Engineering Databases

Lecture 4 – Foreign Key and Joins

November 9, 2022

M. Saeed Mafipour & Mansour Mehranfar

Contents of Lecture 3

- DML (partial):
  - update <t1> set <c1>=<v1> [,<c2>=<c2>,…] [where <condition>]
  - delete from <t1> [where <condition>]
  - insert into for multiple data sets
- Advanced ER-Mapping Schema
  - 1:N, N:1, and 1:1 relation can be eliminated
- The Relational Model by Codd
  - The formal groundings of SQL based on set theory and first-order predicate logic
  - Domains, attributes, relations and relational schemas
  - Projections: $\Pi_{columns}$ (Relation), extract columns from a relation
  - Union: $R_i \cup R_j$, merge identical schemas
  - Selection: $\sigma_{statement}$(Relation), select sub-sets of a relation based on formulas

Foreign key integrity constraint

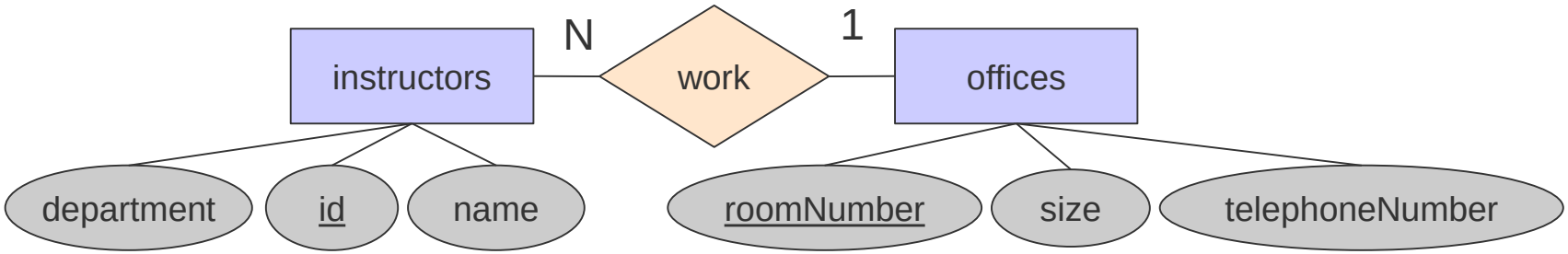
- An essential principle in relational design
- Needed to reflect real-word relations over multiple tables

- A foreign key is an attribute or attribute set
- It is identical to a primary key of another table
- The tables are linked via the foreign key constraint
- Is part of the DDL

- The DBMS ensures the data integrity in case of data changes

Foreign key integrity constraint

- Example:



instructors			
id	name	department	room
1	Bob	CMS	10
2	Tom	CIE	5
3	Alice	CMS	8

offices		
roomNumber	size	telephoneNumber
10	10	22678
8	25	35131

Foreign key integrity constraint

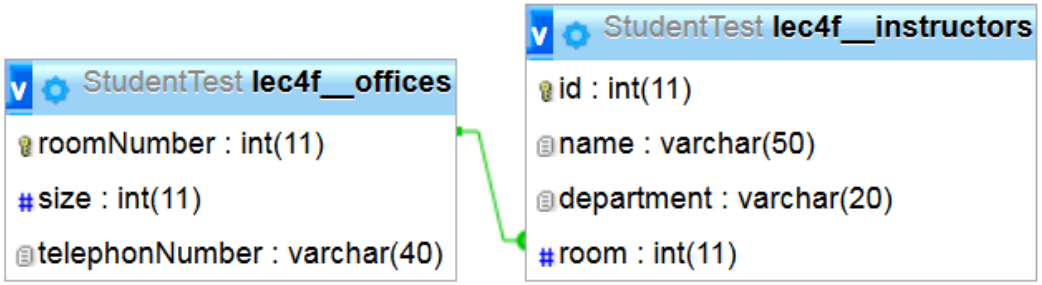
- Ensures that a value in a table is contained in some other table
- The syntax for **create table** with foreign key

```
CREATE TABLE <name of table> (
        [other attribute definitions],
        <name of foreign key> <type of foreign key> <integrity constraints>,
        [other attributes that are part of the foreign key, comma separated]
        FOREIGN KEY (<local foreign key attributes, comma separated>)
        REFERENCES <other table>
                (<other table primary key attributes, comma separated>));
```

- The foreign key attributes (including type) have to match the other table's columns.
- The name of the attributes do not have to match!
- The other table must exists!

https://dev.mysql.com/doc/refman/5.7/en/create-table-foreign-keys.html

Foreign key integrity constraint

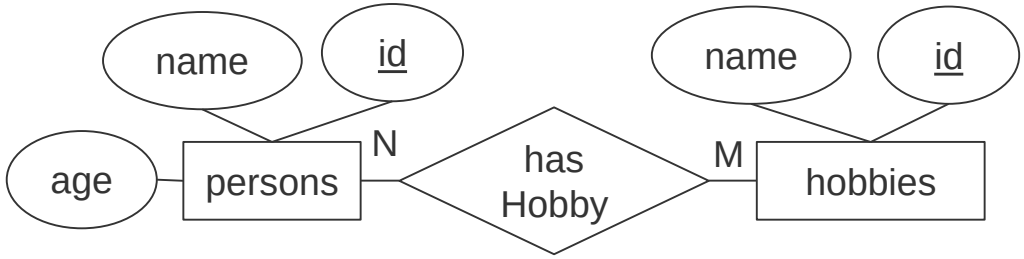


- Example:

```
CREATE TABLE offices (
    roomNumber INTEGER PRIMARY KEY,
    size INTEGER NOT NULL,
    telephonNumber VARCHAR(40) NOT NULL);

CREATE TABLE instructors (
    id INTEGER PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    department VARCHAR(20),
    room INTEGER,
    FOREIGN KEY (room) REFERENCES offices (roomNumber));
```

Foreign key integrity constraint exercise 1



- Create the tables persons and hobbies (moodle sql!)
- Create table hasHobby which includes two foreign key constrains but no primary key.
- Hints: It is not possible to have two attributes of the same name in a single table.

- This step is mandatory for later exercises, don’t skip it!

Foreign key integrity constraint

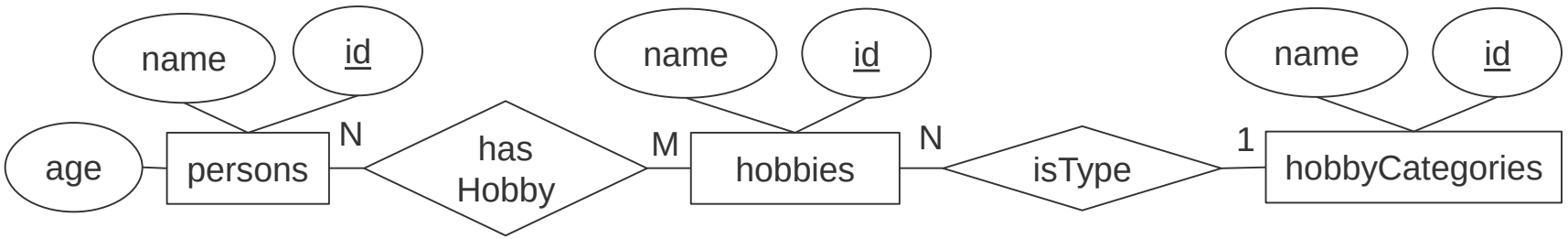
- Ensures that a value in a table is contained in some other table

- The syntax for **alter table** with foreign key

```
ALTER TABLE <name of table> ADD
        FOREIGN KEY (<local foreign key attributes, comma separated>)
        REFERENCES <other table>
                (<other table primary key attributes, comma separated>);
```

https://dev.mysql.com/doc/refman/5.7/en/create-table-foreign-keys.html

Foreign key integrity constraint exercise 2



- Create a new table hobbyCategories
- Alter table hobbies:
  - Add the attribute typeId (new ER-Mapping rule!)
    and the foreign key constraint on typeId for hoobyCategories id
  - Hint: Use a single alter table statement but multiple add statements
    that are comma separated!
- This step is mandatory for later exercises, don’t skip it!

Foreign key integrity constraint

- The foreign key ensures data integrity

instructors			
id	name	department	room
1	Bob	CMS	5
2	Tom	CIE	5
3	Alice	CMS	8

offices		
roomNumber	size	telephoneNumber
5	10	22678
8	25	35131

- Examples:

  delete from offices where roomNumber=5
  update instructors set room=10 where department='CMS'
  insert into instructors values (4, Mark, 'CMS', 10)

- In all cases, an error is given.

Foreign key integrity constraint

- Control updates and changes of other tables
- The syntax for **update and delete** actions in the foreign key definition
  FOREIGN KEY (<local foreign>)
  REFERENCES <other table> (<other table primary key attributes>)
  [ON DELETE [RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT]]
  [ON UPDATE [RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT]]
- Example:
  foreign key (room) references offices (roomNumber) on delete cascade
- Restrict is default, means not allowed
- Cascade means that the other table's updates are propagated
- Set Null will set the value to null (if no NOT NULL constraint is given)
- No action will change nothing
- Set default will set the value to a constraint defined default value

https://dev.mysql.com/doc/refman/5.7/en/create-table-foreign-keys.html

Foreign key integrity constraint

- The foreign key ensures data integrity

instructors			
id	name	department	room
1	Bob	CMS	5
2	Tom	CIE	5
3	Alice	CMS	8

offices		
roomNumber	size	telephoneNumber
5	10	22678
8	25	35131

- Example for delete from offices where roomNumber=5
  - Restrict: error
  - Cascade: Bob und Tom are deleted!
  - Set null: room in Bob and Tom are null (if no NOT NULL constrain is given)
  - No action: the room in Bob and Tom is 5
  - Set default: the room in Bob and Tom is set to a predefined default value

Relational Algebra

- Formal definitions of

  Cartesian product $\times$

  Renaming ρ

  θ-Join $\bowtie_\theta$ and equi-Join $\bowtie_{\theta\ is\ =}$

  (Natural) Join $\bowtie$

  Right $\ltimes$ and left semi join $\rtimes$

  Left $⟕$, right $⟖$, and full outer join $⟗$

Cartesian Product (Cross Product)

In general: R × S

- Contains | R | · | S | combinations (all) of tuples from R and S.
- The resulting relation R × S is the combination of their attributes

- The SQL syntax for the Cartesian product:

  <Relation 1> CROSS JOIN <Relation 2>

  or

  <Relation 1> JOIN <Relation 2>

- CROSS JOIN and JOIN are synonyms in MySQL (but not in other DBMS)

https://dev.mysql.com/doc/refman/5.5/en/join.html

Cartesian Product (Cross Product)

- Example:

  instructors × offices

  SELECT * FROM instructors CROSS JOIN offices

  SELECT * FROM instructors JOIN offices

instructors	
name	chair
Bob	CMS
Tom	CIE

×

offices	
room	size
005	14
008	25

=

instructors × offices			
name	chair	room	size
Bob	CMS	005	14
Tom	CIE	005	14
Bob	CMS	008	25
Tom	CIE	008	25

- Multiple cross products: persons × hasHobby × hobbies

Renaming

- Problem occurs when using the same relation twice.
- E.g. the instructors × instructors will fail

  ~~SELECT * FROM instructors JOIN instructors~~

In general: $\rho_{new\ name}$ (Relation)

- The SQL syntax for the renaming is:
  <Relation> AS <New Name>

- Example:

  $\rho_{inst1}(Instructors) \times \rho_{inst2}(Instructors)$

  SELECT * FROM instructors AS inst1 JOIN instructors AS inst2

- We already used the ρ operation to give unamend relation a name (nesting!)

θ-Join and Equi-Join

In general: $\mathrm{R} \bowtie_\theta S$ is a θ-Join. Also: $\mathrm{R} \bowtie_\theta S = \sigma_\theta(R \times S)$

- Not all rows of the Cartesian product are desired
- Result contains all entities of $R \times S$ for which formula θ is fulfilled
- In case the condition comprises only = it is an equi-join

- The SQL syntax for the Cartesian product:

  SELECT * FROM <relation1> JOIN < relation2 >
  ON <condition1> [AND|OR more Conditions]

  or

  SELECT * FROM <table1> INNER JOIN <table2>
  ON <condition1> [AND|OR more Conditions]

- INNER JOIN and JOIN are synonyms in MySQL (but not in other DBMS)

θ-Join

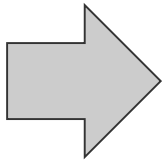
- Example: instructors $\bowtie_{instructors.name!=students.name}$ students
  SELECT * FROM instructors JOIN students ON instructors.name != students.name

$R \times S$

instructors	
name	chair
Bob	CMS
Tom	CIE

×

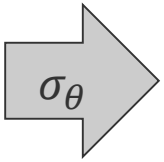
students	
name	major
Alice	Math
Bob	Sports



instructors × students			
name	chair	name	major
Bob	CMS	Alice	Math
Tom	CIE	Alice	Math
Bob	CMS	Bob	Sports
Tom	CIE	Bob	Sports

$\sigma_\theta(R \times S)$

instructors × students			
name	chair	name	major
Bob	CMS	Alice	Math
Tom	CIE	Alice	Math
Bob	CMS	Bob	Sports
Tom	CIE	Bob	Sports



σ_θ(instructors × students)			
name	chair	name	major
Bob	CMS	Alice	Math
Tom	CIE	Alice	Math
Tom	CIE	Bob	Sports

Equi-Join

- Example: instructors $\bowtie_{instructors.name=students.name}$ students

  SELECT * FROM instructors JOIN students ON instructors.name = students.name

  or

  SELECT * FROM instructors INNER JOIN students ON instructors.name = students.name

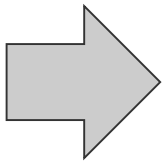
  or

  SELECT * FROM instructors CROSS JOIN students ON instructors.name = students.name

instructors	
name	chair
Bob	CMS
Tom	CIE

$\bowtie_\theta$

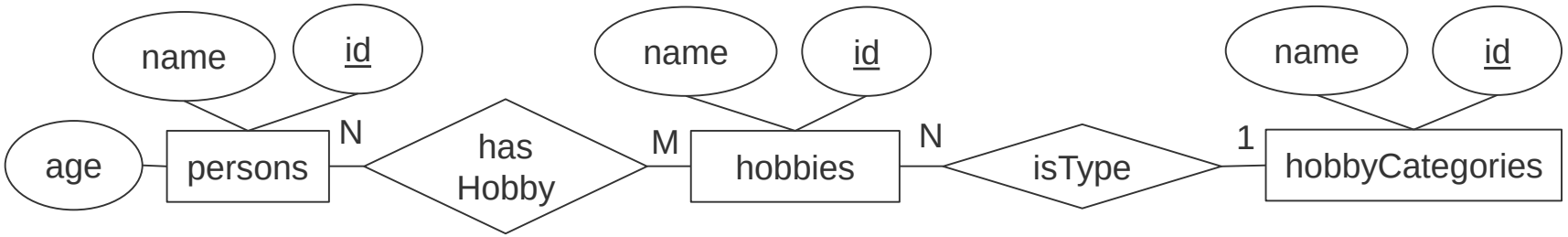
students	
name	major
Alice	Math
Bob	Sports



instructors$\bowtie_\theta$students		
name	chair	major
Bob	CMS	Sports

- Including renaming: select * from instructors as i join students s on i.name = s.name

Equi-Join exercise



- If not already done, create tables persons, hasHobby, hobbies, and hobbyCategories.
- Populate the tables with data (provided in the lecture)

- Execute the SQL commands:

  hobbies $\bowtie_{id\ =hobbyId}$ hasHobby

  ρ_h(hobbies) $\bowtie_{typeId=id}$hobbyCategories

  $\sigma_{age>30}$(persons$\bowtie_{id\ =personId}$ hasHobby)

  $\Pi_{age,name}$(persons $\bowtie_{id\ =personId}$ hasHobby)

  hobbies $\bowtie_{id\ =hobbyId}$ hasHobby$\bowtie_{personId\ =id}$persons

  ρ_h(hobbies) $\bowtie_{id\ =hobbyId}$ ρ_{hH}(hasHobby) $\bowtie_{personId\ =id}$ ρ_p(persons)

Natural Join

In general: $R \bowtie S$

- Same as equi-join with columns, having the same name, set equal.
- Is associative and communicative: $A \bowtie B \bowtie C = A \bowtie (B \bowtie C) = B \bowtie (A \bowtie C)$
- The SQL syntax for the natural join

  SELECT * FROM <relation1> NATURAL JOIN <relation2 >

- Example:

  instructors ⋈ students

  SELECT * FROM instructors NATURAL JOIN students

- Exercise:

  persons ⋈ hobbies (this joins on name and id!)

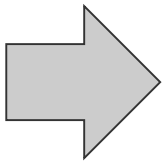
Semi Join

In general: left semi join $R \ltimes S = \Pi_R(R \bowtie S)$, right semi join: $R \rtimes S = \Pi_S(R \bowtie S)$

- Selects one table's columns that were matched during a natural join.
- The SQL syntax for the left semi join
  SELECT <r1>.* FROM <r1> NATURAL JOIN <r2>
- The SQL syntax for the right semi join
  SELECT <r2>.* FROM <r1> NATURAL JOIN <r2>
- Example:
  instructors ⋉ students (left semi join)
  instructors ⋊ students (right semi join)

instructors	
name	chair
Bob	CMS
Tom	CIE

students	
name	major
Alice	Math
Bob	Sports



instructors⋉ students	
name	chair
Bob	CMS

instructors⋊ students	
name	major
Bob	Sports

Outer Join

In general: left outer join ⟕, right outer join ⟖, and full outer join ⟗

- Compared to Equi-Join, outer joins keep the unmatched entries
- The entries are taking from the left (left outer join), the right (right outer join) or both (full outer join) relations.

- The SQL syntax for the outer joins:

  left outer join: SELECT * FROM <r1> LEFT JOIN <r2> ON <Condition>

  right outer join: SELECT * FROM <r1> RIGHT JOIN <r2> ON <Condition>

  full outer join: <left outer join> UNION [ALL] <right outer join>

- Adding ALL after UNION will keep identical rows

Outer Join

- Example: (on name)
  instructors ⟖ students (right outer join)
  instructors ⟕ students (left outer join)
  instructors ⟗ students (full outer join)

instructors	
name	chair
Bob	CMS
Tom	CIE

students	
name	major
Alice	Math
Bob	Sports

instructors⟖ students			
name	chair	name	major
Bob	CMS	Bob	Sports
NULL	NULL	Alice	Math

instructors⟕ students			
name	chair	name	major
Bob	CMS	Bob	Sports
Tom	CIE	NULL	NULL

Instructors ⟗ students			
name	chair	name	major
Bob	CMS	Bob	Sports
Tom	CIE	NULL	NULL
Bob	CMS	Bob	Sports
NULL	NULL	Alice	Math

Homework (solution is on moodle!)

- There are 4 relations: Students, Professors, Rooms, Classes
- Students have a name and a studentId.
- Professors have a name, faculty and a profId
- Rooms have a name, a number of seats, and a roomId.
- Classes have a name, a limit on the number of students, a room, a professor, and a classId.
  - Store **ONLY** the ids of the professors and rooms in this table. No names,…
  - Room and Professor are foreign keys, so only valid ids can be stored.

- Draw an entity relationship diagram.
- Create the tables
- Add 3 Students, 3 Professors, 3 Rooms and 3 Classes
- Output all professors who teach a course.
- Output all courses taught by one of the professors you entered.
- Output all courses with room names and professor names.
- Output one list containing the names of all professors and students

Also write all queries in relational algebra notation

End of Lecture

Thank you for your attention