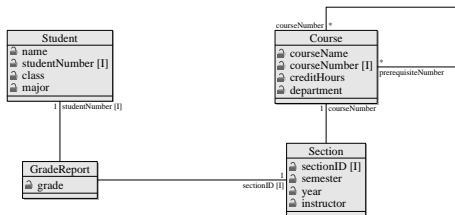
- this is UML from schema
- not schema from UML
- primary key marked [I] (following Comp3110)

# Remarks on this (ab)use of UML



- this is UML from schema
- not schema from UML
- primary key marked [I] (following Comp3110)
- foreign keys are far association ends with multiplicity 1 (problem?)

# Remarks on this (ab)use of UML



- this is UML from schema
- not schema from UML
- primary key marked [I] (following Comp3110)
- foreign keys are far association ends with multiplicity 1 (problem?)
- the reflexive association on Course is a relation

# INSERT

Insert can break all the 4 kinds of constraint we have considered

**domain** the tuple we insert may have an attribute value which violates its domain constraint

# INSERT

Insert can break all the 4 kinds of constraint we have considered

**domain** the tuple we insert may have an attribute value which violates its domain constraint

**key uniqueness** the tuple we insert may have the same primary key values as a tuple already in the relation

# INSERT

Insert can break all the 4 kinds of constraint we have considered

**domain** the tuple we insert may have an attribute value which violates its domain constraint

**key uniqueness** the tuple we insert may have the same primary key values as a tuple already in the relation

**entity integrity** the tuple we insert might have a null value for a primary key attribute

# INSERT

Insert can break all the 4 kinds of constraint we have considered

**domain** the tuple we insert may have an attribute value which violates its domain constraint

**key uniqueness** the tuple we insert may have the same primary key values as a tuple already in the relation

**entity integrity** the tuple we insert might have a null value for a primary key attribute

**referential integrity** the tuple we insert might have values for attributes of a foreign key, where no tuple in the referenced relation has those values for its primary key

# DELETE

Deleting tuples can not violate domain, key uniqueness nor entity integrity constraints.

# DELETE

Deleting tuples can not violate domain, key uniqueness nor entity integrity constraints.

But if another tuple points to this one, deleting it will violate **referential integrity**.

<b>Name</b>	<b>uid</b>	<b>DeptNum</b>
Greg	3996194	11

<b>DeptNum</b>	<b>DeptName</b>
11	Computer Science
42	Philosophy

# DELETE

Deleting tuples can not violate domain, key uniqueness nor entity integrity constraints.

But if another tuple points to this one, deleting it will violate **referential integrity**.

Name	uid	DeptNum
Greg	3996194	11

DeptNum	DeptName
42	Philosophy

Who does Greg work for now?

# UPDATE

The potential impact of an UPDATE depends on which attributes we are updating.

# UPDATE

The potential impact of an UPDATE depends on which attributes we are updating.

- if the attribute has domain constraints, they can be violated

# UPDATE

The potential impact of an UPDATE depends on which attributes we are updating.

- if the attribute has domain constraints, they can be violated
- if attribute is in the primary key

# UPDATE

The potential impact of an UPDATE depends on which attributes we are updating.

- if the attribute has domain constraints, they can be violated
- if attribute is in the primary key
  - key uniqueness violated if we set it to equal primary key of another tuple in the relation

# UPDATE

The potential impact of an UPDATE depends on which attributes we are updating.

- if the attribute has domain constraints, they can be violated
- if attribute is in the primary key
  - key uniqueness violated if we set it to equal primary key of another tuple in the relation
  - entity integrity violated if we set it to NULL

# UPDATE

The potential impact of an UPDATE depends on which attributes we are updating.

- if the attribute has domain constraints, they can be violated
- if attribute is in the primary key

**key uniqueness** violated if we set it to equal primary key  
of another tuple in the relation

**entity integrity** violated if we set it to NULL

**referential integrity** violated if something was pointing to  
the old primary key values

# UPDATE

The potential impact of an UPDATE depends on which attributes we are updating.

- if the attribute has domain constraints, they can be violated
- if attribute is in the primary key
  - key uniqueness violated if we set it to equal primary key of another tuple in the relation
  - entity integrity violated if we set it to NULL
  - referential integrity violated if something was pointing to the old primary key values
- if the attribute is in a foreign key

# UPDATE

The potential impact of an UPDATE depends on which attributes we are updating.

- if the attribute has domain constraints, they can be violated
- if attribute is in the primary key
  - key uniqueness** violated if we set it to equal primary key of another tuple in the relation
  - entity integrity** violated if we set it to NULL
  - referential integrity** violated if something was pointing to the old primary key values
- if the attribute is in a foreign key
  - referential integrity** violated if we set it to point to a non-existent tuple