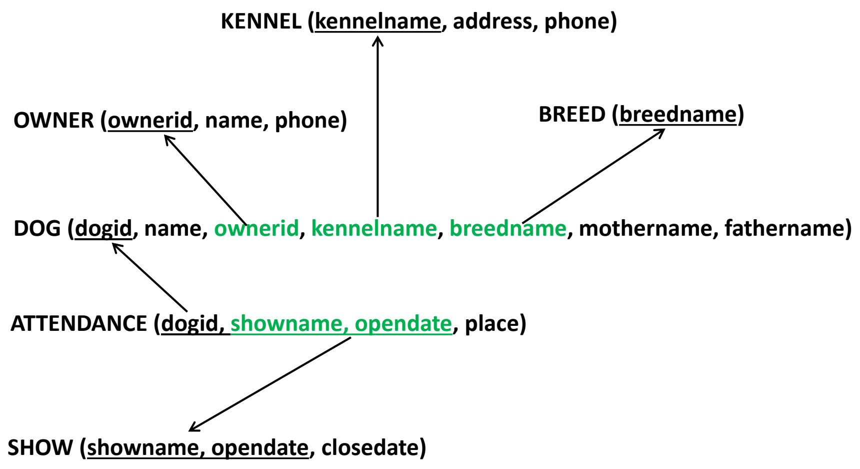
University of Glasgow
School of Computing Science

DB(H) Database Systems

Lab 1 Tasks: Solutions

Task 1: Mapping to Relational Schema

A proposed Relational Schema could be:


KENNEL (kennelname, address, phone)
OWNER (ownerid, name, phone)
BREED (breedname)
DOG (dogid, name, ownerid, kennelname, breedname, mothername, fathername)
ATTENDANCE (dogid, showname, opendate, place)
SHOW (showname, opendate, closedate)

Task 2: Relational Constraints

Description: Consider the following relational schema:

Emp(<u>eid</u>, ename, age, salary)
Works(<u>eid, did</u>, pcttime)
Dept(<u>did</u>, dname, budget, managerid)

Task 2.1: Give an example of a referential constraint that involves the Dept relation.
The referential constraints for establishing consistency in the database are:

- The FK attribute did from Works relation references to the PK did from Dept relation
- The FK attribute eid from Works relation references to the PK eid from Emp relation

Task 2.2: What are the options for enforcing this constraint when a user attempts to delete a Dept tuple?

If a department tuple with a specific did value from the Dept relation is about to be deleted, then this deletion is **propagated** to the **corresponding** tuples in the Works relation with the same did value to avoid inconsistency. The Emp tuples remain unaffected.

Description: Consider the relational schema below.
Task 2.3: What are the best possible primary keys in each relation?

- **employee**(person_name, street, city)
- **works**(person_name, company_name, salary)
- **company**(company_name, city)

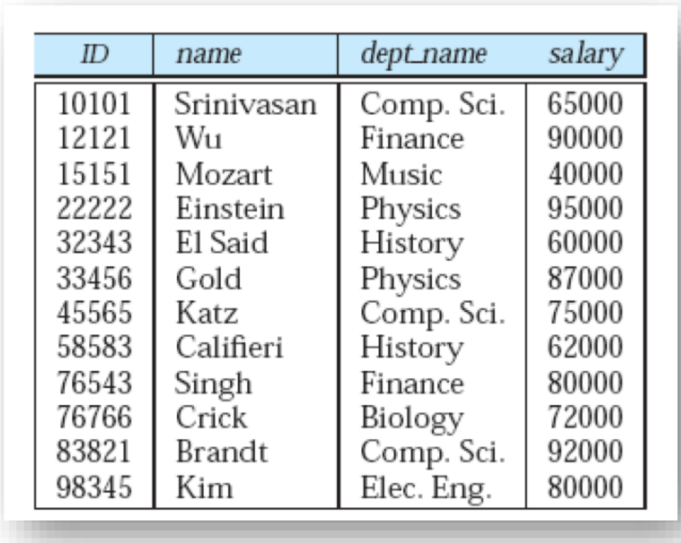
The PKs for each relation and the corresponding FKs are:

- person_name is PK in relation **employee**
- company_name is PK is relation **company**
- person_name in relation **works** references to the person_name in employee relation
- company_name is relation **works** reference to the company_name in company relation.

Description: In the instance of the **instructor** relation shown below, no two instructors have the same name.

Task 2.4: From this, can we conclude that the attribute name can be used as a superkey (or primary key) of the relation instructor?

Instructor



ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

The attributes: ID and name are unique. That is, each of them can uniquely identify a tuple in the relation. The composite attribute: {ID, name} is a superkey, since if we remove e.g., the name attribute then we obtain the ID, which itself uniquely identifies each tuple. The same holds true with the ID attribute. Hence, the minimum sufficient attribute is either the ID or the name (both are called then candidate keys). We choose one of them, say the name to be the PK. However, we decide the name to be the PK based only on the instance of this relation. If, in the future, it might be the case to have an instructor with the same name with some of the already existing instructors, then we will not be able to insert this instructor in the relation to avoid violation of the integrity constraint. At the time being, the name is considered unique, but for future reference this consideration might cause problems. Thus, it is advisable to consider as PK the ID which is something that we can determine its value to a new incoming instructor (who might have the same name with some other existing instructors in the relation). And this is easy, if let's say we define ID as a sequential number attribute, e.g., choosing a number which does not exist in the current relation instance or just the next integer of the last ID in the relation (98346, and so on…) . This cannot be easily determinable with an instructor's name since we cannot 'change' their name 😊. Based on this reasoning, ID is advised to be the PK for current and future **maintenance** of the consistency in our relation.