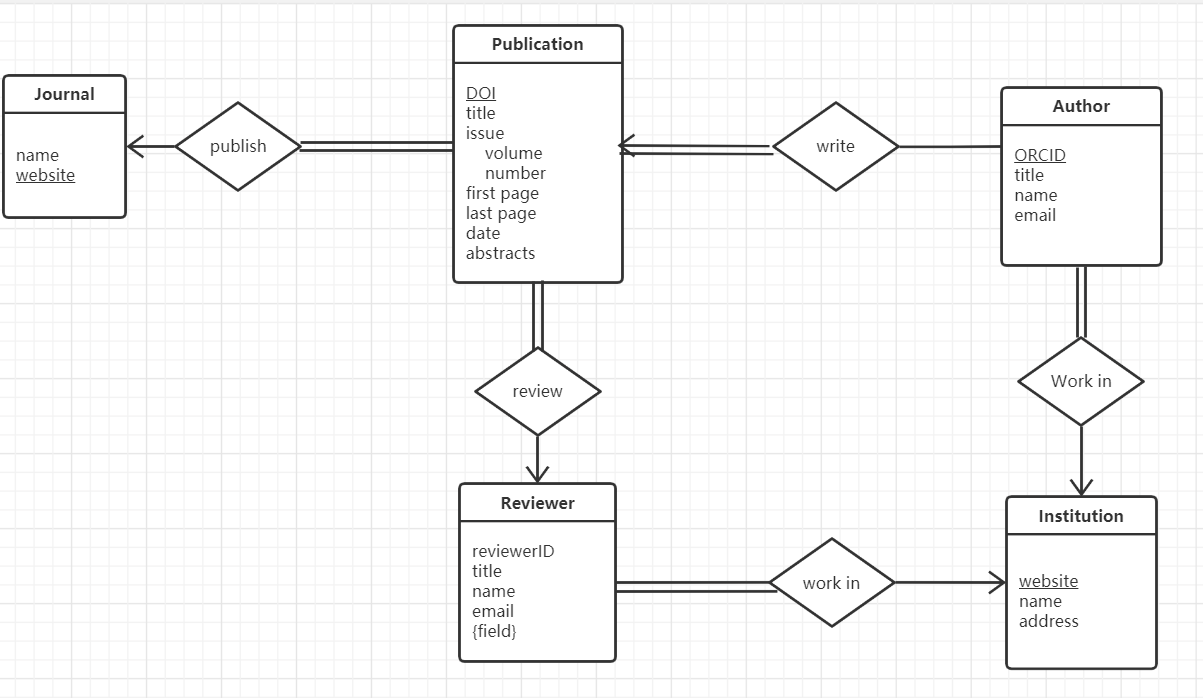
## MatricNO: 200009834

## 1. Database modelling

### (a):



### (b):

Publication(DOI, publication\_title, volume, number, first\_page, last\_page, date, abstracts, journal\_website, reviewerID)

Entity set: Publication

Primary key: DOI

Foreign key: journal\_website, reviewerID

Not null attribute: publication\_title , volume, number, first\_page, last\_page, date, abstracts

Composite attribute: issue

Write\_author(DOI, ORCID)

Entity set: Writer\_author

Primary key: DOI, ORCID

Foreign key: DOI, ORCID

Author(ORCID, title, name, email, institution\_website)

Entity set: Author

Primary key: ORCID

Foreign key: institution\_website

Not null attribute: name, email

Institution(institution\_website, institution\_name, institution\_address)

Entity set: Institution

Primary key: institution\_website

Not null attribute: institution\_name, institution\_address

Journal(journal\_website, name)

Entity set: Journal

Primary key: journal\_website

Not null attribute: name

Reviewer(reviewerID, title, reviewer\_name, reviewer\_email, institution\_website)

Entity set: Reviewer

Primary key: reviewerID

Foreign key: institution\_website

Not null attribute: reviewer\_name, reviewer\_email

Multivalued attribute: field

Reviewer\_fields(reviewerID, field)

Entity set: Reviewer\_fields

Primary key: reviewerID

Foreign key: reviewerID

Not null attribute: field

### (C):

1. SELECT institution\_name, institution\_website FROM Institution

ORDER BY institution\_name ASC;

2. SELECT reviewer\_name, reviewer\_email

FROM Reviewer NATURAL JOIN

(SELECT reviewerID FROM Reviewer\_fields WHERE field = 'DBSM');

3. SELECT publication\_title, abstracts

FROM Publication NATURAL JOIN

(SELECT DOI, ORCID

FROM Write\_author NATURAL JOIN

(SELECT ORCID

FROM Author NATURAL JOIN

(Institution WHERE institution\_name = 'University of St Andrews')

)

)

## 2. SQL

### (a):

CREATE TABLE Bird (

ring\_id CHAR(5),

species VARCHAR(20),

date\_ringed DATE,

age\_ringed INTEGER,

PRIMARY KEY (ring\_id)

);

CREATE TABLE Location (

loc\_id CHAR(5),

loc\_name VARCHAR(20),

latitude DOUBLE(5),

longitude DOUBLE(5),

postcode CHAR(6),

county VARCHAR(20),

PRIMARY KEY (loc\_id)

);

CREATE TABLE Staff (

staff\_id CHAR(5),

name VARCHAR(20),

address VARCHAR(20),

phone INTEGER,

PRIMARY KEY (staff\_id)

);

CREATE TABLE Observation (

ring\_id CHAR(5),

loc\_id CHAR(5),

date DATE,

staff\_id CHAR(5),

weight INTEGER,

bird\_length INTEGER,

wingspan DOUBLE(5),

PRIMARY KEY (ring\_id, loc\_id, staff\_id),

FOREIGN KEY (ring\_id) REFERENCES Bird,

FOREIGN KEY (loc\_id) REFERENCES Location,

FOREIGN KEY (staff\_id) REFERENCES Staff

);

### (b):

1. SELECT loc\_name, latitude, longitude FROM Location

WHERE county = 'Fife'

ORDER BY postcode;

2. SELECT DISTINCT species FROM Bird NATURAL JOIN (SELECT count(ring\_id) FROM Observation);

3. SELECT DISTINCT loc\_name,

(SELECT count(ring\_id) FROM Observation NATURAL JOIN Bird WHERE species = 'swan' AND date LIKE '2019-%-%' ) AS "swan\_number"

FROM Location;

### (c):

ALTER TABLE Location

ADD COLUMN country VARCHAR(10) NOT NULL DEFAULT 'UK';

### (d):

Using GRANT to staff, Here give an example, Tom is a normal staff, he only can access the bird information. Mary is a manager; she can access the staff information and change the bird information. So, it can use GRANT to get the access data:

GRANT SELECT ON Bird TO Tom;

GRANT INSERT, DELETE ON Bird, Staff TO Mary;

GRANT UPDATE ON Bird, Staff TO Mary;

GRANT SELECT ON Bird, Staff TO Mary;

Then, I can use the view to freely construct the relationship view that I want to expose to other objects, thereby protecting other data from being accessed by others. There is a prerequisite that the user needs to have SELECT permissions.

For example:

CREATE VIEW BirdView AS

SELECT ring\_id, species, date\_ringed, age\_ringed

FROM Bird

Also revoke permissions through REVOKE, so it can avoid data leakage.

## 3. Normalization

### (a):

According to the courseware, the definition of unnormalization is: A table that may contain one or more repeating groups. My understanding is that the data is redundant.

According to the courseware, the definition of 1NF is: A relation in which the intersection of each row and column contains one and only one value. My understanding is that only one data can appear at the intersection of the x-axis and y-axis of the table. So, this table does not comply with 1NF.

1. In relational databases, data tables that do not conform to 1NF cannot be created in RDBMS.
2. It is very convenient for humans to observe, but the computer cannot recognize the association.
3. The row attribute naming method cannot be written in the relational database.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Owner\_name | Owner\_phone | Species | PetID | Pet\_name | Food | BoxNo | Amount |
| Jerry Swan | 3265 | Gerbil | 2 | Mario | Muesli | 2 | 1 x 10 g |
| Jerry Swan | 3265 | Gerbil | 2 | Mario | Seeds | 7 | 1 x 5 g |
| Jerry Swan | 3265 | Gerbil | 3 | Luigi | Muesli | 2 | 1 x 10 g |
| Jerry Swan | 3265 | Gerbil | 3 | Luigi | Seeds | 7 | 1 x 5 g |
| Aiko Suzuki | 2466 | Rabbit | 9 | Yoshi | Hay | 8 | 2 x 500g |
| Diego Garcia | 4677 | Guinea pig | 5 | Fluffy | Hay | 8 | 2 x 100g |
| Diego Garcia | 4677 | Guinea pig | 5 | Fluffy | Seeds | 3 | 2 x 20g |
| Tom Katz | 7824 | Cat | 6 | Larry | GoCat | 1 | 3 x 100g |

### (b):

Owner\_name → Owner\_phone

PetID → Species, Pet\_name, Owner\_name, Owner\_phone

BoxNo → Food

Owner\_name, PetID, BoxNo → Amount

### (C):

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Owner\_name | Owner\_phone | Species | PetID | Pet\_name | Food | BoxNo | Amount |
| Jerry Swan | 3265 | Gerbil | 2 | Mario | Muesli | 2 | 1 x 10 g |
| Jerry Swan | 3265 | Gerbil | 2 | Mario | Seeds | 7 | 1 x 5 g |
| Jerry Swan | 3265 | Gerbil | 3 | Luigi | Muesli | 2 | 1 x 10 g |
| Jerry Swan | 3265 | Gerbil | 3 | Luigi | Seeds | 7 | 1 x 5 g |
| Aiko Suzuki | 2466 | Rabbit | 9 | Yoshi | Hay | 8 | 2 x 500g |
| Diego Garcia | 4677 | Guinea pig | 5 | Fluffy | Hay | 8 | 2 x 100g |
| Diego Garcia | 4677 | Guinea pig | 5 | Fluffy | Seeds | 3 | 2 x 20g |
| Tom Katz | 7824 | Cat | 6 | Larry | GoCat | 1 | 3 x 100g |
| Jerry Swan | 4567 | Cat | 7 | Wu | FunCat | 4 | 3 x 100g |

For example, I now insert an extra row in the table with the name Jerry Swan. At this time, when updating the data of Jerry Swan in the table, two people with the same name will appear, which will cause problems when we update the phone number or other data.

### (d):

First this table is in 1NF, and remove partial dependencies on the primary key by splitting into three tables:

(PetID, Species, Pet\_name, Owner\_name, Owner\_phone)

(BoxNo, Food)

(Owner\_name, PetID, BoxNo, Amount)

This set of three tables are now in 2NF.

Of the three tables, the last two are in 3NF, but the first one has transitive dependencies. So, it should split that into two tables

(Owner\_name, Owner\_phone)

(PetID, Species, Pet\_name)

These tables are in 3NF

### (e):

1. Too many tables increase the cost of JOIN. For example, I want to query what foods an owner’s pet needs.

2. If I want to update a customer’s pet information, it means I need to access multiple tables, which will make the query less efficient.