Record Label Database – Report

Application Description

For this project I chose to implement a database describing a record label. A record label is an organisation which has multiple divisions as subsidiaries, each of which employ people and have artists contracted to them. The artists produce releases which are made by a number of manufacturers.

The database contains tables pertaining to the divisions of the label, its employees, the artists signed to it, the releases and the manufacturers of the releases.

The information contained about the label itself is limited to an ID number, a reference to the CEO of the company and the label name.

Each division contains an ID number and information on its name, location and a reference to the manager.

For each employee, information stored is their PPS number, name, address, position within the company, salary and what division they work in.

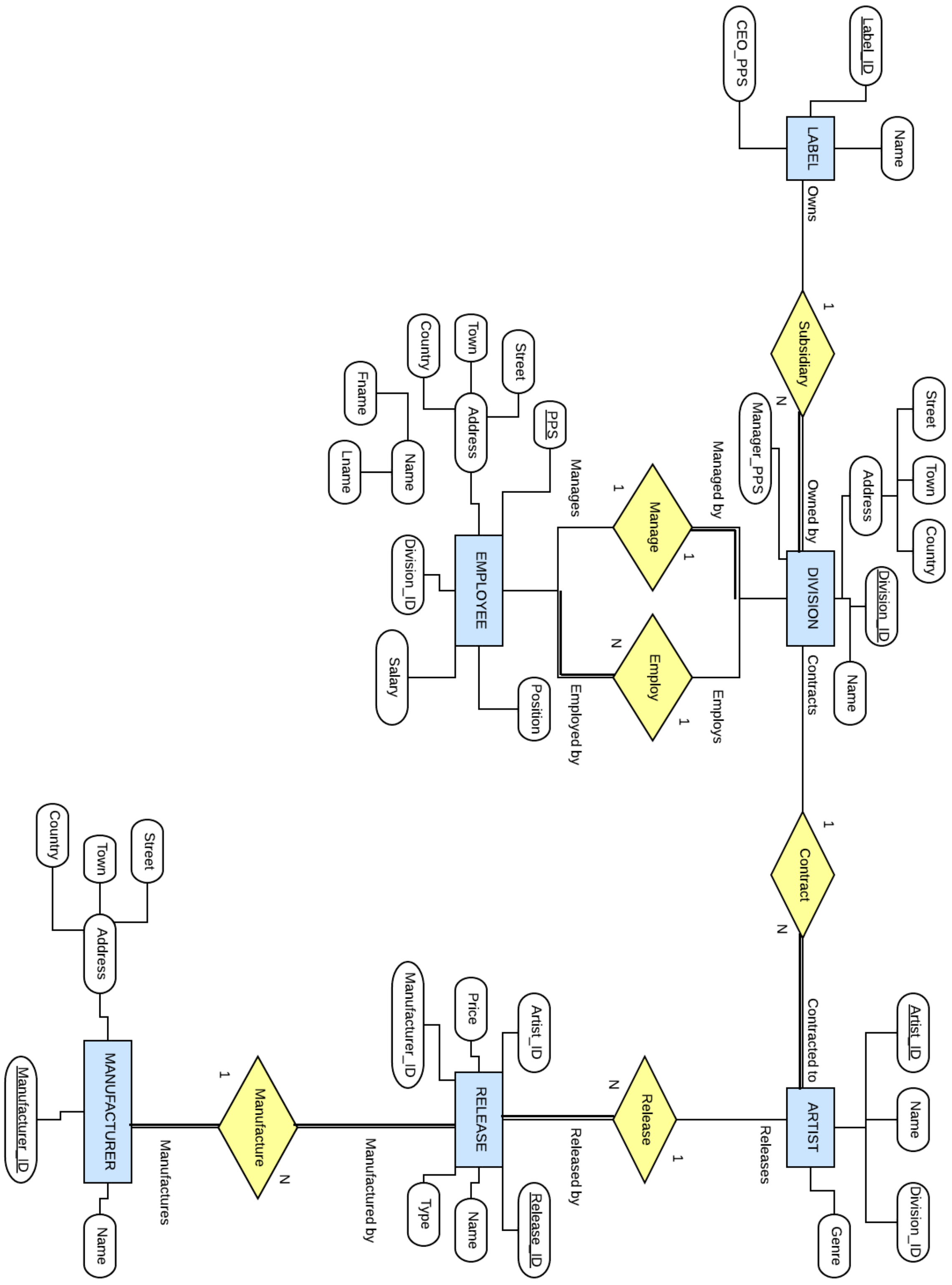
Artists have an ID number, name, genre and a reference to the division they are contracted to.

Releases have an ID number, name, type (album or single) and references to the artist and manufacturer.

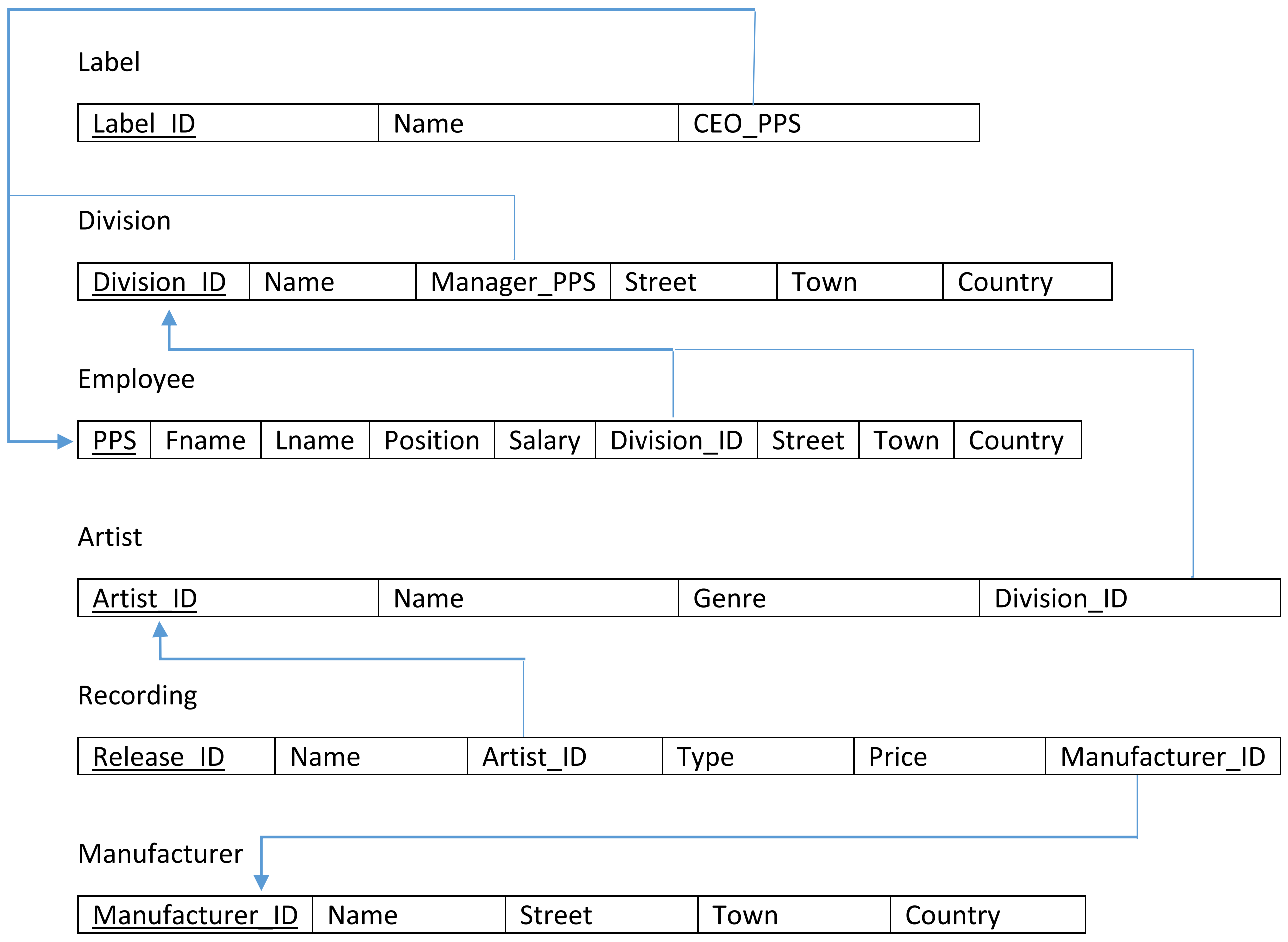
Manufacturers have an ID number, name and address.

The record label I chose to implement is called Play-Tone Records (based on the fictional label in the 1996 Tom Hanks movie "That Thing You Do!"). It has three divisions (Play-Tone Pittsburgh, Play-Tone LA and Play-Tone London), each of which has numerous artists and employees contacted to it, and two manufacturers (RecordCorp and Berlin Audio).

Entity Relationship Diagram



Mapping to a Relationship Schema



Label

Label_ID	Name	CEO_PPS

Division

Division_ID	Name	Manager_PPS	Street	Town	Country

Employee

PPS	Fname	Lname	Position	Salary	Division_ID	Street	Town	Country

Artist

Artist_ID	Name	Genre	Division_ID

Recording

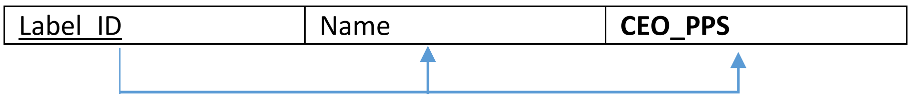
Release_ID	Name	Artist_ID	Type	Price	Manufacturer_ID

Manufacturer

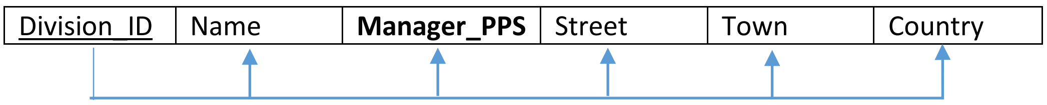
Manufacturer_ID	Name	Street	Town	Country

Functional Dependency Diagram

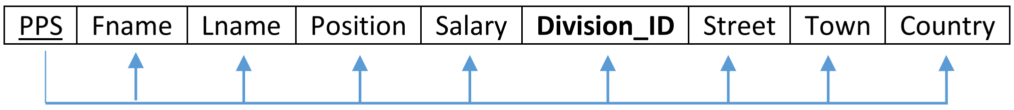
Label



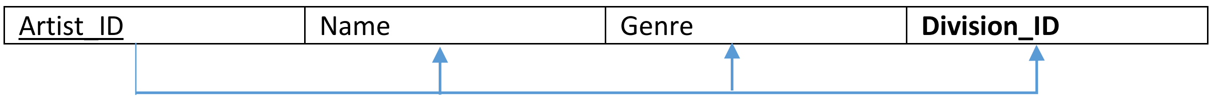
Division



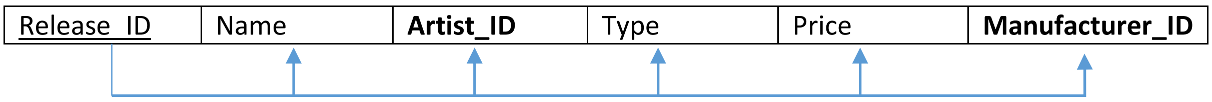
Employee



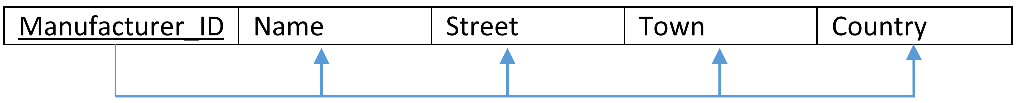
Artist



Recording



Manufacturer



Primary keys are underlined.

Foreign keys are **in bold.**

Normalisation

I found that no normalisation need to be performed on any of the tables, as the schema was already in Boyce-Codd normal form (all attribute values are atomic, every non-key column is dependent on the primary key, no non-key attributes are dependent on another non-key attribute and for all functional dependencies X->Y in the relation R, X is a superkey of R).

Semantic Constraints

There were a few semantic constraints introduced to ensure the integrity of the database. First of all, all values were specified as NOT NULL in order to ensure that all information was contained for each attribute, which was particularly important for primary keys, to ensure that all tuples could be uniquely identified, and for foreign keys, to maintain referential integrity.

Checks were also implemented to maintain data integrity. For example, a PPS number had to be nine digits:

```
CONSTRAINT check_pps CHECK(PPS > 99999999 AND PPS < 1000000000)
```

Other checks made sure that certain string tuples matched a predetermined list of strings:

```
CONSTRAINT check_genre CHECK(Genre IN('Rock','Pop','Dance','Jazz','Classical'))
CONSTRAINT check_type CHECK(Type IN('Single','EP','Album'))
```

Security

Database security could be established by creating various user roles and assigning different levels of access to each role. The roles I create would be as follows:

```
create role managers;
create role sales;
create role aandr;
```

A manager would be granted the required access to add, remove and update employee records, as follows:

```
grant select, insert, delete, update on employee to managers;
```

The sales role would have the same access to the records regarding recordings, and likewise with A&R staff and access to artist records:

```
grant select, insert, delete, update on recording to sales;
grant select, insert, delete, modify on artist to aandr;
```

Finally, a manager may require information on other aspects of how the label is run, so their role was granted SELECT access to the other tables:

```
grant select on label, division, artist, recording, manufacturer to managers;
```

View Creation

Two views were created that made use of table joins to show a SELECT operation across more than one table. The first provides information on the artists signed to the label and uses information from the Artist and Division tables to show the artists' name, genre and division name:

```
create view artist_info (name,genre,division) as select Artist.name, genre,
Division.name from Artist,Division where Artist.division_id = Division.division_id;
```

The second provides information in a similar manner on all the releases on the label using information from the Recording, Artist and Manufacturer tables:

```
create view rec_info (name,artist,type,price,manufacturer) as select
Recording.name, Artist.name, Recording.type, Recording.price, Manufacturer.name
from Recording,Artist,Manufacturer where Recording.artist_id = Artist.artist_id and
Recording.manufacturer_id = Manufacturer.manufacturer_id;
```

Appendix

Create

```sql
create table Label (
Label_ID INTEGER NOT NULL,
Name VARCHAR(20) NOT NULL,
CEO_PPS INTEGER NOT NULL,
PRIMARY KEY(Label_ID)
);

create table Division (
Division_ID INTEGER NOT NULL,
Name VARCHAR(20) NOT NULL,
Manager_PPS INTEGER NOT NULL,
Street VARCHAR(20) NOT NULL,
Town VARCHAR(20) NOT NULL,
Country VARCHAR(20) NOT NULL,
PRIMARY KEY(Division_ID)
);

create table Employee (
PPS INTEGER NOT NULL,
Fname VARCHAR(20) NOT NULL,
Lname VARCHAR(20) NOT NULL,
Position VARCHAR(20) NOT NULL,
Salary VARCHAR(20) NOT NULL,
Division_ID INTEGER NOT NULL,
Street VARCHAR(20) NOT NULL,
Town VARCHAR(20) NOT NULL,
Country VARCHAR(20) NOT NULL,
PRIMARY KEY(PPS),
CONSTRAINT check_pps CHECK(PPS > 99999999 AND PPS < 1000000000)
);

create table Artist (
Artist_ID INTEGER NOT NULL,
Name VARCHAR(50) NOT NULL,
Genre VARCHAR(20) NOT NULL,
Division_ID INTEGER NOT NULL,
PRIMARY KEY(Artist_ID),
CONSTRAINT check_genre CHECK(Genre IN('Rock','Pop','Dance','Jazz','Classical'))
);

create table Recording (
Release_ID INTEGER NOT NULL,
Name VARCHAR(50) NOT NULL,
Artist_ID INTEGER NOT NULL,
Type VARCHAR(20) NOT NULL,
Price FLOAT NOT NULL,
Manufacturer_ID INTEGER NOT NULL,
PRIMARY KEY(Release_ID),
CONSTRAINT check_type CHECK(Type IN('Single','EP','Album'))
);

create table Manufacturer (
Manufacturer_ID INTEGER NOT NULL,
Name Varchar(20) NOT NULL,
Street VARCHAR(20) NOT NULL,
Town VARCHAR(20) NOT NULL,
Country VARCHAR(20) NOT NULL,
PRIMARY KEY(Manufacturer_ID)
);
```

Alter

```
alter table Label ADD FOREIGN KEY(CEO_PPS) REFERENCES Employee(PPS);
alter table Division ADD FOREIGN KEY(Manager_PPS) REFERENCES Employee(PPS);
alter table Employee ADD FOREIGN KEY(Division_ID) REFERENCES Division(Division_ID);
alter table Artist ADD FOREIGN KEY(Division_ID) REFERENCES Division(Division_ID);
alter table Recording ADD FOREIGN KEY(Artist_ID) REFERENCES Artist(Artist_ID);
alter table Recording ADD FOREIGN KEY(Manufacturer_ID) REFERENCES
Manufacturer(Manufacturer_ID);
```

Insert

```
insert into Label Values(100000001,'Play-Tone',100000001);

insert into Division Values(100000001,'Play-Tone Pittsburgh',100000002,'Main
Street','Pittsburgh','USA');
insert into Division Values(100000002,'Play-Tone LA',100000003,'Hollywood St','Los
Angeles','USA');
insert into Division Values(100000003,'Play-Tone London',100000004,'Record
Street','London','UK');

insert into Employee
Values(100000001,'Sol','Siler','CEO','500000',100000002,'Sunset Bvd','Los
Angeles','USA');
insert into Employee
Values(100000002,'Andy','White','Manager','100000',100000001,'Leaf
Street','Pittsburgh','USA');
insert into Employee
Values(100000003,'Johnny','Bigshot','Manager','100000',100000002,'Sunset
Strip','Hollywood','USA');
insert into Employee
Values(100000004,'William','Scott','Manager','100000',100000003,'Buckingham
Street','London','UK');
insert into Employee Values(100000005,'Jim','Smith','Sales','30000',100000001,'Grey
Street','Pittsburgh','USA');
insert into Employee Values(100000006,'Jane','Smith','A and
R','45000',100000001,'Blue Street','Pittsburgh','USA');
insert into Employee
Values(100000007,'Rob','Gordon','Sales','30000',100000002,'Green Street','Los
Angeles','USA');
insert into Employee
Values(100000008,'Sandra','Williams','Sales','30000',100000002,'Palm Road','Los
Angeles','USA');
insert into Employee Values(100000009,'Dick','Johnson','A and
R','50000',100000001,'Beech Avenue','Los Angeles','USA');
insert into Employee
Values(100000010,'Barbra','Reilly','Sales','30000',100000003,'Fulham
Street','London','UK');
insert into Employee Values(100000011,'Johnny','Fresh','A and
R','48000',100000003,'Corporate Street','London','UK');
insert into Employee Values(100000012,'Ann','Jones','A and
R','48000',100000003,'Liverpool Street','London','UK');

insert into Artist Values(100000001,'The Wonders','Pop',100000001);
insert into Artist Values(100000002,'Del Paxton','Jazz',100000002);
insert into Artist Values(100000003,'Captain Geech and the Shrimp Shack
Shooters','Pop',100000002);
insert into Artist Values(100000004,'Diane Dane','Pop',100000002);
insert into Artist Values(100000005,'The Wannabees','Rock',100000003);

insert into Recording Values(100000001,'That Thing You
Do',100000001,'Single',5,100000001);
insert into Recording Values(100000002,'Shrimp
Shack',100000003,'Single',3,100000001);
insert into Recording Values(100000003,'Time to
Blow',100000002,'Album',15,100000001);
```

```
insert into Recording Values(100000004,'Wannabees EP',100000005,'EP',7,100000002);

insert into Manufacturer Values(100000001,'RecordCorp','Factory
Street','Tennessee','USA');
insert into Manufacturer Values(100000002,'Berlin
Audio','Musikstrasse','Berlin','Germany');
```