Lecture 7

Database Construction and Usage

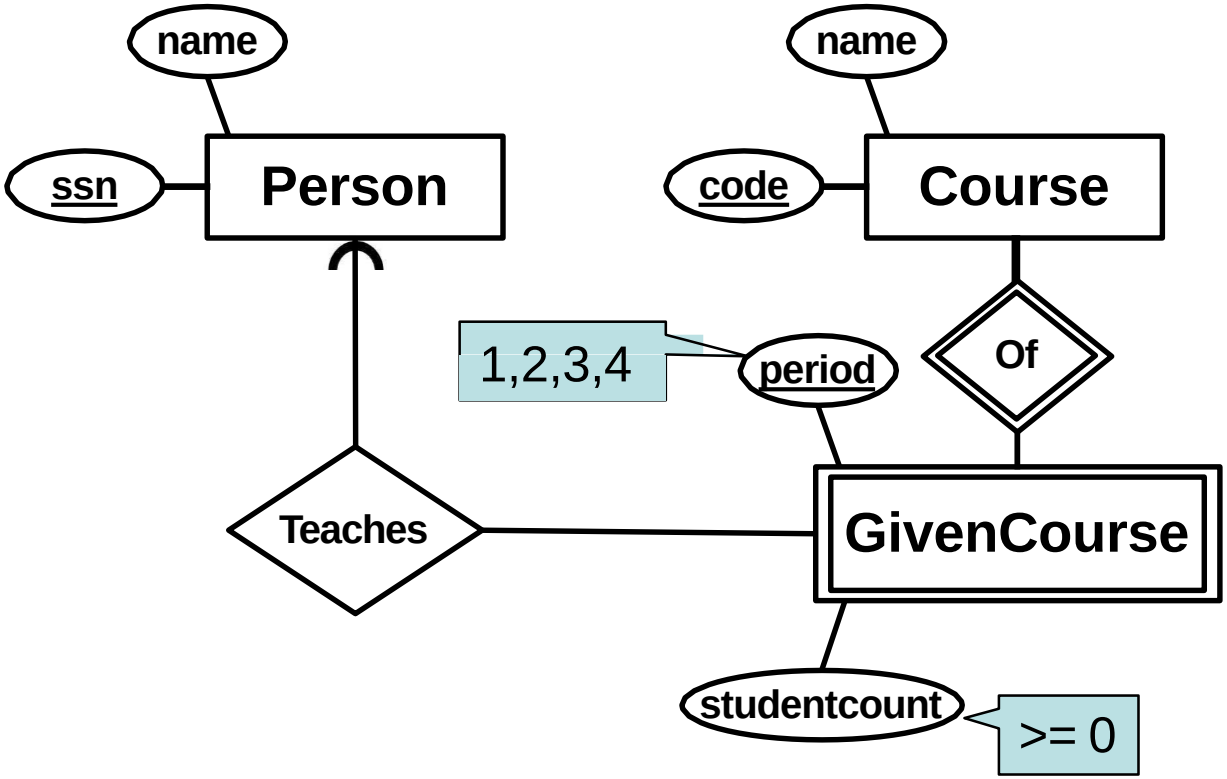
SQL DDL and DML
Relational Algebra

Case convention

- SQL is completely case insensitive. Upper-case or Lower-case makes no difference. We will use case in the following way:
  - **`UPPERCASE`** marks keywords of the SQL language.
  - **`lowercase`** marks the name of an attribute.
  - **`Capitalized`** marks the name of a table.

SQL Data Definition Language

Working example



Person(ssn, name)

Course(code, name)

GivenCourse(code, period, studentcount, teacher)

code -> Course.code

teacher -> Person.ssn

Creating and dropping tables

- Relations become tables, attributes become columns.

```
CREATE TABLE Tablename (
  <list of table elements>
);
```

- Get all info about a created table:

```
\d+ Tablename;
```

PostgreSQL specific!

- Remove a created table:

```
DROP TABLE Tablename;
```

Table declaration elements

- The basic elements are pairs consisting of a column name and a type.
- Most common SQL types:
  - INT or INTEGER (synonyms)
  - REAL or FLOAT (synonyms)
  - CHAR(*n*) = fixed-size string of size *n*.
  - VARCHAR(n) = variable-size string of up to size *n*.
  - TEXT = string of unrestricted length

Example

Example:

```
CREATE TABLE Courses (
   code CHAR(6),
   name TEXT NOT NULL
);
```

NULL is allowed by default!

Created the table courses:

code	name

Declaring keys

- An attribute or a list of attributes can be declared PRIMARY KEY or UNIQUE
  - PRIMARY KEY: (At most) One per table, never NULL. Efficient lookups in all DBMS.
  - UNIQUE: Any number per table, can be NULL. Could give efficient lookups (may vary in different DBMS).
- Both declarations state that all other attributes of the table are functionally determined by the given attribute(s).

Example

```
CREATE TABLE Courses(
   code CHAR(6),
   name TEXT NOT NULL,
   PRIMARY KEY (code)
);
```

Foreign keys

- Referential constraints are handled with references, called *foreign keys*.
  - FOREIGN KEY *attribute*
    REFERENCES *table(attribute)*.

```
FOREIGN KEY course
  REFERENCES Courses(code)
```

Foreign keys

- **General:**

```
FOREIGN KEY course REFERENCES Courses(code)
```

- **If course is Primary Key in Courses:**

```
FOREIGN KEY course
    REFERENCES Courses
```

- **Give a name to the foreign key:**

```
CONSTRAINT ExistsCourse
FOREIGN KEY course
REFERENCES Courses
```

Example

```
CREATE TABLE GivenCourses (
    course          CHAR(6),
    period          INT,
    numStudents     INT,
    teacher         INT REFERENCES People(ssn) NOT
    NULL,
    PRIMARY KEY (course, period),
    FOREIGN KEY (course) REFERENCES Courses(code)
);
```

Example

```
CREATE TABLE GivenCourses (
    course CHAR(6) REFERENCES Courses,
    period              INT,
    numStudents         INT,
    teacher             INT REFERENCES People(ssn) NOT
    NULL,
    PRIMARY KEY (course, period)
);
```

Value constraints

- Use CHECK to insert simple value constraints.
  - CHECK (*some test on attributes*)

```
CHECK (period IN (1,2,3,4))
```

Example

```
CREATE TABLE GivenCourses (
    course CHAR(6) REFERENCES Courses,
    period          INT CHECK (period IN (1,2,3,4)),
    numStudents     INT,
    teacher         INT REFERENCES People(ssn) NOT
    NULL,
    PRIMARY KEY (course, period)
);
```

Example

```
CREATE TABLE GivenCourses (
    course CHAR(6) REFERENCES Courses,
    period              INT,
    numStudents         INT,
    teacher             INT REFERENCES People(ssn) NOT
    NULL,
    PRIMARY KEY (course, period),
    CONSTRAINT ValidPeriod CHECK (period in (1,2,3,4))
);
```

SQL Data Manipulation Language: Modifications

Inserting data

```
INSERT INTO tablename
  VALUES (values for attributes);

INSERT INTO Courses
  VALUES ('TDA357', 'Databases');
```

code	*name*
TDA357	Databases

Example

- **Legal:**
  - **INSERT INTO GivenCourses VALUES ('TDA357',2,199,1);**
- **Not Legal:**
  - **INSERT INTO GivenCourses VALUES ('TDA357',7,199,1);**
- **ERROR: new row for relation "givencourses" violates check constraint "givencourses_period_check"DETAIL: Failing row contains (TDA357, 7, 199, 1).**

Deletions

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

```
DELETE FROM Courses
  WHERE code = 'TDA357';
```

Updates

```
UPDATE  tablename
SET     attribute = ...
WHERE   test over rows

UPDATE  GivenCourses
SET     teacher = 'Graham Kemp'
WHERE   course = 'TDA357'
  AND   period = 2;
```