Lecture 10

Database Construction (and Usage)

More on Modifications and Table Creation Assertions Triggers

Summary – Modifications

- Modifying the contents of a database:
 - Insertions INSERT INTO tablename VALUES tuple
 - Deletions

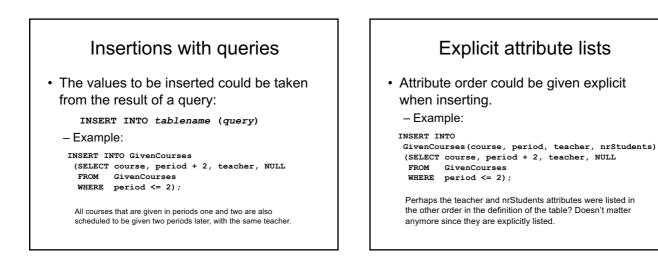
```
DELETE FROM tablename WHERE test over rows
```

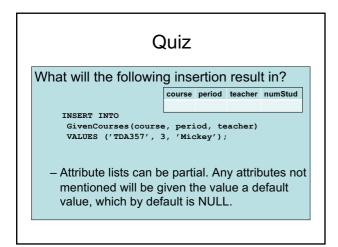
```
- Updates

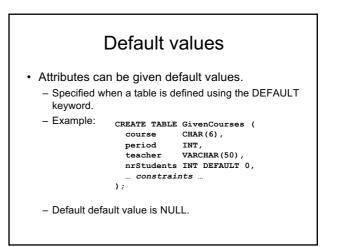
UPDATE tablename

SET attribute = value

WHERE test over rows
```







Insertion with default values

• Leaving out an attribute in an insertion with explicitly named attributes gives that row the default value for that attribute:

GivenCourses(course, period, teacher) VALUES ('TDA357', 3, 'Mickey');

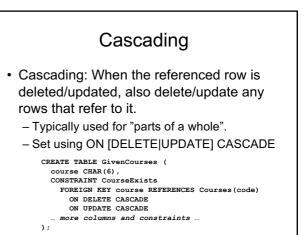
 When no attribute list is given, the same effect can be achieved using the DEFAULT keyword:

INSERT INTO GivenCourses
VALUES ('TDA357', 3, 'Mickey', DEFAULT);

			GivenCourses			
Courses		course	per	teacher	nrSt	
<u>code</u>	name		TDA357	2	Mickey	130
TDA357	Databases		TDA357	4	Tweety	95
TIN090	Algorithms		TIN090	1	Pluto	62
DELI	ETE FRO	M Co	ourses 'TDA35	-,		

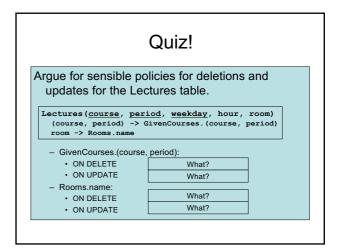
Policies for updates and deletions

- Rejecting a deletion or update in the presence of a reference isn't always the best option.
- SQL provides two other methods to resolve the problem: Cascading or Set NULL.
 - Default is RESTRICT: reject the deletion/update.

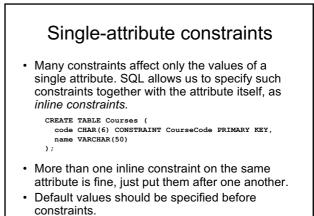


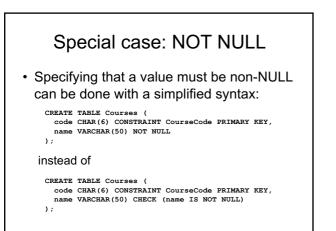
Set NULL

- Set NULL: When the referenced row is deleted/updated, set the corresponding attribute in any referencing rows to NULL.
 - Typically used when there is a connection, but one that does not affect the actual existence of the referencing row.
 - Set using ON [DELETE|UPDATE] SET NULL CREATE TABLE GivenCourses (teacher VARCHAR(50), CONSTRAINT TeacherExists
 - FOREIGN KEY teacher REFERENCES Teachers(name) ON DELETE SET NULL ON UPDATE CASCADE
 - ... more columns and constraints ...
);



Quiz!					
rgue for sensible policies for deletions and updates for the Lectures table.					
	<pre>eriod, weekday, hour, room) > GivenCourses.(course, period)</pre>				
room -> Rooms.name					
- GivenCourses.(cour	rse, period):				
	rse, period): CASCADE				
- GivenCourses.(cour					
 GivenCourses.(cour ON DELETE 	CASCADE				
 GivenCourses.(cour ON DELETE ON UPDATE 	CASCADE				





Special case: REFERENCES

- When a foreign key constraint is defined inline, the FOREIGN KEY keywords can be left out.
- An attribute that references another attribute could be seen as holding copies of that other attribute. Why specify the type again?

CREATE TABLE GivenCourses (
 course REFERENCES Courses(code),
 ... more columns and constraints ...
);

- The type can be left out even if the foreign key constraint is specified separately.

Quiz! It might be tempting to write

course REFERENCES Courses(code) PRIMARY KEY, period INT CHECK (period IN (1,2,3,4)) PRIMARY KEY, ... more columns and constraints ...);

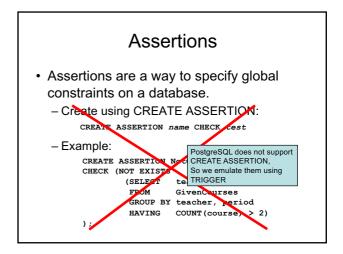
Why will this not work?

An inline constraint only constrains the current attribute. What the above tries to achieve is to declare two separate primary keys, which is not allowed in a table.

Constraints

- We have different kinds of constraints:
 Dependency constraints (X → A)
 - Table structure, PRIMARY KEY, UNIQUE
 - Referential constraints
 FOREIGN KEY ... REFERENCES
 - Value constraints
 CHECK
 - Miscellaneous constraints (like multiplicity)
 - E.g. no teacher may hold more than 2 courses at the same time.
 - · How do we handle these?

Quiz!	
"No teacher may hold more than two courses in the same period!"	
How can we formulate this constraint SQL?	in
NOT EXISTS (SELECT teacher, period FROM GivenCourses GROUP BY teacher, period HAVING COUNT(course) > 2);	
course period teacher	numStud

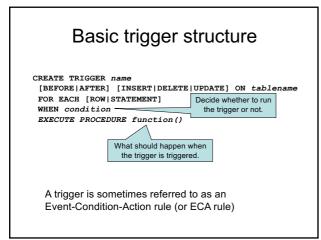


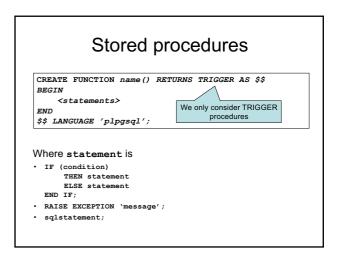
Triggers

- When something wants to change the database in some way, trigger another action as well or instead.
 - Example (silly): Whenever a new course is inserted in Courses, schedule that course to be given in period 1, with NULL for the teacher and nrStudents fields.
 - Example: Whenever a lecture is scheduled to take place at 8:00, schedule the lecture to 10:00 instead.

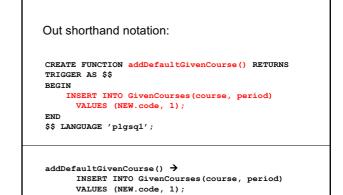
Assertions as triggers

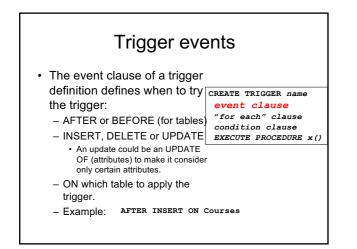
- "Instead" could mean to do nothing, i.e. reject the update, which means we can use triggers to simulate assertions.
 - Still costly, but puts the burden on the user to specify when the conditions should be checked (hand optimization).
 - Example: Whenever a teacher is scheduled to hold a course in a period where he or she already holds two courses, reject the insertion.





Example trigger:	
CREATE FUNCTION addDef TRIGGER AS \$\$	FaultGivenCourse() RETURNS
BEGIN	
INSERT INTO Given VALUES (NEW.code	Courses(course, period) a, 1);
END	
\$\$ LANGUAGE 'plgsql';	'NEW' refers to the newly inserted tuple
CREATE TRIGGER Default	Scheduling
AFTER INSERT ON Cours	ses

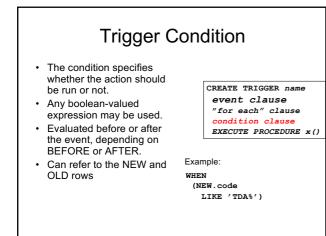


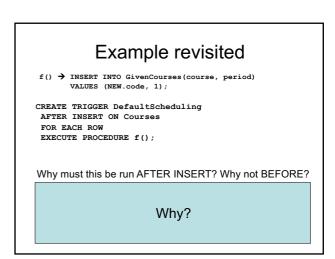


FOR EACH ROW

- A single insert, update or deletion statement could affect more than one row.
- If FOR EACH ROW is specified, the trigger is run once for each row affected, otherwise once for each statement.
- Default is FOR EACH STATEMENT, which could also be stated explicitly.

CREATE TRIGGER name event clause "for each" clause condition clause EXECUTE PROCEDURE x()





Example revisited

f() → INSERT INTO GivenCourses(course, period) VALUES (NEW.code, 1);

CREATE TRIGGER DefaultScheduling AFTER INSERT ON Courses FOR EACH ROW EXECUTE PROCEDURE f();

Why must this be run AFTER INSERT? Why not BEFORE?

Because there is a foreign key constraint from GivenCourses to Courses, and until we have inserted the row into Courses, there would be nothing for the new row in GivenCourses to refer to.

Recap on views

- Views are persistent named queries they can be referred to just as if they were tables, but their data is contained in other (base) tables.
- Also referred to as virtual tables.

```
CREATE VIEW DBLectures AS
SELECT room, hour, weekday
FROM Lectures
WHERE course = 'TDA357'
AND period = 3;
```

Updating views

- Views contain no data of their own, and so cannot normally be updated.
- But views can be queried without containing any data of their own. The trick is to translate the query on the view into what it really means, i.e. the view definition.
- · Why not do the same for modifications?

Triggers on views

- We can define what modifications on views mean using triggers.
- Special form of event for views only: INSTEAD OF.

```
f() → INSERT INTO Lectures
VALUES ('TDA357', 2, NEW.weekday,
NEW.hour, NEW.room);
```

CREATE TRIGGER DBLectureInsert INSTEAD OF INSERT ON DBLectures FOR EACH ROW EXECUTE PROCEDURE f()

Summary – Triggers

- Triggers specify extra actions to take on certain events.
 - Event: BEFORE or AFTER a modification
 - Condition: test if we should run the trigger
 - Action: The stuff to be done.
 - SET to change values in the rows being modified.
- Triggers can be defined on views
 Event: INSTEAD OF

Next time, Lecture 11