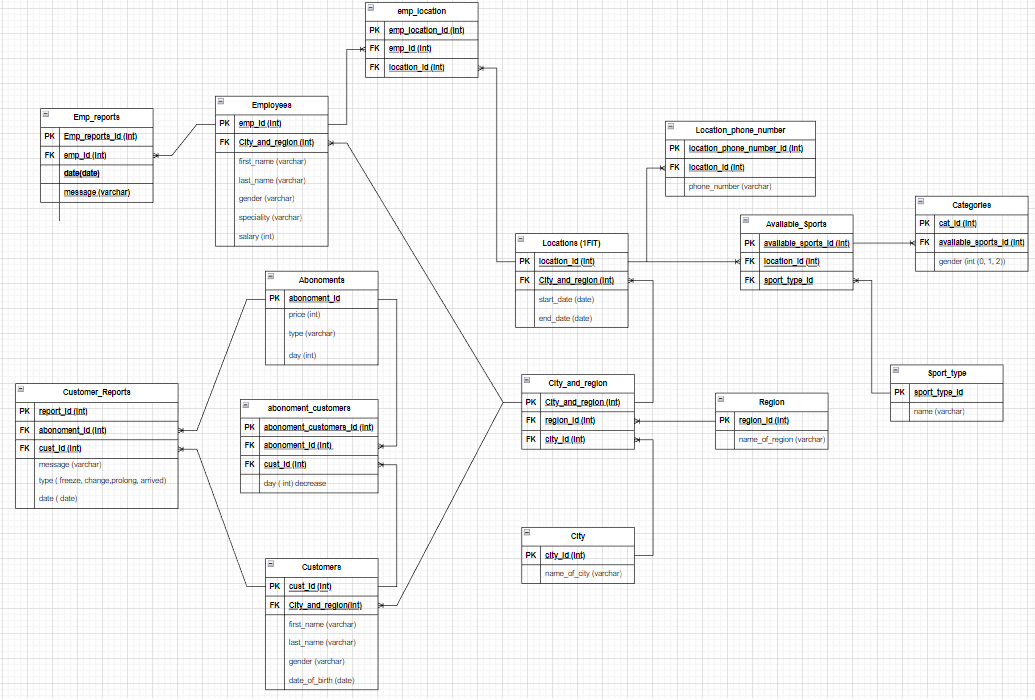
DATABASE 1FIT

Hello everyone. My team and I have chosen the project – 1fit

1fit is a network of gyms

Our ER diagram for database of 1fit is <https://drive.google.com/file/d/1VHRK-fvT0Wo6eBamQG3iucUtgTePvIJz/view?usp=sharing>  
  


Our ER diagram follows to requirments of 1NF, 2NF, 3NF  
1NF - Each table cell should contain a single value. Each record needs to be unique.  
2NF - Be in 1NF. Single Column Primary Key that does not functionally dependant on any subset of candidate key relation  
3NF - Be in 2NF. Has no transitive functional dependencies

**Our creation of tables is following:**

create table City (

city\_id int primary key,

name\_of\_city varchar2(32)

);

create table Region(

region\_id int primary key,

name\_of\_region varchar2(32)

);

create table City\_Region(

city\_region int primary key,

city\_id int references City(city\_id),

region\_id int references Region(region\_id)

);

create table Customers(

cust\_id int primary key,

city\_region int references City\_Region(city\_region),

first\_name varchar2(32),

last\_name varchar2(32),

gender varchar2(32),

date\_of\_birth date

)

create table Locations(

location\_id int primary key,

city\_region references City\_Region(city\_region),

start\_date date,

end\_date date

)

create table Abonoment\_customers(

abonoment\_customers\_id int primary key,

abonoment\_id int references Abonoments(abonoment\_id),

cust\_id int references Customers(cust\_id),

day int

)

create table Abonoments(

abonoment\_id int primary key,

price int,

type varchar2(32),

day int

)

create table Location\_phone\_number(

location\_phone\_number\_id int primary key,

location\_id int references Locations(location\_id),

phone\_number varchar2(32)

)

create table Reports(

report\_id int primary key,

abonoment\_id int references Abonoments(abonoment\_id),

cust\_id int references Customers(cust\_id),

message varchar2(32),

type varchar2(32),

dates date

)

create table Employees(

emp\_id int primary key,

city\_region int references City\_region(city\_region),

first\_name varchar2(32),

last\_name varchar2(32),

gender varchar2(32),

speciality varchar2(32),

salary int

)

create table Emp\_reports(

emp\_reports\_id int primary key,

emp\_id int references Employees(emp\_id),

dates date,

message varchar2(32)

)

create table Emp\_location(

emp\_location\_id int primary key,

emp\_id int references Employees(emp\_id),

location\_id int references Locations(location\_id)

)

create table Sport\_type(

sport\_type\_id int primary key,

sport\_type\_name varchar2(32)

)

create table Available\_sports(

available\_sports\_id int primary key,

location\_id int references Locations(location\_id),

sport\_type\_id int references Sport\_type(sport\_type\_id)

)

create table Categories(

cat\_id int primary key,

available\_sports\_id int references Available\_sports(available\_sports\_id),

gender varchar2(32)

)

**TRIGGERS**

create or replace trigger custReportArrive

before insert on reports

for each row

begin

if :new.type like 'arrived' then

update Abonoment\_customers

set day = day-1

where cust\_id = :new.cust\_id;

end if;

end;

create or replace trigger custReportProlong

before insert on reports

for each row

declare

temp int;

begin

select day into temp

from abonoments

where abonoment\_id = :new.abonoment\_id;

if :new.type like 'prolong' then

update Abonoment\_customers

set day = temp

where cust\_id = :new.cust\_id;

end if;

end;

create or replace trigger custReportChange

before insert on reports

for each row

declare

temp int;

begin

select day into temp from abonoments where abonoment\_id = :new.abonoment\_id;

if :new.type like 'change' then

update Abonoment\_customers

set day = temp

where cust\_id = :new.cust\_id;

end if;

end;

create or replace trigger CustNum

before insert on customers

for each row

declare

temp int;

begin

select count(\*) into temp from customers;

dbms\_output.put\_line('current number of rows: '||temp);

end;

create or replace trigger reportNum

before insert on reports

for each row

declare

temp int;

begin

select count(\*) into temp from reports;

dbms\_output.put\_line('current number of rows: '||temp);

end;

create or replace trigger employeeNum

before insert on employees

for each row

declare

temp int;

begin

select count(\*) into temp from employees;

dbms\_output.put\_line('current number of rows: '||temp);

end;

**EXCEPTIONS**

declare

c\_name customers.first\_name%type := 'Abay';

c\_lname customers.last\_name%type := 'Baxa';

symbol\_invalid exception;

begin

if c\_name like '%@%' or c\_lname like '%@%' then

raise symbol\_invalid;

else

dbms\_output.put\_line('Registration successful.');

end if;

exception

when symbol\_invalid then

dbms\_output.put\_line('First name or last name has invalid symbol');

when others then

dbms\_output.put\_line('Error!');

end;

declare

c\_lname customers.first\_name%type := 'Abay';

less\_symbol\_invalid exception;

begin

if c\_lname not like '%\_\_\_\_%' then

raise less\_symbol\_invalid;

else

dbms\_output.put\_line('Registration successful.');

end if;

exception

when less\_symbol\_invalid then

dbms\_output.put\_line('Less than 4 characters');

when others then

dbms\_output.put\_line('Error!');

end;

declare

c\_id customers.cust\_id%type;

c\_name customers.first\_name%type;

c\_lname customers.last\_name%type;

begin

select first\_name, last\_name into c\_name, c\_lname

from customers

where cust\_id = c\_id;

dbms\_output.put\_line('Name: '|| c\_name);

dbms\_output.put\_line('Last name: '|| c\_lname);

exception

when no\_data\_found then

dbms\_output.put\_line('No such customer!');

when others then

dbms\_output.put\_line('Error');

end;

declare

city\_name city.name\_of\_city%type := 'A';

city\_invalid exception;

begin

if city\_name is null then

raise city\_invalid;

else

select name\_of\_city into city\_name

from city

where name\_of\_city = city\_name;

end if;

exception

when city\_invalid then

dbms\_output.put\_line('Fill in the fields');

when others then

dbms\_output.put\_line('Successfully');

dbms\_output.put\_line('Name of city: '|| city\_name);

end;

**PROCEDURES**

1)create or replace procedure countOfSection(c1 in out sys\_refcursor)

as

begin

open c1 for

select sport\_type.sport\_type\_name, count(\*), locations.city\_region from locations

join available\_sports on locations.location\_id = available\_sports.location\_id

join sport\_type on sport\_type.sport\_type\_id = available\_sports.sport\_type\_id

group by locations.city\_region, sport\_type.sport\_type\_name;

DBMS\_SQL.RETURN\_RESULT(c1);

end;

variable cursor\_output refcursor

execute countOfSection(:cursor\_output);

2)create or replace procedure updateSalary(c1 in out sys\_refcursor)

as

avg\_man int;

avg\_women int;

countOfUpdates int;

begin

open c1 for

select avg(salary) into avg\_women from Employees

where Employees.gender = 'women';

select avg(salary) into avg\_man from Employees

where Employees.gender = 'man';

if avg\_women != avg\_man then

if avg\_women > avg\_man then

update employees set salary = (salary + salary \* 0.05) where employees.gender = 'man';

elsif avg\_women < avg\_man then

update employees set salary = (salary + salary \* 0.05) where employees.gender = 'women';

end if;

end if;

countOfUpdates := SQL%ROWCOUNT;

dbms\_output.put\_line(countOfUpdates);

end;

variable cursor\_output refcursor

execute updateSalary(:cursor\_output);

**FUNCTIONS**

1)CREATE OR REPLACE FUNCTION avg\_price\_of\_customers(city\_name in city.name\_of\_city%type, region\_name in region.name\_of\_region%type)

return int

is

avg\_price int;

temp\_city\_id int;

temp\_region\_id int;

temp\_city\_and\_region\_id int;

BEGIN

select city\_id into temp\_city\_id

from City

where name\_of\_city = city\_name;

select region\_id into temp\_region\_id

from Region

where name\_of\_region = region\_name;

select c.city\_region into temp\_city\_and\_region\_id

from City\_Region c

where city\_id = temp\_city\_id and region\_id = temp\_region\_id;

avg\_price := 200;

select AVG(price) into avg\_price

from abonoments

where abonoment\_id = (select abonoment\_id

from abonoment\_customers

where cust\_id = (select cust\_id

from customers

where city\_region = temp\_city\_and\_region\_id));

return avg\_price;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No region and city this names.');

end;

2) create or replace function COUNT\_OF\_RECORDS

return number

is

count\_of\_records number;

BEGIN

SELECT COUNT(\*) INTO count\_of\_records

FROM CUSTOMERS

GROUP BY customer\_id;

RETURN count\_of\_records

END;