1. Gear(GearName, Cost, ClassId, Weight)

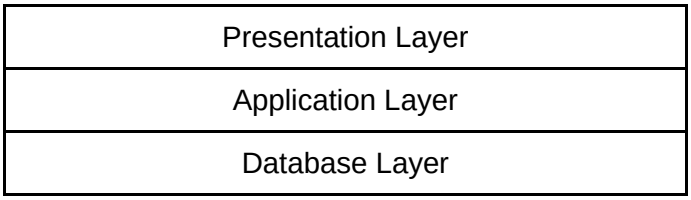
Super Keys:
(GearName)
(Cost)
(ClassId)
(Weight)
(GearName, Cost, ClassId, Weight)
(GearName, Cost, ClassId)
(GearName, Cost, Weight)
(GearName, ClassId, Weight)
(GearName, Cost)
(GearName, ClassId)
(GearName, Weight)
(Cost, ClassId, Weight)
(Cost, Weight)
(Cost, ClassId)
(ClassId, Weight)

Candidate Keys: (ClassId is the most atomic value)
(ClassId)
(ClassId, GearName)
(ClassId, Weight)
(ClassId, Cost)
(ClassId, GearName, Weight)
(ClassId, GearName, Cost)
(ClassId, Weight, Cost)
(ClassId, GearName, Weight, Cost)

2. When a database is in third normal form it means it has already achieved second normal form, has no transitive dependencies, and it is possible to preserve dependencies and ensure lossless decomposition. This means based on our known dependencies, we can write a query and get our desired results. A composition is lossless if replacing a single relation by two results in no loss of data and can be inner joined to create the original relation.

3. This table is not in second normal form since it has partial dependencies with the professor and student information related to IDSt and IDProf keys. This should be split into two separate tables that can be joined together with student and professor information.

4.



Presentation:

- Model View Controller
- How data is presented to a user such as a graphical user interface
- Dependent on the user device
- Receives events, executes actions, and returns the view to the user

Application:

- Business Logic Layer
- A high level view on the data and actions that can be performed on it
- Hides details of data storage
- Often uses object data model

Database Layer

- Data Access Layer
- Interface between business logic layer and the database
- Provides object model mapping from business layer to relational model of database

5.

a) Opening a new connection to the database each time

- Provides more security needed the credentials to be entered each time.
- More computationally expensive requiring an extra step.
- Slower for the programmer to execute actions.

b) Opening one connection to the database

- Less secure no longer requiring credentials when connection is set up.
- Less computationally expensive having the connection constantly open.
- Faster for the programmer to execute actions.

- Larger applications rely on a connections pool that handles authentication, connection timeouts, and security concerns.

6. Results sets should be kept in the DAL and parsed into another data structure because they provide far more information than the business layer needs.
- Meta data such column names, ordering, types, version information, and information about keys and indexes.
- The call should only receive the necessary data and the rest should be abstracted.

7. A partial dependency would be when a column in the relation depends on only some of the columns in the primary key. Candidate keys lay out all the potential primary keys from the list of super keys. It can be seen when a value not in the candidate key only depends on a partial number of the values present, which indicate further revisions are needed for the database design.

8. A transitive dependency is when a column in a table depends on another for uniqueness that is not the primary key. Third normal form aims to eliminate these transitive dependencies and further relations may need to be created if this condition is not satisfied.

9. First Normal Form

- All columns have a unique name
- All columns have a well-defined domain
- All data in a column must be of the same type
- All columns contain one distinct value per row
- The order in which you store data in a table does not matter

10. Holiday Party

BuildingInformation
- BuildingName (varchar(50)) PK
- NumberOfFloors (int)
- NumberOfConferenceRooms (int)

BuildingFloors
- Floor (int)
- ConferenceRoomNumber (int) FK ConferenceRoom
- BuildingName (varchar(50)) PK, FK to BuildingInformation

ConferenceRoom
- RoomNumber (int) PK

Playlist
- PlaylistName (varchar(50)) PK
- SongName (varchar(50))
- ArtistName (varchar(50))

FavoriteSong
- SongName (varchar(50)) PK, FK to Song
- PersonName (varchar(50)) FK to EmployeeName

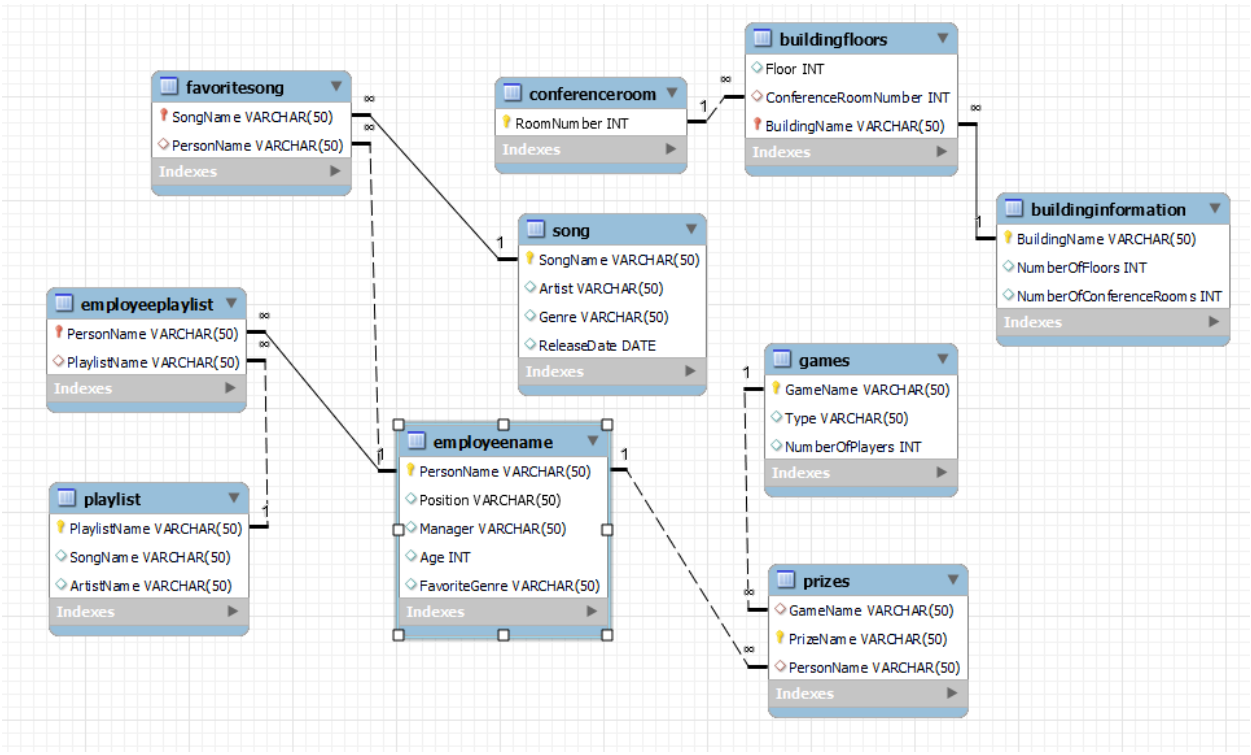
EmployeeName
- PersonName (varchar(50)) PK
- Position (varchar(50))
- Manager (varchar(50))
- Age (int)
- FavoriteGenre (varchar(50))

EmployeePlaylist
- PersonName (varchar(50)) PK, FK to EmployeeName
- PlaylistName (varchar(50)) FK to Playlist

Song
- SongName (varchar(50)) PK
- Artist (varchar(50))
- Genre (varchar(50))
- RelaseDate (date)

Games
- GameName (varchar(50)) PK
- Type (varchar(50))
- NumberOfPlayers (int)

Prizes
- GameName (varchar(50)) FK to Games
- PrizeName (varchar(50)) PK
- PersonName (varchar(50)) FK to EmployeeName


favoritesong
SongName VARCHAR(50)
PersonName VARCHAR(50)
Indexes
conferenceroom
RoomNumber INT
Indexes
buildingfloors
Floor INT
ConferenceRoomNumber INT
BuildingName VARCHAR(50)
Indexes
buildinginformation
BuildingName VARCHAR(50)
NumberOfFloors INT
NumberOfConferenceRooms INT
Indexes
song
SongName VARCHAR(50)
Artist VARCHAR(50)
Genre VARCHAR(50)
ReleaseDate DATE
Indexes
employeeplaylist
PersonName VARCHAR(50)
PlaylistName VARCHAR(50)
Indexes
games
GameName VARCHAR(50)
Type VARCHAR(50)
NumberOfPlayers INT
Indexes
employeename
PersonName VARCHAR(50)
Position VARCHAR(50)
Manager VARCHAR(50)
Age INT
FavoriteGenre VARCHAR(50)
Indexes
playlist
PlaylistName VARCHAR(50)
SongName VARCHAR(50)
ArtistName VARCHAR(50)
Indexes
prizes
GameName VARCHAR(50)
PrizeName VARCHAR(50)
PersonName VARCHAR(50)
Indexes