Data management lecture 3

Cardinalities

- `1 -*` , one to many
- `*-1` , many to one
- `*-*` , many to many
- `1-1` , one to one
- `0..1-1` , zero/one to one

Unnormalized form (UNF)

example of unnormalized table:

Customer Name	Item	Shipping Address	Newsletter	Supplier	Supplier Phone	Price
Jens Bergholm	XBOX One	Mølletoften 20, 7100 Vejle	Xbox News	Microsft	12341234	1800
Dennis Jørgensen	Playstation 4	Vinkelvej 167, 7100 Vejle	Playstation news	Sony	23452345	1900
Hans Jensen	XBOX One, PS Vita	Kongensgade 45, 5000 Odense	Xbox news, playstation news	Wholesale	00000000	3300
Anne Johansen	Playstation 4	Vinkevej 167, 7100 Vejle	Playstation news	Sony	23452345	1900

The table above is valuable for:

- insertion anomalies
- update anomalies
- deletion anomalies

1st Normal Form (NF1)

rules of NF1:

- Each column must contain atomic values
  - Only allowed to describe one thing at the time
  - values like `x,y` violates NF1
- A column should contain values of the same type
- Each column name must be unique
- Each row must be unique

table of first normal form

The table below has been normalized to NF1:

ID	Name	Item	Address	Zip	City	Newsletter	Supplie
1	Jens Bergholm	XBOX One	Mølletoften 20	7100	Vejle	Xbox News	Microsft
2	Dennis Jørgensen	Playstation 4	Vinkelvej 167	7100	Vejle	Playstation News	Sony
3	Hans Jensen	XBOX One	Kongensgrade 45	5000	Odense	Xbox News	Microsft
3	Hans Jensen	PS Vita	Kongensgrade 45	5000	Odense	Playstation News	Sony
4	Anne Johansen	Playstation 4	Vinkelvej 167	7100	Vejle	Playstation News	Sony

Functional Dependencies

- Strong Connection between two attributes in a table
  - denoted as `A -> B`
- A functionally determines B or B is functionally dependent on A
  - Name and Item is the Determinant
- `Name: -> {Address, City, Newsletter}`
- `Zip -> City`
- `Item -> {Supplier, Phone, Price}`

Partial Dependency

- An xbox one purchase does not require you to be Jens Bergholm
- But there is an decency as "Jens Bergholm" purchased this item

2nd Normal Form (NF2)

- Since a customer can buy multiple items, and multiple items can also be purchased by multiple users we have a many to many relationship between `Name` and `Item`
  - Which will result in a new table being created named `Order table`

The tables below have been normalized to NF2

Customer Table:

ID	Name	Address	Zip	City	Newsletter
1	Jens Bergholm	Mølletoften 20	7100	Vejle	Xbox News
2	Dennis Jørgensen	Vinkelvej 167	7100	Vejle	Playstation News
3	Hans Jensen	Kongensgrade 45	5000	Odense	Xbox News
3	Hans Jensen	Kongensgrade 45	5000	Odense	Playstation News
4	Anne Johansen	Vinkelvej 167	7100	Vejle	Playstation News

Products table:

ID	Name	Supplier	Phone	Price
1	XBOX Oone	Microsft	12341234	1800
2	Playstation 4	Sony	23452345	1900
3	PS Vita	Sony	23452345	1500

Orders table:

Customer ID	Product ID
1	1
2	2
3	1

Customer ID	Product ID
3	3
4	2

Third Normal Form (NF3)

- A table is said to be in NF if and only if:
    - The table is in NF2
    - Every attribute in the table that do not belong to a candidate key should depend on every candidate key of that table
        - `candidate key` : a column or a combination of columns that uniquely identifies each row in a table

Tables below have been normalized to NF3

ID	Name	Address	Zip	City	Newsletter
1	Jens Bergholm	Mølletoften 20	7100	Vejle	Xbox News
2	Dennis Jørgensen	Vinkelvej 167	7100	Vejle	Playstation News
3	Hans Jensen	Kongensgrade 45	5000	Odense	Xbox News
3	Hans Jensen	Kongensgrade 45	5000	Odense	Playstation News
4	Anne Johansen	Vinkelvej 167	7100	Vejle	Playstation News

Products table:

ID	Name	Phone	Price	Supplier ID
1	XBOX Oone	12341234	1800	1
2	Playstation 4	23452345	1900	2
3	PS Vita	23452345	1500	2

Suppliers table:

ID	Name	Phone
1	Microsoft	12341234
2	Sony	23452345

Orders table:

Customer ID	Product ID
1	1
2	2
3	1
3	3
4	2

Boyce-Codd Normal Form (BCNF)

- a table supports BCNF if:
    - the table is in NF3
    - If a relational schema is in BCNF then all it's redundancy based on functional dependencies has been removed, although other types of redundancy may still exist

Tables below supports bcnf

ID	Name	Address	Newsletter	Zip ID
1	Jens Bergholm	Mølletoften 20	Xbox News	7100
2	Dennis Jørgensen	Vinkelvej 167	Playstation News	7100
3	Hans Jensen	Kongensgrade 45	Xbox News	5000
3	Hans Jensen	Kongensgrade 45	Playstation News	5000
4	Anne Johansen	Vinkelvej 167	Playstation News	7100

zip table:

zip	name
7100	Vejle
5000	Odense

Products table:

ID	Name	Phone	Price	Supplier ID
1	XBOX Oone	12341234	1800	1

ID	Name	Phone	Price	Supplier ID
2	Playstation 4	23452345	1900	2
3	PS Vita	23452345	1500	2

Suppliers table:

ID	Name	Phone
1	Microsoft	12341234
2	Sony	23452345

Orders table:

Customer ID	Product ID
1	1
2	2
3	1
3	3
4	2

4th Normal Form (NF4)

- All columns can be determined only by the key in the table and no other column
- Table must support BCNF
- No Multi-valued Dependencies

Tables below supports NF4

Newsletters table:

ID	Newsletter
1	Xbox News
2	PlayStation News

Subscription table:

Newsletter ID	Customer ID

Newsletter ID	Customer ID
1	1
2	2
1	3
2	3
2	4

Customer Table:

ID	Name	Address	Zip ID
1	Jens Bergholm	Mølletoften 20	7100
2	Dennis Jørgensen	Vinkelvej 167	7100
3	Hans Jensen	Kongensgrade 45	5000
4	Anne Johansen	Vinkelvej 167	7100

zip table:

zip	name
7100	Vejle
5000	Odense

Products table:

ID	Name	Phone	Price	Supplier ID
1	XBOX Oone	12341234	1800	1
2	Playstation 4	23452345	1900	2
3	PS Vita	23452345	1500	2

Suppliers table:

ID	Name	Phone
1	Microsoft	12341234
2	Sony	23452345

Orders table:

Customer ID	Product ID
1	1
2	2
3	1
3	3
4	2

Exercise

```
create type room_types as enum ('Office', 'Normal', 'Two Beds', 'Special');

create table if not exists Position(
    id serial not null primary key,
    designation varchar not null,
    charges_per_hour int not null);

create table if not exists  Employee(
    id serial not null primary key,
    name varchar not null,
    phone varchar not null,
    postition_id int not null references Position(id));


create table if not exists RoomAddress(
    id serial not null primary key ,
    name varchar not null,
    employee_id int not null references Employee(id));

create table if not exists Department(
    id serial not null primary key,
    name varchar not null,
    employee_id int not null references Employee(id));

--===============[ Inserting employee stuff below ]===============--

insert into Position(designation, charges_per_hour) VALUES ('Professor', 5000), ('As

insert into Employee(name, phone, postition_id) VALUES
    ('Dr. Peterson', '12341234', 1),
    ('Dr. Jensen', '23452345', 1),
    ('Dr. Poetch', '34563456', 2),
    ('Dr. Neurenheim', '45674567', 2);
```

```
insert into RoomAddress(name, employee_id) VALUES
    ('U45', 1),
    ('U32', 2),
    ('U186', 3),
    ('U150', 4);

insert into department(name, employee_id) VALUES
    ('Neurology', 1),
    ('Orthopedic', 2),
    ('ENT/Neurology', 3),
    ('SKin/Orthopedic', 4);

--===============[ Patient stuff below ]===============--

create table if not exists Patients (
    id serial not null primary key ,
    name varchar, cpr varchar(10) not null unique,
    phone varchar not null);

create table if not exists RoomType(
    id serial not null primary key,
    room_t room_types);

create table if not exists Beds(
    id serial not null primary key,
    bed_number varchar(3) not null);

create table if not exists Rooms(
    id serial not null primary key,
    name varchar,
    room_type_id int references RoomType(id));



create table if not exists Appointment(
    patient_id int not null references Patients(id),
    employee_id int not null references Employee(id),
    primary key (patient_id, employee_id));

create table Admission(
    room_id int not null references Rooms(id),
    bed_id int not null references Beds(id)) inherits(Appointment);


--===============[ Inserting Patient stuff below ]===============--

insert into patients (name, cpr, phone ) VALUES ('Jan', '190582-1113', '98769876');
insert into patients (name, cpr, phone ) VALUES ('Peter', '300175-2359', '87658765')
insert into patients (name, cpr, phone ) VALUES ('Jens', '041298-1257', '76547654');
insert into patients (name, cpr, phone ) VALUES ('Ole', '051165-9863', '65436543');
insert into patients (name, cpr, phone ) VALUES ('Anna', '260792-1050', '54325432');
insert into patients (name, cpr, phone ) VALUES ('Dennis', '150893-1151', '43214321'
insert into patients (name, cpr, phone ) VALUES ('Ahmed', '010211-7853', '32103210')
insert into patients (name, cpr, phone ) VALUES ('Annika', '051285-8072', '21092109'

insert into RoomType(room_t) values
    ('Office'),
    ('Normal'),
    ('Two Beds'),
    ('Special');

INSERT INTO rooms (name, room_type_id) VALUES ('R2', 2);
INSERT INTO rooms (name, room_type_id) VALUES ('R4', 3);
INSERT INTO rooms (name, room_type_id) VALUES ('R5', 4);
INSERT INTO rooms (name, room_type_id) VALUES ('R6', 4);

insert into beds(bed_number) values ('B1'), ('B5'), ('B7'), ('B8'), ('B8');

INSERT INTO appointment (patient_id, employee_id) values (3, 1); -- Jens, Dr. Peters
INSERT INTO appointment (patient_id, employee_id) values (7, 4); -- Ahmed, Dr. Neure
INSERT INTO appointment (patient_id, employee_id) values (8, 4); -- Annika, Dr. Neur

INSERT INTO admission (room_id, bed_id, patient_id, employee_id) VALUES (1, 5, 1, 1)
INSERT INTO admission (room_id, bed_id, patient_id, employee_id) VALUES (2, 5, 1, 1)
INSERT INTO admission (room_id, bed_id, patient_id, employee_id) VALUES (4, 5, 1, 2)
INSERT INTO admission (room_id, bed_id, patient_id, employee_id) VALUES (2, 3, 2, 2)
INSERT INTO admission (room_id, bed_id, patient_id, employee_id) VALUES (2, 3, 5, 2)
INSERT INTO admission (room_id, bed_id, patient_id, employee_id) VALUES (3, 4, 1, 3)
INSERT INTO admission (room_id, bed_id, patient_id, employee_id) VALUES (4, 5, 8, 4)


select p.name, p.cpr, r.name, b.bed_number, e.name from Admission ad
    inner join Patients p on ad.patient_id = p.id
    inner join Rooms r on ad.room_id = r.id
    inner join Beds b on ad.bed_id = b.id
    inner join Employee e on ad.employee_id = e.id
    where e.id = 1;
```