Database project: Zoo

Halmai Levente
RS7P1M

1. Specification:

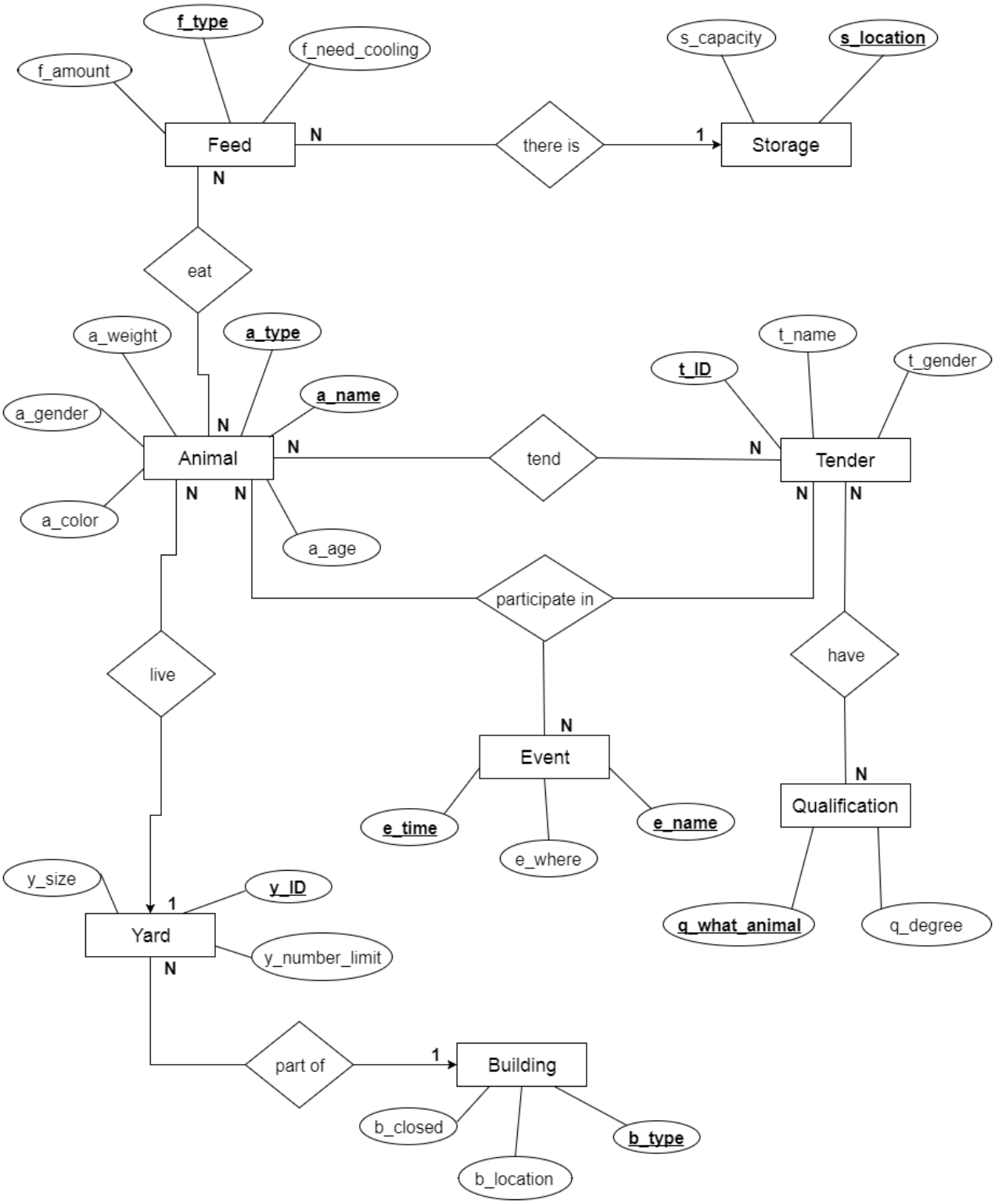
In this project I will demonstrate a database of a zoo. I planned to get 9 species in the zoo, each have at least one, but mainly more. The animals have a private ID, which discriminates each other. Each animal have a zoo tender, who only can be their tender if he/she has a qualification for that type of animal. An animal can have one or more tender(s), and a tender can tend one or more animals from the same type or get qualification for more than one type of animal. The zoo's main attractions are special events in the year. One or more animals can participate with their tenders in the events to show their production and hard work of the year interval. The zoo shows each event once a year, on predetermined days and time, because it could be boring for the visitors and animals if there is too many of the same animal show. Aside from the events the we can get informed about the feeding of the animals, the type of foods and about the available food in the storage. And of course we can check the living yard of the animals, where this yard is located in the zoo, if it has any small building in it, and the equipments (e.g. games of the animal, like balls, etc.) in the yard.

2. Relation model:

ANIMAL(**a_name**, **a_type**, a_age, a_gender, a_color, a_weight, y_ID)
TENDER(**t_ID**, t_name, t_gender)
QUALIFICATION(**q_what_animal**, q_degree)
EVENT(**e_name**, **e_time**, e_where)
FEED(**f_type**, f_amount, f_need_cooling, s_location)
STORAGE(**s_location**, s_capacity)
YARD(**y_ID**, y_size, y_number_limit, b_type)
BUILDING(**b_type**, b_location, b_closed)

TEND(**animal.a_name**, **animal.a_type, tender.t_ID**)
HAVE(**tender.t_ID**, **qualification.q_what_animal**)
PARTICIPATEIN(**animal.a_name**, **animal.a_type, tender.t_ID**, **event.e_name**, **event.e_time**)
EAT(**animal.a_name**, **animal.a_type**, **feed. f_type**)

3. E/R diagram:


f_type
f_need_cooling
f_amount
Feed
N
there is
1
Storage
s_capacity
s_location
N
eat
a_weight
a_type
a_name
a_gender
N
Animal
N
tend
N
Tender
t_ID
t_name
t_gender
a_color
N
N
a_age
N
N
participate in
live
have
N
Event
N
Qualification
e_time
e_where
e_name
q_what_animal
q_degree
y_size
1
y_ID
Yard
y_number_limit
N
part of
1
Building
b_closed
b_location
b_type

4. SQL: Create, Insert, Drop:

```
CREATE TABLE building
(
    b_type VARCHAR2(100) NOT NULL PRIMARY KEY,
    b_location VARCHAR2(100) NOT NULL,
    b_closed VARCHAR2(5) NOT NULL
);

CREATE TABLE yard
(
    y_ID VARCHAR2(10) NOT NULL PRIMARY KEY,
    y_size NUMBER NOT NULL,
    y_number_limit NUMBER NOT NULL,
    b_type  VARCHAR2(100) NOT NULL,
    FOREIGN KEY(b_type) REFERENCES building(b_type)
);

CREATE TABLE animal
(
    a_name VARCHAR2(100) NOT NULL,
    a_type VARCHAR2(100) NOT NULL,
    a_age NUMBER NOT NULL,
    a_gender VARCHAR2(10) NOT NULL,
    a_color VARCHAR2(100),
    a_weight NUMBER,
    y_ID VARCHAR2(10) NOT NULL,
    PRIMARY KEY (a_name, a_type),
    FOREIGN KEY(y_ID) REFERENCES yard(y_ID)
);

CREATE TABLE tender
(
    t_ID VARCHAR2(10) NOT NULL PRIMARY KEY,
    t_name VARCHAR2(100) NOT NULL,
    t_gender VARCHAR2(10)
);

CREATE TABLE qualification
(
    q_what_animal VARCHAR2(100) NOT NULL PRIMARY KEY,
    q_degree VARCHAR2(100) NOT NULL
);

CREATE TABLE event
(
    e_name VARCHAR2(200) NOT NULL,
    e_time DATE NOT NULL,
    e_where VARCHAR2(100) NOT NULL,
    PRIMARY KEY(e_name, e_time)
);

CREATE TABLE storage
(
    s_location VARCHAR2(100) NOT NULL PRIMARY KEY,
    s_capacity NUMBER NOT NULL
);
```

```
CREATE TABLE feed
(
    f_type VARCHAR2(100) NOT NULL PRIMARY KEY,
    f_amount NUMBER NOT NULL,
    f_need_cooling VARCHAR2(5) NOT NULL,
    s_location VARCHAR2(100) NOT NULL,
    FOREIGN KEY(s_location) REFERENCES storage(s_location)
);

CREATE TABLE tend
(
    a_name VARCHAR2(100) NOT NULL,
    a_type VARCHAR2(100) NOT NULL,
    t_ID VARCHAR2(10) NOT NULL,
    FOREIGN KEY(a_name, a_type) REFERENCES animal(a_name, a_type),
    FOREIGN KEY(t_ID) REFERENCES tender(t_ID),
    PRIMARY KEY(a_name, a_type, t_id)
);

CREATE TABLE have
(
    t_ID VARCHAR2(10) NOT NULL,
    q_what_animal VARCHAR2(100) NOT NULL,
    FOREIGN KEY(t_ID) REFERENCES tender(t_ID),
    FOREIGN KEY(q_what_animal) REFERENCES qualification(q_what_animal),
    PRIMARY KEY(t_ID, q_what_animal)
);

CREATE TABLE participatein
(
    a_name VARCHAR2(100) NOT NULL,
    a_type VARCHAR2(100) NOT NULL,
    t_ID VARCHAR2(10) NOT NULL,
    e_name VARCHAR2(200) NOT NULL,
    e_time DATE NOT NULL,
    FOREIGN KEY(a_name, a_type) REFERENCES animal(a_name, a_type),
    FOREIGN KEY(t_ID) REFERENCES tender(t_ID),
    FOREIGN KEY(e_name, e_time) REFERENCES event(e_name, e_time),
    PRIMARY KEY(a_name, a_type, t_ID, e_name, e_time)
);


CREATE TABLE eat
(
    a_name VARCHAR2(100) NOT NULL,
    a_type VARCHAR2(100) NOT NULL,
    f_type VARCHAR2(100) NOT NULL,
    FOREIGN KEY(a_name, a_type) REFERENCES animal(a_name, a_type),
    FOREIGN KEY(f_type) REFERENCES feed(f_type),
    PRIMARY KEY(a_name, a_type, f_type)
);

--BUILDING(b_type, b_location, b_closed)
INSERT INTO building VALUES('parrot house', 'north-center', 'true');
INSERT INTO building VALUES('ice park', 'north-west-center', 'false');
INSERT INTO building VALUES('safari park', 'south-east', 'false');
INSERT INTO building VALUES('swamp park', 'west', 'false');
INSERT INTO building VALUES('jungle park', 'east', 'false');
INSERT INTO building VALUES('magic forest', 'east-center', 'false');
```

```
--YARD(y_ID, y_size, y_number_limit, b_type)
INSERT INTO yard VALUES('y-201', 300, 4,'safari park');   --lion
INSERT INTO yard VALUES('y-202', 350, 3,'safari park');   --giraffe
INSERT INTO yard VALUES('y-203', 200, 5,'swamp park');    --hippo
INSERT INTO yard VALUES('y-204', 340, 6,'safari park');   --zebra
INSERT INTO yard VALUES('y-205', 400, 4,'safari park');   --elephant
INSERT INTO yard VALUES('y-206', 100, 15,'parrot house'); --parrot
INSERT INTO yard VALUES('y-207', 300, 4,'jungle park');   --tiger
INSERT INTO yard VALUES('y-208', 400, 5,'magic forest');  --bear
INSERT INTO yard VALUES('y-209', 200, 10,'ice park');     --penguin

--ANIMAL(a_name, a_type, a_age, a_gender, a_color, a_weight, y_ID)
INSERT INTO animal VALUES('Chakra','lion',4,'male','yellow-orange', 270, 'y-201');
INSERT INTO animal VALUES('Groove','lion',2,'male','yellow-orange', 240, 'y-201');
INSERT INTO animal VALUES('Jazzy','lion',3,'female','yellow-orange', 250, 'y-201');
INSERT INTO animal VALUES('Bessie','giraffe',5,'female','brown-orange', 300, 'y-202');
INSERT INTO animal VALUES('Loony','giraffe',6,'male','brown-orange', 350, 'y-202');
INSERT INTO animal VALUES('Duncan','hippopotamus',6,'male','gray', 310, 'y-203');
INSERT INTO animal VALUES('Kai','hippopotamus',8,'female','gray', 320, 'y-203');
INSERT INTO animal VALUES('Goosebump','zebra',4,'female','black-white', 220, 'y-204');
INSERT INTO animal VALUES('Hansel','zebra',3,'male','black-white', 220, 'y-204');
INSERT INTO animal VALUES('Pistachio','zebra',5,'female','black-white', 240, 'y-204');
INSERT INTO animal VALUES('Newman','elephant',15,'male','gray', 1040, 'y-205');
INSERT INTO animal VALUES('Pistachio','elephant',18,'female','gray', 1100, 'y-205');
INSERT INTO animal VALUES('Otto','parrot',2,'male','yellow', 1, 'y-206');
INSERT INTO animal VALUES('Kenji','parrot',1,'female','blue', 2, 'y-206');
INSERT INTO animal VALUES('Crook','parrot',2,'female','red', 1, 'y-206');
INSERT INTO animal VALUES('Sherlock','parrot',3,'male','green', 2, 'y-206');
INSERT INTO animal VALUES('Rubert','tiger',5,'male','orange-black', 270, 'y-207');
INSERT INTO animal VALUES('Sasha','tiger',3,'female','white', 250, 'y-207');
INSERT INTO animal VALUES('Talia','bear',5,'female','brown', 550, 'y-208');
INSERT INTO animal VALUES('Kodo','bear',6,'female','brown', 570, 'y-208');
INSERT INTO animal VALUES('Mulder','bear',4,'male','brown', 600, 'y-208');
INSERT INTO animal VALUES('Galileo','penguin',3,'male','black-white', 10, 'y-209');
INSERT INTO animal VALUES('Shady','penguin',4,'female','black-white', 11, 'y-209');
INSERT INTO animal VALUES('McKenzie','penguin',1,'female','gray', 14, 'y-209');
INSERT INTO animal VALUES('Zander','penguin',3,'male','black-white', 13, 'y-209');

--TENDER(t_ID, t_name, t_gender)
INSERT INTO tender VALUES('t-101', 'Kis Pista', 'male');
INSERT INTO tender VALUES('t-102', 'Szepessy Tamara', 'female');
INSERT INTO tender VALUES('t-103', 'Kovacs Peter', 'male');
INSERT INTO tender VALUES('t-104', 'Hufnagel Bea', 'female');
INSERT INTO tender VALUES('t-105', 'Nagy Laszlo', 'male');
INSERT INTO tender VALUES('t-106', 'Szakats Fulop', 'male');
INSERT INTO tender VALUES('t-107', 'Bognar Balazs', 'male');
INSERT INTO tender VALUES('t-108', 'Revesz Fulop', 'male');
INSERT INTO tender VALUES('t-109', 'Fazekas Adrienn', 'female');
INSERT INTO tender VALUES('t-110', 'Makkay Ilona', 'female');

--QUALIFICATION(q_what_animal, q_degree)
INSERT INTO qualification VALUES('lion','base');
INSERT INTO qualification VALUES('tiger','high');
INSERT INTO qualification VALUES('giraffe','base');
INSERT INTO qualification VALUES('hippopotamus','base');
INSERT INTO qualification VALUES('zebra','high');
INSERT INTO qualification VALUES('elephant','base');
INSERT INTO qualification VALUES('parrot','high');
INSERT INTO qualification VALUES('bear','high');-
INSERT INTO qualification VALUES('penguin','base');
```

```
--HAVE(tender.t_ID, qualification.q_what_animal)
INSERT INTO have VALUES('t-101', 'lion');
INSERT INTO have VALUES('t-101', 'tiger');
INSERT INTO have VALUES('t-101', 'bear');
INSERT INTO have VALUES('t-102', 'lion');
INSERT INTO have VALUES('t-103', 'parrot');
INSERT INTO have VALUES('t-103', 'penguin');
INSERT INTO have VALUES('t-104', 'hippopotamus');
INSERT INTO have VALUES('t-105', 'zebra');
INSERT INTO have VALUES('t-105', 'giraffe');
INSERT INTO have VALUES('t-105', 'elephant');
INSERT INTO have VALUES('t-106', 'bear');
INSERT INTO have VALUES('t-106', 'giraffe');
INSERT INTO have VALUES('t-107', 'tiger');
INSERT INTO have VALUES('t-108', 'giraffe');
INSERT INTO have VALUES('t-108', 'zebra');
INSERT INTO have VALUES('t-109', 'penguin');
INSERT INTO have VALUES('t-109', 'parrot');
INSERT INTO have VALUES('t-110', 'tiger');
INSERT INTO have VALUES('t-110', 'lion');

--TEND(animal.a_name, animal.a_type, tender.t_ID)
INSERT INTO tend VALUES('Chakra','lion','t-101');
INSERT INTO tend VALUES('Chakra','lion','t-102');
INSERT INTO tend VALUES('Groove','lion','t-102');
INSERT INTO tend VALUES('Groove','lion','t-110');
INSERT INTO tend VALUES('Groove','lion','t-101');
INSERT INTO tend VALUES('Jazzy','lion','t-101');
INSERT INTO tend VALUES('Jazzy','lion','t-110');
INSERT INTO tend VALUES('Bessie','giraffe','t-105');
INSERT INTO tend VALUES('Bessie','giraffe','t-106');
INSERT INTO tend VALUES('Loony','giraffe','t-108');
INSERT INTO tend VALUES('Duncan','hippopotamus','t-104');
INSERT INTO tend VALUES('Kai','hippopotamus','t-104');
INSERT INTO tend VALUES('Goosebump','zebra','t-105');
INSERT INTO tend VALUES('Goosebump','zebra','t-108');
INSERT INTO tend VALUES('Hansel','zebra','t-108');
INSERT INTO tend VALUES('Pistachio','zebra','t-105');
INSERT INTO tend VALUES('Newman','elephant','t-105');
INSERT INTO tend VALUES('Pistachio','elephant','t-105');
INSERT INTO tend VALUES('Otto','parrot','t-103');
INSERT INTO tend VALUES('Kenji','parrot','t-103');
INSERT INTO tend VALUES('Kenji','parrot','t-109');
INSERT INTO tend VALUES('Crook','parrot','t-103');
INSERT INTO tend VALUES('Crook','parrot','t-109');
INSERT INTO tend VALUES('Sherlock','parrot','t-109');
INSERT INTO tend VALUES('Rubert','tiger','t-101');
INSERT INTO tend VALUES('Rubert','tiger','t-107');
INSERT INTO tend VALUES('Sasha','tiger','t-110');
INSERT INTO tend VALUES('Sasha','tiger','t-101');
INSERT INTO tend VALUES('Talia','bear','t-101');
INSERT INTO tend VALUES('Kodo','bear','t-106');
INSERT INTO tend VALUES('Mulder','bear','t-101');
INSERT INTO tend VALUES('Mulder','bear','t-106');
INSERT INTO tend VALUES('Galileo','penguin','t-103');
INSERT INTO tend VALUES('Galileo','penguin','t-109');
INSERT INTO tend VALUES('Shady','penguin','t-109');
INSERT INTO tend VALUES('McKenzie','penguin','t-109');
INSERT INTO tend VALUES('Zander','penguin','t-103');
INSERT INTO tend VALUES('Zander','penguin','t-109');
```

```
--EVENT(e name, e time, e where)
INSERT INTO event VALUES('Fly with parrots', TO_DATE('2018-05-22','YYYY-MM-DD'),
'parrot house');
INSERT INTO event VALUES('Difference between zebras and crosswalk', TO_DATE('2018-
04-12','YYYY-MM-DD'), 'safari park');
INSERT INTO event VALUES('Can you run faster than a tiger?', TO_DATE('2018-04-
12','YYYY-MM-DD'), 'safari park');
INSERT INTO event VALUES('Fly with penguins, or how they fall from rocks',
TO_DATE('2018-04-12','YYYY-MM-DD'), 'ice park');
INSERT INTO event VALUES('Hide and seek with hippos underwater', TO_DATE('2018-06-
27','YYYY-MM-DD'), 'safari park');
INSERT INTO event VALUES('Sing with parrots', TO_DATE('2018-07-04','YYYY-MM-DD'),
'parrot house');
INSERT INTO event VALUES('Throw a bear and run!', TO_DATE('2018-02-14','YYYY-MM-
DD'), 'magic forest');
INSERT INTO event VALUES('Jumpscare an elephant with a mouse!', TO_DATE('2018-08-
01','YYYY-MM-DD'), 'safari park');

--PARTICIPATEIN(animal.a_name, animal.a_type, tender.t_ID, event.e_name,
event.e_time)
INSERT INTO participatein VALUES('Galileo','penguin','t-103','Fly with penguins, or
how they fall from rocks', TO_DATE('2018-04-12','YYYY-MM-DD'));
INSERT INTO participatein VALUES('Galileo','penguin','t-109','Fly with penguins, or
how they fall from rocks', TO_DATE('2018-04-12','YYYY-MM-DD'));
INSERT INTO participatein VALUES('Kenji','parrot','t-103','Fly with parrots',
TO_DATE('2018-05-22','YYYY-MM-DD'));
INSERT INTO participatein VALUES('Kenji','parrot','t-109','Fly with parrots',
TO_DATE('2018-05-22','YYYY-MM-DD'));
INSERT INTO participatein VALUES('Sherlock','parrot','t-109','Fly with parrots',
TO_DATE('2018-05-22','YYYY-MM-DD'));
INSERT INTO participatein VALUES('Hansel','zebra','t-108','Difference between
zebras and crosswalk', TO_DATE('2018-04-12','YYYY-MM-DD'));
INSERT INTO participatein VALUES('Pistachio','zebra','t-105','Difference between
zebras and crosswalk', TO_DATE('2018-04-12','YYYY-MM-DD'));
INSERT INTO participatein VALUES('Sasha','tiger','t-101','Can you run faster than a
tiger?', TO_DATE('2018-04-12','YYYY-MM-DD'));
INSERT INTO participatein VALUES('Sasha','tiger','t-110','Can you run faster than a
tiger?', TO_DATE('2018-04-12','YYYY-MM-DD'));
INSERT INTO participatein VALUES('Rubert','tiger','t-101','Can you run faster than
a tiger?', TO_DATE('2018-04-12','YYYY-MM-DD'));
INSERT INTO participatein VALUES('Rubert','tiger','t-107','Can you run faster than
a tiger?', TO_DATE('2018-04-12','YYYY-MM-DD'));
INSERT INTO participatein VALUES('Duncan','hippopotamus','t-104','Hide and seek
with hippos underwater', TO_DATE('2018-06-27','YYYY-MM-DD'));
INSERT INTO participatein VALUES('Kai','hippopotamus','t-104','Hide and seek with
hippos underwater', TO_DATE('2018-06-27','YYYY-MM-DD'));
INSERT INTO participatein VALUES('Crook','parrot','t-109','Sing with parrots',
TO_DATE('2018-07-04','YYYY-MM-DD'));
INSERT INTO participatein VALUES('Sherlock','parrot','t-109','Sing with parrots',
TO_DATE('2018-07-04','YYYY-MM-DD'));
INSERT INTO participatein VALUES('Talia','bear','t-101','Throw a bear and run!',
TO_DATE('2018-02-14','YYYY-MM-DD'));
INSERT INTO participatein VALUES('Kodo','bear','t-106','Throw a bear and run!',
TO_DATE('2018-02-14','YYYY-MM-DD'));
INSERT INTO participatein VALUES('Newman','elephant','t-105','Jumpscare an elephant
with a mouse!', TO_DATE('2018-08-01','YYYY-MM-DD'));
INSERT INTO participatein VALUES('Pistachio','elephant','t-105','Jumpscare an
elephant with a mouse!', TO_DATE('2018-08-01','YYYY-MM-DD'));
```

```
--STORAGE(s_location, s_capacity)
INSERT INTO storage VALUES('1. storage unit', 3000);
INSERT INTO storage VALUES('2. storage unit', 3560);
INSERT INTO storage VALUES('3. storage unit', 3610);
INSERT INTO storage VALUES('4. storage unit', 3940);
INSERT INTO storage VALUES('5. storage unit', 2500);

--FEED(f_type, f_amount, f_need_cooling, s_location)
INSERT INTO feed VALUES('hay', 1000,'false', '1. storage unit');--giraffe
INSERT INTO feed VALUES('straw', 800,'false', '1. storage unit');--zebra
INSERT INTO feed VALUES('salmon', 900,'true', '2. storage unit'); --bear
INSERT INTO feed VALUES('seafood', 400,'true', '2. storage unit');--penguin
INSERT INTO feed VALUES('raw chicken', 500,'true', '4. storage unit');--tiger
INSERT INTO feed VALUES('raw meat', 450,'true', '4. storage unit');--lion
INSERT INTO feed VALUES('fruit mix', 460,'false', '5. storage unit');--hippo
INSERT INTO feed VALUES('seed mix', 200,'false', '5. storage unit');--parrot
INSERT INTO feed VALUES('vegatable mix', 870,'true', '3. storage unit');--elephant

--EAT(animal.a_name, animal.a_type, feed. f_type)
INSERT INTO eat VALUES('Chakra','lion','raw meat');
INSERT INTO eat VALUES('Groove','lion','raw chicken');
INSERT INTO eat VALUES('Jazzy','lion','raw meat');
INSERT INTO eat VALUES('Bessie','giraffe','hay');
INSERT INTO eat VALUES('Loony','giraffe','hay');
INSERT INTO eat VALUES('Duncan','hippopotamus','fruit mix');
INSERT INTO eat VALUES('Kai','hippopotamus','fruit mix');
INSERT INTO eat VALUES('Goosebump','zebra','straw');
INSERT INTO eat VALUES('Hansel','zebra','straw');
INSERT INTO eat VALUES('Pistachio','zebra','straw');
INSERT INTO eat VALUES('Newman','elephant','vegatable mix');
INSERT INTO eat VALUES('Pistachio','elephant','vegatable mix');
INSERT INTO eat VALUES('Otto','parrot','seed mix');
INSERT INTO eat VALUES('Kenji','parrot','seed mix');
INSERT INTO eat VALUES('Crook','parrot','seed mix');
INSERT INTO eat VALUES('Sherlock','parrot','seed mix');
INSERT INTO eat VALUES('Rubert','tiger','raw chicken');
INSERT INTO eat VALUES('Sasha','tiger','raw meat');
INSERT INTO eat VALUES('Talia','bear','salmon');
INSERT INTO eat VALUES('Kodo','bear','salmon');
INSERT INTO eat VALUES('Mulder','bear','salmon');
INSERT INTO eat VALUES('Galileo','penguin','seafood');
INSERT INTO eat VALUES('Shady','penguin','seafood');
INSERT INTO eat VALUES('McKenzie','penguin','seafood');
INSERT INTO eat VALUES('Zander','penguin','seafood');

DROP TABLE building CASCADE CONSTRAINTS;
DROP TABLE yard CASCADE CONSTRAINTS;
DROP TABLE animal CASCADE CONSTRAINTS;
DROP TABLE tender CASCADE CONSTRAINTS;
DROP TABLE qualification CASCADE CONSTRAINTS;
DROP TABLE event CASCADE CONSTRAINTS;
DROP TABLE storage CASCADE CONSTRAINTS;
DROP TABLE feed CASCADE CONSTRAINTS;
DROP TABLE tend CASCADE CONSTRAINTS;
DROP TABLE have CASCADE CONSTRAINTS;
DROP TABLE participatein CASCADE CONSTRAINTS;
DROP TABLE eat CASCADE CONSTRAINTS;
```

5. SQL querries:

```
--List all animals whom particape in the events, and show which event, what is the
anaimal name, type and gender.
SELECT distinct animal.a_name, animal.a_type, animal.a_gender, participatein.e_name
FROM animal, participatein
WHERE animal.a_name=participatein.a_name AND animal.a_type=participatein.a_type
ORDER BY participatein.E_NAME;

--List all feed with storage place, and storage capacity
SELECT feed.F_TYPE, storage.S_LOCATION , storage.S_CAPACITY
FROM feed, storage
WHERE feed.S_LOCATION=storage.S_LOCATION;

--List how many of animals have in each type and in each gender
SELECT a.A_TYPE, a.A_GENDER, COUNT(a.A_NAME)
FROM animal a
GROUP BY a.A_TYPE, a.A_GENDER
ORDER BY a.A_TYPE;

--List how many type of qualification have each tender (name and id)
SELECT tender.T_ID, tender.T_NAME, COUNT(*) AS number_of_qualification
FROM tender, have
WHERE tender.T_ID=have.T_ID
GROUP BY tender.T_ID, tender.T_NAME
ORDER BY tender.T_ID;

--List all tenders name with how many animal in each type they tend, except parrots and
penguins, you can list parrot or penguin tender if he tend another type
SELECT tender.T_NAME, tend.A_TYPE, COUNT(*) AS countAnimal
FROM tender, tend
WHERE tend.T_ID=tender.T_ID  AND tend.A_TYPE NOT IN ('parrot','penguin')
GROUP BY tender.T_NAME,  tend.A_TYPE
ORDER BY tender.T_NAME;

--List how many qualified tender can tend each type of animal
SELECT q.Q_WHAT_ANIMAL, COUNT(distinct have.T_id) AS number_of_tender
FROM have, qualification q
WHERE have.Q_WHAT_ANIMAL=q.Q_WHAT_ANIMAL
GROUP BY q.Q_WHAT_ANIMAL;

--List all animal names and types which weight is more than all of the tiger and lions
weights
SELECT A_NAME, A_TYPE, A_WEIGHT
FROM animal
WHERE A_WEIGHT > ALL (SELECT A_WEIGHT FROM animal WHERE A_TYPE IN ('lion', 'tiger'));

--Select all animals, except elephants, who's weight is more than the avarage weight,
except the elephants
SELECT A_NAME, A_TYPE, A_WEIGHT
FROM animal
WHERE A_TYPE NOT LIKE 'elephant'
AND A_WEIGHT > (SELECT AVG(a.A_WEIGHT) FROM animal a WHERE a.A_TYPE NOT LIKE
'elephant')
ORDER BY A_WEIGHT DESC;

--SELECT all storage unit where the used amount in each unit is under the 1/3 of the
avarage maximum capacity of the units
SELECT S_LOCATION, SUM(F_AMOUNT)
FROM feed
GROUP BY S_LOCATION
HAVING SUM(F_AMOUNT) < (SELECT AVG(S_CAPACITY)/3 FROM storage);
```

```
--List all yard id with the number of animals, where the yard building type contains
"park" and the location conatins east or north
SELECT a.Y_ID, COUNT(a.A_NAME) AS number_of_animals
FROM animal a
GROUP BY a.Y_ID
HAVING a.Y_ID IN (
    SELECT yard.Y_ID
    FROM yard, building b
    WHERE yard.B_TYPE=b.B_TYPE AND b.B_TYPE LIKE '%park%' AND (b.B_LOCATION LIKE
'%east%' or b.B_LOCATION LIKE '%north%'))
ORDER BY a.Y_ID;

--List all storage unit with used and free capacity in it
SELECT f.S_LOCATION, SUM(f.F_AMOUNT) AS used_amount,  (s.S_CAPACITY- SUM(f.F_AMOUNT))
AS free_capacity
FROM feed f, storage s
WHERE f.S_LOCATION=s.S_LOCATION
GROUP BY f.S_LOCATION, s.S_CAPACITY
ORDER BY f.S_LOCATION;
```

6. SQL Update, Delete, View:

```
--View
DROP VIEW birds;

CREATE VIEW birds AS
SELECT A_NAME, A_TYPE, A_AGE, Y_ID
FROM animal
where A_TYPE IN ('parrot','penguin');

--Update
UPDATE ANIMAL
SET A_COLOR='dark-brown'
WHERE A_TYPE='bear';

--Alter table
ALTER TABLE storage
ADD ISFULL VARCHAR2(10) DEFAULT 'false';

ALTER TABLE storage
MODIFY ISFULL VARCHAR2(5);

ALTER TABLE storage
DROP COLUMN ISFULL;
```

7. RA queries:

1. List all animals whom particape in the events, and show which event, what is the anaimal name, type and gender.

   $$\pi_{a\_name,a\_type,a\_gender,e\_name}(animal \bowtie participatein)$$

2. List all feed with storage place, and storage capacity.

   $$\pi_{f\_type,s\_location,s\_capacity}(feed \bowtie storage)$$

3. List how many of animals have in each type and in each gender.

   $$\pi_{a.a\_type,a.a\_gender,count(a.a\_name)}(\rho_a(animal))$$

4. List how many type of qualification have each tender (name and id).

   $$\pi_{t\_ID,t\_name,\rho_{number\_of\_qualification}(count(*))}(tender \bowtie have)$$

5. List all tenders name with how many animal in each type they tend, except parrots and penguins, you can list parrot or penguin tender if he tend another type

   $$\pi_{t\_name,t\_type,\rho_{countAnimal}(count(*))}(tender \bowtie \sigma_{a\_tpye!='parrot' \text{ or } a\_tpye!='penguin'}(tend))$$

6. List how many qualified tender can tend each type of animal

   $$\pi_{q.q\_what\_animal,\rho_{number\_of\_tender}[count(t\_ID)]}(have \bowtie \rho_q(qualification))$$

7. List all animal names and types which weight is more than all of the tiger and lions weights

   $$\pi_{a\_name,a\_type,a\_weight}[\sigma_{a\_weight>\pi_{a\_weight}[\sigma_{a\_tpye='lion' or\ a\_tpye='tiger'}(animal)]}(animal)]$$

8. Select all animals, except elephants, who's weight is more than the avarage weight, except the elephantsweights

   $$\pi_{a\_name,a\_type,a\_weight}[\sigma_{a\_tpye!='elephant' \text{ and } a\_weight>\pi_{avg(a.a\_weight)}[\sigma_{a.a\_tpye!='elephant'}(\rho_a(animal))]}(animal)]$$

9. Select all storage unit where the used amount in each unit is under 1/3 of the avarage maximum capacity of the units.

   $$\pi_{s\_location,sum(f\_amount)}\left[\sigma_{sum(f\_amount)<\pi_{\frac{avg(s\_capacity)}{3}}(storage)}(feed)\right]$$

10. List all storage unit with used and free capacity in it.

    $$\pi_{f.s\_location,\rho_{used\_amount}(f.f\_amount),\rho_{free\_capacity}(s.s\_capacity-sum(f.f\_amount))}[\rho_f(feed) \bowtie \rho_s(storage)]$$

8. Normalization:

Almost all the table in my database, except some like animal, are in all normal form, and Boyce Codd normal form too. The reason for that, is if my table has more than one primary key, there will be no more columns in the table, only primary keys, so all the normal forms are completed, becasue there are no functional dependencies. If the table has one primary key, all column dependent on the primary key, which is a super key as well, so it is complete all the normal forms and also the BCNF too. In „event" table, there are two primary keys, but the thirds column is dependent from both „e_name" and „e_time" too, so there are no functional dependencies.

For example:

- FEED(**f_type**, f_amount, f_need_cooling, s_location)
  f_type → f_amount,
  f_type → f_need_cooling,
  f_type → s_location
- QUALIFICATION(**q_what_animal**, q_degree)
  q_what_animal → q_degree
- EVENT(**e_name**, **e_time**, e_where)
  e_name, e_time → e_where
- YARD(**y_ID**, y_size, y_number_limit, b_type)
  y_ID → y_size,
  y_ID → y_number_limit,
  y_ID → b_type

One exception is the „animal" table. We need here decomposition rule to get all normal forms completed. I use letters for better code follow instead of column names.

R(A,B,C,D,E,G,H) ~ ANIMAL(**a_name**, **a_type**, a_age, a_gender, a_color, a_weight, y_ID)

a_name → a_gender,	A → D
a_type → a_color,	B → E
a_type, a_gender → a_age,	BD → C
a_type, a_age, a_gender → a_weight,	BCD → G
a_type → y_ID	B → H

CK: $\{A,B\}^+$
1NF ✓, 2NF is not good (e.g: A→D or B→E is partial dependency) so decomposition

R_1(A,D) $F_1\{A\rightarrow D\}$, CK:$\{B,D\}^+$
1NF ✓, 2NF ✓, BCNF ✓ (because A is also a SK)

R_2(B,C,D,G,H) $F_2\{B\rightarrow E, B\rightarrow H, BD\rightarrow C, BCD\rightarrow G\}$, CK:$\{B,D\}^+$
1NF ✓, 2NF is not good again (e.g: B→E or B→H), decompostion again.

R_{21}(B,E,H) F_{21}(B→E, B→H), CK: $\{B\}^+$
1NF ✓, 2NF ✓, BCNF ✓ (B is also a SK)

R_{22}(B,C,D,G) F_{22}(BD→C, BCD→G) , CK:$\{B,D\}^+$
1NF ✓, 2NF ✓ (no partial dependencies), BCNF ✓ (BD is also a SK)

So every table now completed all the normal forms and the result is:

R_1(A,D) F_1{A→D}
R_{21}(B,E,H) F_{21}(B→E, B→H)
R_{22}(B,C,D,G) F_{22}(BD→C, BCD→G)

9. Triggers:

In my trigger I collect all the changes on the feed table, and log into the file which food's amount has been changed after all insert, update or delete and log into the feed_logs table. If we list all rows from the feed_logs, we can see what has been changed from the first time.

```
DROP TABLE feed_logs CASCADE CONSTRAINTS;

CREATE TABLE feed_logs
(
    log_time TIMESTAMP PRIMARY KEY,
    log_info VARCHAR2(40),
    f_type VARCHAR2(100),
    f_amount NUMBER,
    FOREIGN KEY(f_type) REFERENCES feed(f_type)
);

CREATE OR REPLACE TRIGGER log_ins_upd_del_feed
    AFTER INSERT OR UPDATE OR DELETE
    ON feed
    FOR EACH ROW
BEGIN
    CASE
        WHEN INSERTING THEN
            INSERT INTO feed_logs VALUES
                (systimestamp, 'INSERT', :NEW.f_type, :NEW.f_amount);
        WHEN UPDATING THEN
            INSERT INTO feed_logs VALUES
                (systimestamp, 'UPDATE', :NEW.f_type, :NEW.f_amount);
        WHEN DELETING THEN
            INSERT INTO feed_logs VALUES
                (systimestamp, 'DELETE', :NEW.f_type, :NEW.f_amount);
    END CASE;
```