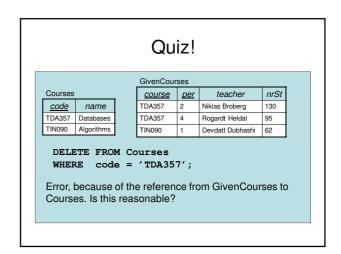
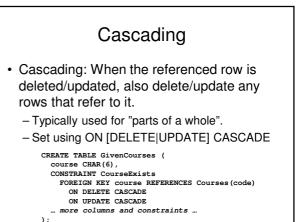
Database Construction (and Usage)

More on Modifications and Table Creation Assertions Triggers



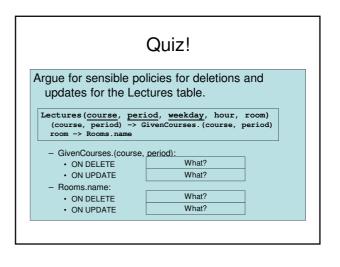
Policies for updates and deletions

- 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 to reject the deletion/update.





- 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 DELETE SET NULI ON UPDATE CASCADE
 - ... more columns and constraints ...

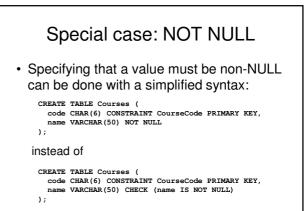


Single-attribute constraints

· Many constraints affect only the values of a single attribute. SQL allows us to specify such constraints together with the attribute itself, as inline constraints.

CREATE TABLE Courses (code CHAR(6) CONSTRAINT CourseCode PRIMARY KEY, name VARCHAR(50));

- · More than one inline constraint on the same attribute is fine, just put them after one another.
- · Default values should be specified before constraints.



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

CREATE TABLE GivenCourses (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 \rightarrow 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 at the same time!"

How can we formulate this constraint in SQL?

> NOT EXISTS (SELECT FROM GivenCourses GROUP BY teacher, period HAVING count (course) > 2

);

Assertions • Assertions are a way to specify global constraints on a database. - Create using CREATE ASSERTION: CREATE ASSERTION name CHECK test - Example: CREATE ASSERTION NotOverworked CHECK (NOT EXISTS (SELECT * FROM GivenCourses GROUP BY teacher, period HAVING count (course) > 2));

Assertions vs Checks

- Assertions are global they are guaranteed to hold throughout any modifications.
- Checks on attributes in a table are only checked when that table is updated. If the check involves another table in a subquery, modifications on that table will not be checked.
- Inline checks on attributes are only checked when that attribute is modified. If the check involves references to other attributes in the same table through a subquery, modifications on those attributes will not be checked.

Assertion Properties

- Assertions are static constraints on a database that, like all constraints, are guaranteed always to be true.
- Any modification that would violate an assertion is rejected.
- Assertions potentially need to be checked on every modification. This is very costly!
- · And what does Oracle do with inefficient things?
 - Disallows them!



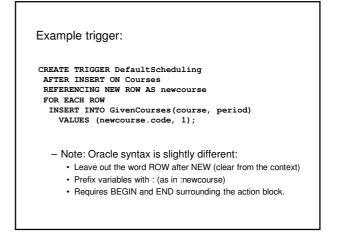
Triggers

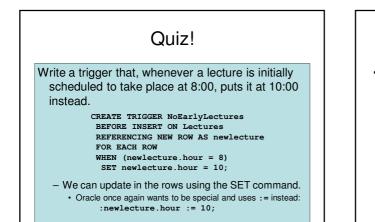
- When something wants to change the database in some way, trigger another action as well or instead.
 - Example (silly): Whenever a new course is inserted in Courses, schedule that course to be given in period 1, with NULL for the teacher and nrStudents fields.
 - Example: Whenever a lecture is scheduled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

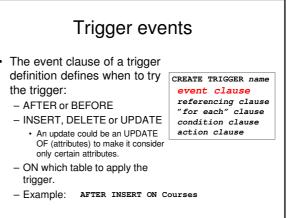
Assertions as triggers

- "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.

Basic trigger structure
CREATE TRIGGER name [BEFORE AFTER] [INSERT DELETE UPDATE] ON tablename REFERENCING [NEW OLD] [ROW TABLE] AS variable FOR EACH [ROW STATEMENT] WHEN condition action to perform Underform What should happen when the trigger is triggered.
A trigger is sometimes referred to as an Event-Condition-Action rule (or ECA rule)



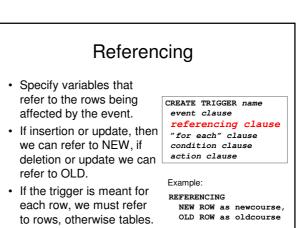




FOR EACH ROW

- 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, otherwise once for each statement.
- Default is FOR EACH STATEMENT, which could also be stated explicitly.

CREATE TRIGGER name event clause referencing clause "for each" clause condition clause action clause



Trigger Condition

- The condition specifies whether the action should be run or not.
- Any boolean-valued
 expression may be used.
- Evaluated before or after the event, depending on BEFORE or AFTER.
- Can refer to the new and old rows using the variables defined in the referencing clause.

CREATE TRIGGER name event clause referencing clause "for each" clause condition clause action clause

Example: WHEN (newcourse.code LIKE 'TDA%')

Action block

- Specifies what statements to run when the trigger is executed.
- Multiple statements must be enclosed using BEGIN and END.
- Can refer to row (or table) variables, and change in them (only rows) using the SET statement.

CREATE TRIGGER name event clause referencing clause "for each" clause condition clause action clause

Example: INSERT INTO

GivenCourses (course, period) VALUES (newcourse.code, 1):

Example revisited

CREATE TRIGGER DefaultScheduling AFTER INSERT ON Courses REFERENCING NEW ROW AS newcourse FOR EACH ROW INSERT INTO GivenCourses(course, period) VALUES (newcourse.code, 1);

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.

Triggers vs. Oracle

 Triggers are costly if the condition involves joining tables together to find some data. What does Oracle do with inefficient stuff?

– Disallows it!

- Oracle only allows conditions over the table on which the trigger is defined.
- A trigger on a table may not change that same table in the action block.
- · Lots of special syntax.
- Oracle has its own built-in programming language PL/SQL in which action blocks can be specified.
- Rejecting a statement can be done in Oracle using the RAISE_USER_ERROR function, check the Oracle reference for details.

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 = 2;

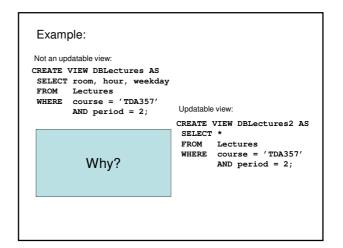
Updating views

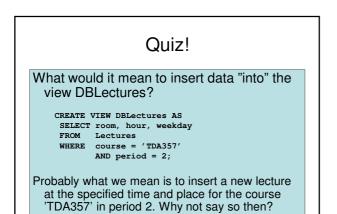
- Views contain no data of their own, and so cannot normally be updated.
- But views can be queried without containing any data of their own. The trick is to translate the query on the view into what it really means, i.e. the view definition.
- · Why not do the same for modifications?

Updatable views

- If the view is built by selecting from a single relation R such that
 - no references to R in subqueries in the WHERE clause
 - the attribute list of the view contains enough attributes of R so that the inserted tuple will show up in the view again even if we pad the rest with NULL.

then the view is *updatable*, and any updates on it can be pushed back into R.





Triggers on views

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

CREATE TRIGGER DBLectureInsert INSTEAD OF INSERT ON DBLectures REFERENCING NEW ROW AS newl FOR EACH ROW INSERT INTO Lectures VALUES ('TDA357', 2, newl.weekday, newl.hour, newl.room);

Quiz: Fooling Oracle!

- When writing the condition for a trigger, Oracle will not allow us to look in tables other than the one that the trigger is defined on.
- How can we get around this?
 - We create a view with all the information we need, and then write our trigger on the view. We can then look at all the needed information, and translate the modifications to be done on the base table.

How to really do it, the Oracle way

- Use PL/SQL statements
 IF-THEN-ELSE-END IF
 - Condition in the IF may look at any tables, unlike the WHEN
 - Note that there is a difference:
 - If the WHEN clause fails, the trigger will not be run.
 - The IF test is part of the action block, so if we reach it the trigger is already running, and will run to the end unless aborted.

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

Course Objectives - Construction

When the course is through, you should

- Given a database schema with related constraints, implement the database in a relational (SQL) DBMS
 - Courses(<u>code</u>, name, dept, examiner) Rooms(<u>roomNr</u>, name, building) Lectures(<u>roomNr</u>, <u>day</u>, hour, course) roomNr -> Rooms.roomNr course -> Courses.code

Exam – SQL DDL

- "A grocery store wants a database to store information about products and suppliers. After studying their domain you have come up with the following database schema. ..."
- Write SQL statements that create the relations with constraints as tables in a DBMS.
- Write a trigger that, whenever the quantity in store of an item drops below a given threshhold, inserts an order for that product with the supplier.

Next Lecture Database Applications: SQL/PSM Embedded SQL JDBC