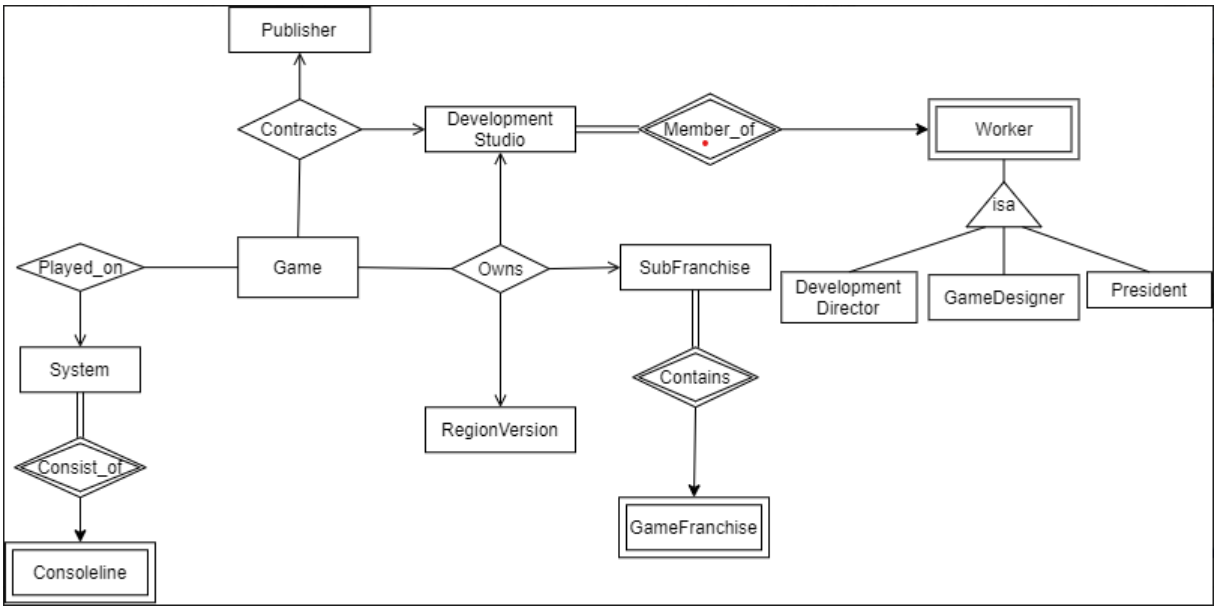
ER-model



Attributes

1. The entity, “Game”, has attributes: GameID, DSID, PublisherID, SystemID, SubFranchiseName, Title, Genre.
2. The entity, “DevelopmentDirector”, has attributes: UserID, Function, Experience.
3. The entity, “GameFranchise”, has attributes: Franchise Name, Release.
4. The entity, “System”, has attributes: SystemID, Name, platform, ConsoloLineID, ReleaseDate, Bitsize.
5. The entity, “ConsoleLine”, has attributes: ConsoloLineID, Name, ReleaseDate, Bitsize.
6. The entity, “DevelopmentStudio”, has attributes: DSID ,Name ,Location, Phone.
7. The entity, “Publisher”, has attributes: PublisherID, Name ,Location.
8. The entity, “SubFranchise”, has attributes: FranchiseName, SubFranchiseName, Release.
9. The entity, “RegionVersion”, has attributes: GameID, RegionName, Release.
10. The entity, “President”, has attributes: UserID, Function, TimeframeOfPresidency.
11. The entity, “GameDesigner”, has attributes: UserID, Function, Experience.
12. The entity, “Worker”, has attributes: DSID, UserID, FirstName, LastName, Street, Phone, DateOfBirth.

Relationships

1. *Played_on* connects each Game to a System, where each game is played on at least one system. This is logical since games are always supported on some sort of system.
2. *Owns* connects every Game to SubFranchise, Development Studio and RegionVersion. Similar to relationship 1. A game owns at least one of each of these entities. These entities are a bit like properties that define a specific game.
3. *Contracts* connects each game to a Publisher and DevelopmentStudio. A game needs to contact a publisher and studio to get produced, but unlike the last 2 relationships is different. Each publisher is contracted to at least one or more multiple games. A game has only one publisher and needs to contract a studio aside from a publisher to start production. In that sense a development studio is contracted to a game and vice versa.

4. *Member_of* connects each DevelopmentStudio is connected to a Worker. Each studio has a specific amount of workers. Each worker is part of max one studio.
5. *Consist_of* connects each System with a ConsoleLine, since each console line exists out of multiple systems or only one.
6. *Contains* connects each SubFranchise with a GameFranchise. A game franchise like Mario exists out of multiple sub-franchises. So that means that each sub-franchise must have a game-franchise.
7. *Isa-hierarchy* is the Worker (parent) that has the children: GameDesigner, President and DevelopmentDirector. Meaning that the children are going to inherit the attributes of the parent.

Cardinality ratios

The described relationships of the er model have cardinality ratios to give a quick overview:

1. 1:n relations: Publisher(1) with Game(n), GameFranchise(1) with SubFranchise(n), ConsoleLine(1) with System(n), SubFranchise(1) with Game(n) and DevelopmentStudio(1) with Worker(n).
2. m:n relations: Game(m) with System(n), Game(m) with RegionVersion(n) and Game(m) with DevelopmentStudio(n)
3. 1:1 relations: Worker(1) with DevelopmentDirector(1), Worker(1) with President(1) and Worker(1) with GameDesigner(1).

Constraints

The open arrows are representing arrows with rounded heads.

<u>Referential constraints</u>.

1. If the entities connected to *Own* excluding Game do not exist , the entity Game would cease to exist.
2. Similarly to situation 1, the entities connected to *Contracts* excluding Game if they do not exist , the entity Game would cease to exist.
3. For the relation *Played_on* it is also the same as situation 1 and 2, but rather with entities Game and System.

<u>Participant constraints</u>:

1. Mandatory: GameFranchise with SubFranchise is mandatory, since each SubFranchise needs a GameFranchise. ConsoleLine with System is mandatory, since each System needs a ConsoleLine. DevelopmentStudio with Worker is mandatory, since each DevelopmentStudio needs a worker. *Played_on*, Contracts and *Owns* do not need the mandatory constraint, since they already have the referential constraint.
2. Disjoint and overlapping: In the er-model the only isa hierarchy present is a disjoint one. This is the case, since a worker cannot have multiple functions (president, director and designer). Thus, there are no overlapping constraints within our ER-model. (overlapping = the parent can be classified into 2 or more childs, disjoint= the parent can only be classified to 1 child out of multiple).