Weekly exercises IN3020&4020 (2026)
Sheet 1: Relational Data Model

1 The relational model revision

1. Briefly recall the main concepts and ideas of the relational data model:
   - relation: relation name, attribute names, signature, relation schema, tuples, relation state, attribute values, domain
   - relation properties: tuples are unordered and unique, attribute names in the signature may be seen as ordered or unordered

   Solution: See the slides

2. Give the ideas of the following notions and explain the differences between them:
   - Super key
   - Candidate key
   - Primary key
   - Foreign key

   Solution: Super key is a property of a set of attributes in a relation (with tuples). A set of attributes is a super key in a relation if the tuples of the relation projected (i.e., restricted) to these attributes do not have repetitions.

   Candidate key is also a property of a set of attributes in a relation. A set of attributes is candidate key if it is a minimal super key—that is , it is a super key and no subset of it is a super key.

   Primary key is from a different category. In fact, 'primary key' is a short version of 'primary key constraint', and it is a declaration for a set of attributes in a relation schema (i.e., no tuples, only the relation name and attribute names). This declaration imposes a constraint of any relation state (i.e., tuples, data itself) for the schema; in particular,

it says that any relation state that satisfies this schema with primary key must have these attributes as super key. Additional restriction is that only one primary key may be declared for the schema.

Foreign key has two meanings. First, in the spirit of super key and candidate key, it is a property of two sets of attributes in two relations. It holds when the attributes of the second relation (alone) is a super key and each tuple of the first relation projected to the corresponding set of attributes appear in the second relation (projected to its set). The second meaning is the short version of 'foreign key constraint'. In spirit of the primary key, it is a declaration for two sets of attributes for two (not necessarily different) relation schemas. Then, two relations over these two schemas satisfy this constraint if these two sets of attributes are a foreign key in the first sense.

3. Why are the following statements false?

   - A relation cannot have multiple super keys.
   - A candidate key can include excess attributes as long as the key still uniquely identifies tuples.
   - A foreign key cannot refer to a candidate key in the same relation.

   Solution: All three follow from the answers to the previous question. Further good examples for illustration:

   - In a relation with one tuple, every set of attributes is a super key.
   - In such a relation, all candidate keys are the singleton sets of attributes ('singleton set' is a set with exactly one element)
   - Every set of attributes in every relation is a foreign key referring to itself.

4. (*) In the lectures, we defined: 'Relation with schema $R[A_1, A_2, \ldots, A_n]$ *satisfies* the key constraint if the key subset of the constraint is a super key in the relation'. Explain what is the problem if we replace 'super key' by 'candidate key' in this definition (in this case, a problem is an undesirable, counter-intuitive behaviour).

   Solution: The main problem is that many relations that intuitively should satisfy such key constraint will actually not do it. In particular, a relation with schema $City[Name, Longitude, Latitude, \ldots]$ with key constraint $\{Longitude, Latitude\}$ (which intuitively means that the geographical coordinates of the city center uniquely identify the city), cannot contain only one city. Indeed, relation with a single

tuple $City\langle Oslo, 59.913333, 10.738889, \ldots\rangle$ has $\{Longitude, Latitude\}$ as a super key, but not as a candidate key (while $\{Longitude\}$ and $\{Latitude\}$ are candidate keys).

5. Can you think of other types of constraints that we may put on data, that are not expressible via primary and foreign keys? Hint: look for "Domain Constraints" the Web.

   Solution: Domain constraints specify that the value of each attribute must be an atomic value from the a specific domain. Note that, in our formalisation of the relational model, there is no need to specify domain constraints explicitly, because each attribute automatically comes with its domain. Same holds for SQL. You can read more about various types of constraints in Section 5.2 of R. Elmasri and S.B. Navathe book.

2 PostgreSQL Practical Review

Note: To use the PostgreSQL databases in this course, you should log in to a Linux terminal at IFI or log in remotely. Then,
`psql -h dbpg-ifi-kurs02 -U [username] -d in3020_[username]`
will log you in to you personal database. Here you will be able to create tables, insert, update, etc., while
`psql -h dbpg-ifi-kurs02 -U [username] -d fdb`
will log you in to the movie database. This database is read only.

1. Log in to your PostgreSQL environment; information on how this can be done is found at `https://www.uio.no/studier/emner/matnat/ifi/IN2090/h21/programvare/postgres.pdf`

2. Create the table `Ships` with the following fields (i.e., attributes):

   - `ship_id`
   - `name`

   Make attribute `ship_id` the primary key.

   Solution:

   ```
   CREATE TABLE Ships (
     id    INT,
     name  VARCHAR,
     PRIMARY KEY id
   ) ;
   ```

3. Insert the following data into the table `Ships`:

- (1, Pequod)
- (2, Titanic)
- (3, Le pharaon)

Solution:

```
INSERT INTO Ships
  VALUES (1, 'Pequod'), (2, 'Titanic'), (3, 'Le pharaon');
```

4. Create a table `Persons` with the attributes:
   - `name`
   - `ssn`
   - `street address`
   - `ZIP code`

   Use reasonable datatypes.

   Solution:

```
CREATE TABLE Persons (
  name  VARCHAR,
  ssn   CHAR(9),
  street_address    VARCHAR,
  zip   CHAR(5)
);
```

5. What is a reasonable primary key for `Persons`?

   Solution: Social security number (ssn) is the best candidate for primary key. However, if a schema designer wants to allow for two people with the same ssn, all attributes are a safe option for the primary key.

6. Alter table`Persons` and add a primary key.

   Solution:

```
ALTER TABLE Persons add PRIMARY KEY(ssn);
```

7. Insert the following data into `Persons`:
   - (Ishmael, 111111110, 20 Cooper Square Manhattan, 10001)

- (Captain Ahab, 111111112, 16 Nantucket Avenue, 02554)
- (Queequeg, 111111113, Rocovoco, 99991)
- (Molly Brown, 111111114, Hannibal Missouri, 63401)
- (Edmond Dantes, 111111115, Marseille, 13003)

Solution:

```
INSERT INTO Persons
VALUES (Ishmael, 111111110, 20 Cooper Square Manhattan, 10001, 1),
       (Captain Ahab, 111111112, 16 Nantucket Avenue, 02554, 1),
       (Queequeg, 111111113, Rocovoco, 99991, 1),
       (Molly Brown, 111111114, Hannibal Missouri, 63401, 2),
       (Edmond Dantes, 111111115, Marseille, 13003, 3) ;
```

8. Create a table `Crew` with attributes
   - `ssn`
   - `ship_id`

   Solution: See the solution to Question 11

9. Let the combination of `ssn` and `ship_id` be the primary key.

   Solution: See the solution to Question 11

10. Add foreign keys to `Persons` and `Ships`.

    Solution:

```
CREATE TABLE Crew (
  ssn CHAR(9) REFERENCES Persons(ssn),
  ship_id INT REFERENCES Ships(ship_id),
  PRIMARY KEY(ssn, ship_id)
);
```

11. Insert the following data into `Crew`:
    - (111111110, 1)
    - (111111112, 1)
    - (111111113, 2)
    - (111111114, 3)

    Solution:

```
INSERT INTO Crew (
VALUES ('111111110', 1),
       ('111111112', 1),
       ('111111113', 1),
       ('111111114', 2),
       ('111111115', 3)
```

12. Use SQL to find all persons with an address containing Manhattan.

    Solution:

    ```
    SELECT * FROM persons
    WHERE street_address LIKE '%Manhattan%' ;
    ```

13. Print out the name of the persons that were crew on the ship `Pequod`. Order the names alphabetically.

    Solution:

    ```
    SELECT p.name FROM persons p
    INNER JOIN crew c ON p.ssn = c.ssn
    INNER JOIN ships s ON c.ship_id = s.ship_id
    WHERE s.name LIKE 'Pequod'
    ORDER BY p.name ;
    ```

14. Use SQL to find how many crew members there were on the ship `Titanic`.

    Solution:

    ```
    SELECT COUNT(*) AS crew_on_titanic FROM persons p
    INNER JOIN crew c ON p.ssn = c.ssn
    INNER JOIN ships s ON c.ship_id = s.ship_id
    WHERE s.name LIKE 'Titanic' ;
    ```