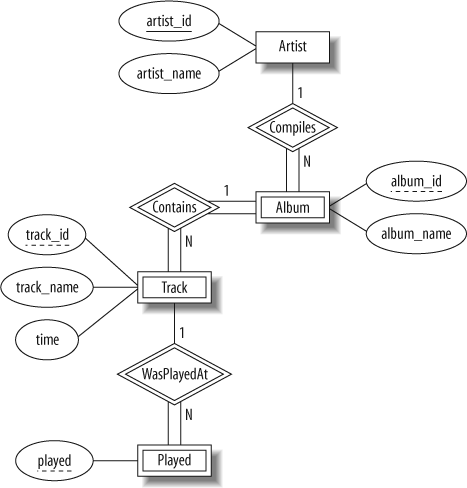
**Báo cáo bài tập tuần 1**

Nhóm thực hiện

* Võ Đăng Quang – 3120410429
* Phạm Minh Quân – 3120410438
* Trần Nguyên Lộc – 3120410297
* Trịnh Hùng Thái – 3120410471

**The Music Database**

Mô hình ERD ban đầu



* Support for compilations or various-artists albums, where each track may be by a different artist and may then have its own associated album-like details such as a recording date and time. Under this model, the album would be a strong entity, with many-to-many relationships between artists and albums.

Theo yêu cầu chưa được thực hiện bên trên thì mối quan hệ Artist và Album có 1 vấn đề cần được giải quyết như sau:

Diagram

Description automatically generated

Quan hệ giữa thực thể Artist và thực thể Album ban đầu

Trong thực tế thì một Album có thể có một hoặc nhiều nghệ sĩ (Artist) cùng tham gia sáng tác. Nhưng trong ERD ban đầu chưa thể hiện được điều đó vì quan hệ giữa Artist và Album là quan hệ 1-n (1 – nhiều), tức là một Artist có thể sáng tác nhiều Album và mỗi Album chỉ có thể sáng tác bởi 1 Artist.

Để xử lí vấn đề trên ta chỉnh sửa lại quan hệ giữa 2 thực thể Artist và Album như sau:

Diagram

Description automatically generated

Quan hệ giữa thực thể Artist và thực thể Album sau khi được chỉnh sửa

Một số thứ được thay đổi trong mối quan hệ giữa Artist và Album như sau:

* Chuyển quan hệ giữa Artist và Album thành quan hệ nhiều – nhiều vì khi đó ta sẽ giải quyết được vấn đề mà ta đã đề cập trước đó. Khi này một nghệ sĩ (Artist) có thể sáng tác nhiều Album và một Album có thể được sáng tác bởi nhiều nghệ sĩ (Artist).
* Khi một quan hệ giữa 2 thực thể là nhiều – nhiều thì mối quan hệ đó sẽ trở thành một thực thể mới mang khoá liên kết của 2 bảng. (Ví dụ bảng A có khoá chính AT và bảng B có khoá chính BT và có quan hệ nhiều – nhiều thì mối quan hệ giữa 2 bảng đó sẽ trở thành một thực thể mới và có cả 2 thuộc tính khoá phụ là AT và BT)
* Bổ sung thuộc tính **date** và **time** cho quan hệ Compiles thể hiện ngày và thời gian sáng tác của Album đó.
* Album sẽ được chuyển từ thực thể yếu (Weak Entity) sang thực thể mạnh (Strong Entity) vì khi này thực thể Album không còn phụ thuộc vào thực thể mạnh nào để tồn tại (Trước đó thực thể Album phải phụ thuộc vào thực thể Artist để tồn tại vì quan hệ **1 – nhiều** giữa Artist và Album, tức là trong bảng dữ liệu Artist phải có dữ liệu thì Album mới tồn tại vì bảng dữ liệu Album phụ thuộc vào dữ liệu đó để khởi tạo dữ liệu mới).
* Playlists, a user-controlled collection of tracks. For example, you might create a playlist of your favorite tracks from an artist.

Theo yêu cầu chưa chưa được thực hiện bên trên thì chúng ta cần phải tạo thêm một Playlist (Một bộ sưu tập nhạc yêu thích của người dùng).

Để xử lí yêu cầu trên thì ta cần tạo một quan hệ nhiều – nhiều giữa Artist và Track như sau:

Diagram

Description automatically generated  
Quan hệ nhiều nhiều giữa thực thể Artist và thực thể Track được thêm vào

Lý do mà ta tạo thêm một quan hệ nhiều – nhiều mang tên Playlist giữa 2 thực thể rất đơn giản.

* Thứ nhất: Vì quan hệ nhiều – nhiều sẽ sinh ra một thực thể mới. Đồng nghĩa ta có thể lưu trữ nhiều dữ liệu bản ghi trùng lặp hơn dựa vào quan hệ khoá chính – khoá ngoại (bên cạnh đó các bản ghi (record) không được phép giống nhau hoàn toàn).
* Thứ hai: Người dùng có thể lưu nhiều bài hát (Track) của nghệ sĩ (Artist) ở một bộ sưu tập nhạc (Playlist) và một bài hát (Track) của nghệ sĩ (Artist) cũng có thể được lưu ở nhiều bộ sưu tập nhạc (Playlist) khác nhau. Ví dụ: Track A, B của Artist A, B lưu ở Playlist A hoặc Track A của Artist A được lưu ở Playlist A, B.

Đồng thời chúng ta phân biệt tên của các bộ sưu tập nhạc (Playlist) dựa theo thuộc tính **playlist\_name.**

* Track ratings, to record your opinion on how good a track is.

Yêu cầu chưa được thực hiện yêu cầu chúng ta xếp hạng bản nhạc(Track) theo độ yêu thích.

Để giải quyết thì đơn giản chỉ thêm vào một thuộc tính cho thực tể Track là **rating**.

Diagram

Description automatically generated  
Thực thể Track trước

Diagram

Description automatically generated

Thực thể Track sau

Thuộc tính **rating** thể hiện xếp loại của từng bản nhạc theo điểm số

* Source details, such as when you bought an album, what media it came on, how much you paid, and so on.

Yêu cầu chưa được thực hiện trong đề đề cập tới việc mua bán giữa người dùng (played) và album, giữa thực thể Played và thực thể Album.

Do đó ta tạo thêm một mối quan hệ gọi là **Sell** giữa 2 thực thể Played và Album và đây là mối quan hệ nhiều – nhiều giữa 2 thực thể. Vì trong thực tế người sử dụng có thể mua nhiều Album và một Album có thể được nhiều người mua.

Diagram

Description automatically generated

Mối quan hệ Sell giữa thực thể Played và Album

Khi một mối quan hệ có quan hệ nhiều – nhiều thì nó sẽ trở thành một thực thể mới, thực thể này sẽ đóng vai trò như cầu nối dữ liệu của cả 2 thực thể Album và Played thông qua quan hệ khoá chính – khoá phụ. Thực thể **Sell** này sẽ có bao gồm 5 thuộc tính:

* played: mã người dùng (khoá phụ tham chiếu tới khoá chính ở bảng Played)
* album\_id: mã album (khoá phụ tham chiếu tới khoá chính ở bảng Album)
* price – thể hiện cho thành tiền mà người dùng trả để mua Album.
* media – thiết bị mà nó phát.
* time – thời gian mua Album đó.
* Album details, such as when and where it was recorded, the producer and label, the band members or sidemen who played on the album, and even its artwork.

Yêu cầu trên đề cập tới việc bổ sung thuộc tính cho Album như producer (nhà sản xuất) và label (nhãn hiệu), date\_time và place\_recorded (thời gian và địa điểm album được thực hiện) bên cạnh đó là thông tin thành viên, vai trò của họ trong album đó

Đối với yêu cầu thêm thuộc tính cho Album thì thông tin của các thuộc tính thể hiện như sau:

* producer: thông tin nhà sản xuất album.
* label: nhãn hiệu của album hoặc của nhà sản xuất.
* date\_time: thời gian và ngày mà album được ghi lại.
* place\_recorded: địa điểm mà album được ghi lại.

Nhưng đối với việc làm rõ thông tin ban nhạc trong album (cụ thể là thông tin mà thành viên sáng tác album và vai trò của họ trong album đó), ta có thể thực hiện bằng cách thay đổi mối quan hệ giữa 2 thực thể **Album** và **Artist** từ quan hệ **1 – nhiều** thành quan hệ **nhiều – nhiều**. Vì khi đó mối quan hệ giữa 2 thực thể sẽ trở thành một thực thể mới, giúp cho việc lưu trữ và truy vấn trở nên dễ dàng hơn.

Diagram

Description automatically generated

Mối quan hệ giữa thực thể Artist và Album ban đầu

Diagram

Description automatically generated

Mối quan hệ giữa thực thể Artist và Album sau khi thay đổi chỉnh sửa

Sau khi thay đổi mối quan hệ của **Compiles** như thế ta sẽ thu được một thực thể mới và thực thể này sẽ đóng vai trò như cầu nối dữ liệu của cả 2 thực thể Album và Artist thông qua quan hệ khoá chính – khoá phụ.

Thực thể **Compiles** bao gồm các thuộc tính như:

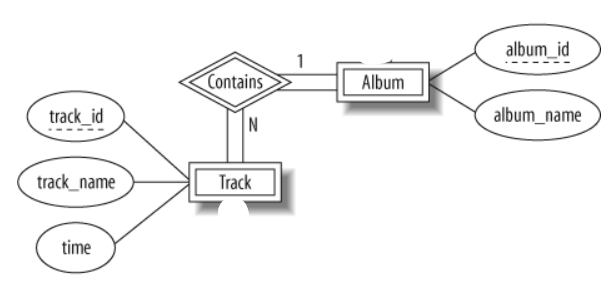
* artist\_id: mã id của artist (khoá phụ tham chiếu tới khoá chính ở bảng Artist)
* album\_id: mã album (khoá phụ tham chiếu tới khoá chính ở bảng Album)
* \*role: vai trò của người đó trong album được chỉ định.

Notes \*: Đối với thuộc tính **role** thì nó là một thuộc tính thể hiện vai trò của 1 Artist trong 1 Album và 1 Artist thì có thể tham gia sáng tác/sản xuất nhiều Album cho nên vai trò của họ trong mỗi Album là có thể khác nhau. Ví dụ như Artist A trong Album A có thể là nhạc chính nhưng trong Album B người đó lại là người nhạc phụ,.v..vv..

* Smarter track management, such as modeling that allows the same track to appear on many albums.

Yêu cầu trên đề cập tới việc quản lí bài nhạc một cách thông minh, cụ thể là một bài nhạc có thể xuất hiện trên nhiều Album khác nhau.

Do đó để xử lí được yêu cầu trên ta đơn giản là thay đổi mối quan hệ của 2 thực thể **Track** và **Album** như sau:



Mối quan hệ giữa thực thể Album và Track ban đầu

Diagram

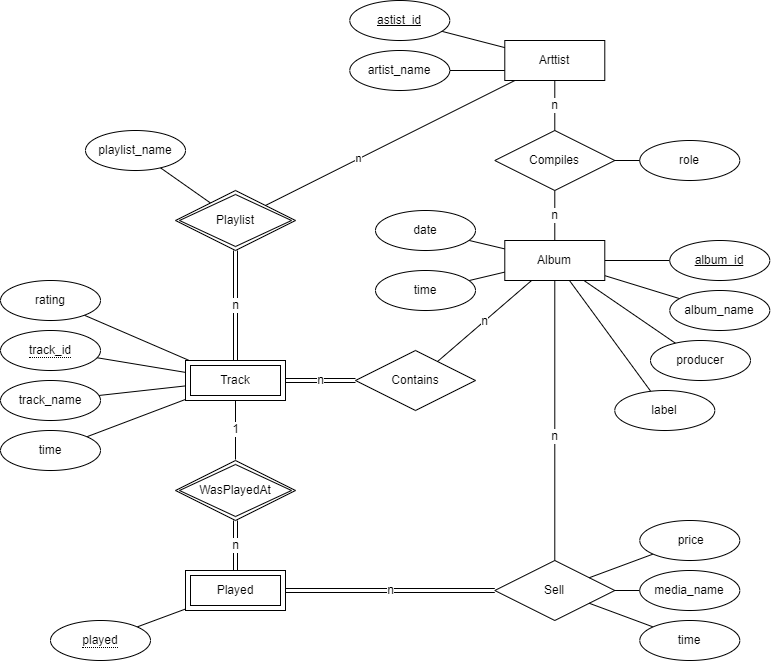
Description automatically generated

Mối quan hệ giữa thực thể Album và Track sau khi thay đổi chỉnh sửa

Việc thay đổi mối quan hệ từ **1 – nhiều** sang quan hệ **nhiều – nhiều** sẽ khiến cho mối quan hệ giữa 2 thực thể trở thành một thực thể mới. Và điều đó đồng nghĩa với việc thực thể mới này đóng vai trò như một kho lưu trữ bài hát (Track) của các Album, nó cho phép một Album có thể lưu trữ nhiều bài hát (Track) và một bài hát (Track) có thể được lưu trữ ở nhiều Album. Thực thể mới này chỉ bao gồm các thuộc tính như:

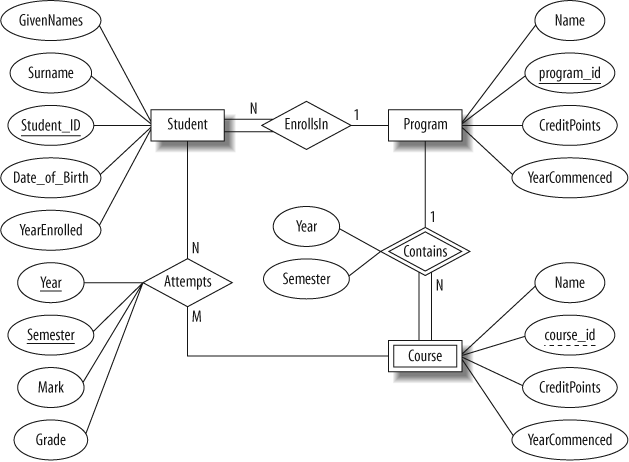
* track\_id: mã bài hát
* album\_id: mã album

Mô hình ERD sau khi thực hiện tất cả các yêu cầu:



**The University Database**

Mô hình ERD ban đầu:



* Our database design is rather simple, but this is because the requirements are simple. For a real university, many more aspects would need to be captured by the database. For example, the requirements don’t mention anything about campus, study mode, course prerequisites, lecturers, timetabling details, address history, financials, or assessment details. The database also doesn’t allow a student to be in more than one degree program, nor does it allow a course to appear as part of different programs.

Chuyển đổi mối quan hệ

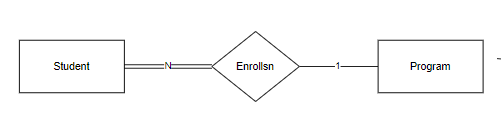
Cách các mối quan hệ được biểu diễn phụ thuộc vào bản chất và mức độ của mối quan hệ

Các bản chất có thể có là: 1:N, N:M

Mức độ là:

* Unary
* Binary
* Ternary…

Binary 1:N



Tổng cộng

Một phần

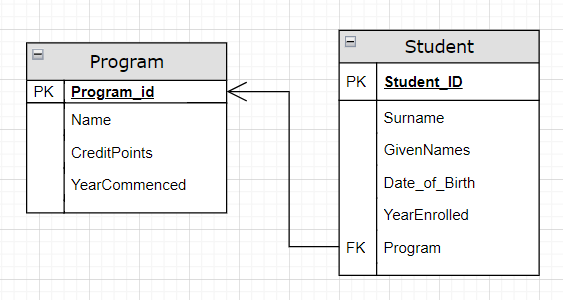
Trường hợp 1:

Theo yêu cầu chưa được thực hiện bên trên thì chúng ta cần phải thêm

Khóa chính của người ghi danh một phần chương trình quan hệ phía “1” của mối quan hệ trở thành khóa ngoại của tổng số người ghi danh trong quan hệ ở phía “N” ở thực thể Student nhằm thể hiện một sinh viên ghi danh duy nhất chương trình cấp bằng đó hay là nhiều sinh viên cùng tham gia duy nhất một chương trình cấp bằng đó

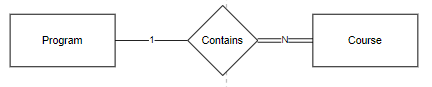
Program (Program\_id, Name, CreditPoints, YearCommenced)

Student (Student\_ID, Surname, GivenNames, Date\_of\_Birth, YearEnrolled, Program)



Binary 1:N

Một phần

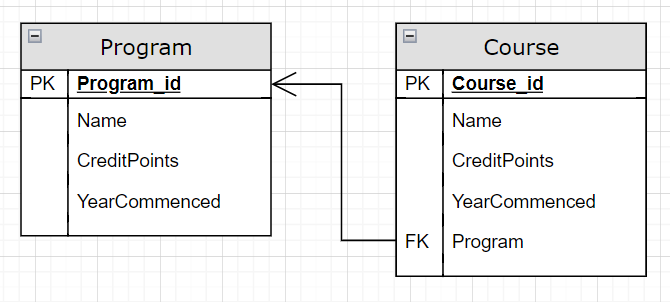


Tổng cộng

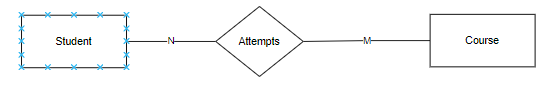
Khóa chính của người ghi danh một phần chương trình quan hệ phía “1” của mối quan hệ trở thành khóa ngoại của tổng số chương trình chứa trong quan hệ ở phía “N” ở thực thể khóa học nhằm thể hiện một hay nhiều khóa học đó chỉ xuất hiện trong đúng một chương trình đó

Program (Program\_id, Name, CreditPoints, YearCommenced)

Course (Course\_id, Name, CredittPoints, YearEnrolled, Program)



Binary N:M



Sinh viên và khóa học có liên quan với nhau thông qua mối quan hệ nỗ lực, một khóa học có thể tồn tại mà không có sinh viên mà ngược lại vì sự tham gia không toàn bộ nên một bảng mới được tạo để đại diện cho mối quan hệ hai thực thể trên

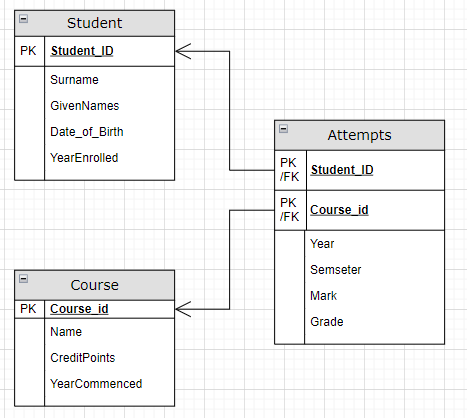
Chứa hai khoá ngoại - một khóa từ mỗi người tham gia trong mỗi quan hệ

Khóa chính của bảng mới là sự kết hợp hai khóa ngoại

Student (Student\_ID, Surname, GivenNames, …)

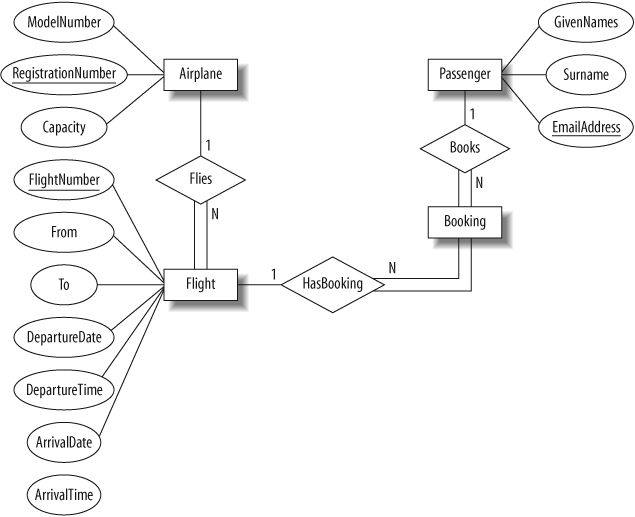
Attempts (Student\_ID, Course\_id, Semseter, Year, Mark, Grade)

Course (Course\_id, Name, CredittPoints, ….)



**The Flight Database**

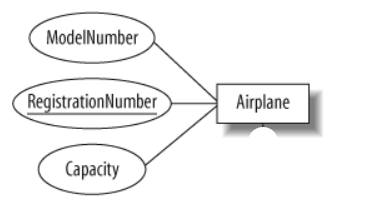
Mô hình ERD ban đầu:



* We’ve treated the capacity of the airplane as an attribute of an individual airplane. If, instead, we assumed that the capacity is determined by the model number, we would have created a new AirplaneModel entity with the attributes ModelNumber and Capacity. The Airplane entity would then not have a Capacity attribute.

Yêu cầu chưa được thực hiện này yêu cầu chúng ta mở thêm một thực thể mới mang tên **AirplaneModel** bao gồm các thuộc tính như **ModelNumber** và **Capacity** và như chúng ta đã biết 2 thuộc tính trên ban đầu là thuộc về thực thể **Airplane**.

=> Vậy ta có thể suy ra yêu cầu của đề bài đang muốn chúng ta mở rộng cho lớp thực thể **Airplane** và 2 thuộc tính đã được tách ra đó sẽ được chuyển sang cho thực thể mới là **AirplaneModel** trong đó thuộc tính **ModelNumber** sẽ đóng vai trò là khoá chính giúp phân biệt các đối tượng dữ liệu với nhau. Thực thể **AirplaneModel** đóng vai trò là thực thể mạnh và thực thể **Airplane** sẽ đóng vai trò là thực thể yếu phụ thuộc vào thực thể mạnh **AirplaneModel** thông qua mối quan hệ **1 – nhiều** giữa 2 thực thể.



Thực thể Airplane ban đầu

Diagram

Description automatically generated

Thực thể Airplane sau khi được mở rộng

* We’ve mapped a different flight number to each flight between two destinations. Airlines typically use a flight number to identify a given flight path and schedule, and they specify the date of the flight independently of the flight number. For example, there is one IR655 flight on April 1, another on April 2, and so on. Different airplanes can operate on the same flight number over time; our model would need to be extended to support this.

Yêu cầu chưa được thực hiện này yêu cầu chúng ta mở rộng và nâng cấp chức năng cho mô hình thực thể ban đầu, mô hình thực thể ban đầu bao gồm:

Diagram

Description automatically generated

Thực thể Flight ban đầu

Ta nhận thấy là thực thể này mang chức năng lưu trữ như một số hiệu bay của từng chuyến bay, tức là nó bao gồm các thông tin như: Thời gian đi, thời gian đến, ngày đi, ngày đến, đi từ đâu, đến đâu và mã số hiệu chuyến bay.

Nhưng mô hình trên cũng có vấn đề như việc lưu trữ các thông tin dữ liệu chưa được tối ưu hoá và hiệu suất cũng không được ổn định, khi ta biểu diễn thực thế đó lên CSDL nó sẽ là 1 bảng có rất nhiều cột và mỗi cột đều có thể lặp đi lặp lại dữ liệu nhiều lần, điều đó sẽ khiến cho dung lượng lưu trữ trong CSDL trở nên rất nặng và việc truy vấn dữ liệu sẽ rất khó khăn khi có nhiều bản ghi (record) trong bảng. Do đó ta cần chuẩn hoá dữ liệu cho thực thể này.

Ta phân tách các thuộc tính cùa thực thể **Flight** ban đầu như sau:

Diagram

Description automatically generated

Thực thể Flight được mở rộng

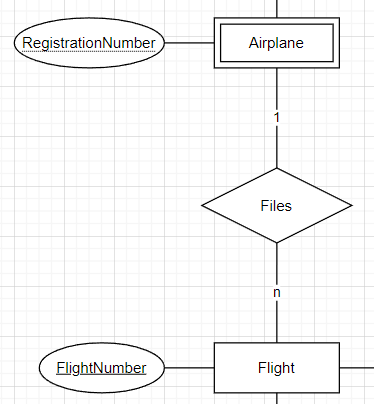
Đối với 2 thuộc tính **From** và **To** ở thực thể **Flight** ban đầu, ta có thể tách chúng thành một thực thể mới mang tên **Location** và thêm một thuộc tính khoá chính **LocationID** để giúp phân biệt giữa các đối tượng dữ liệu. Và lý do ta chọn tách 2 thuộc tính trên thành một thực thể mới là vì trong thực tế địa điểm bay từ chỗ này đến chố khác tại một sân bay là cố định (Ví dụ sân bay Tân Sân Nhất có chuyến bay từ TPHCM – Hà Nội hoặc TPHCM – Đà Nẵng).

Và khi này ta sẽ có bao gồm 2 thực thể chính: **Flight** (Chuyến bay), **Location** (Địa điểm) và giữa 2 thực thể đó ta tạo mối quan hệ **nhiều – nhiều** mang tên **Schedule** (Lịch trình). Lý do chúng có quan hệ n – n là bởi vì trong thực tế số hiệu chuyến bay có thể tái sử dụng lại nhiều lần và vì thế một số hiệu chuyến bay có thể có nhiều địa điểm sân bay và một địa điểm sân bay có thể có nhiều số hiệu chuyến bay đi đến.

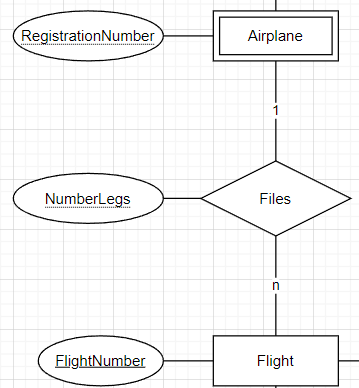
Thực thể mới **Schedule** này biễu diễn thông tin của một lịch trình bay bao gồm như: Số hiệu chuyến bay (FlightNumber), Địa điểm bắt đầu và kết thúc (From, To) và Thời gian đi, Thời gian đến (DepartureTime, ArrivalTime) và Ngày đi, Ngày đến (DepartureDate, ArrivalDate).

* The system also assumes that each leg of a multihop flight has a different FlightNumber. This means that a flight from Dubai to Christchurch via Singapore and Melbourne would need a different FlightNumber for the Dubai-Singapore, Singapore-Melbourne, and Melbourne-Christchurch legs.

Yêu cầu này muốn chúng ta giải quyết bằng cách cung cấp mã chặng bay mỗi lần quá cảnh.



Mối quan hệ giữa Airplane và Flight ban đầu



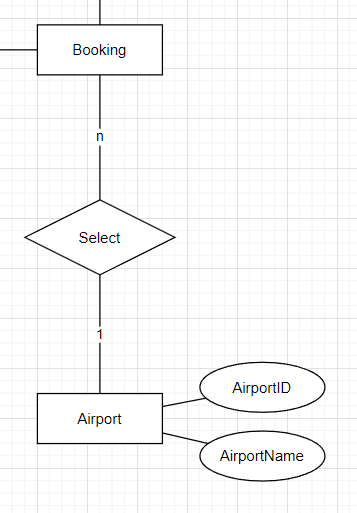
Mối quan hệ giữa Airplane và Flight sau chỉnh sửa

Thuộc tính **NumberLegs** là thuộc tính thể hiện vai trò của mã chặng bay trong một chặng bay mỗi lần quá cảnh tại một sân bay nào đó. Với mỗi lần quá cảnh sẽ có một mã chặng bay khác nhau. Ví dụ như trong chuyến bay từ sân bay Tân Sơn Nhất (TPHCM) – sân bay Nội Bài (Hà Nội) sẽ có những chặng bay quá cảnh tại sân bay Liên Khương (Đà Lạt) và sân bay Đà Nẵng (Đà Nẵng). Như vậy chặng bay: TPHCM – Đà Lạt, Đà Lạt – Đà Nẵng và Đà Nẵng – Hà Nội, mỗi chặng sẽ có một **NumberLegs** khác nhau.

* Our database also has limited ability to describe airports. In practice, each airport has a name, such as “Melbourne Regional Airport,” Mehrabad,” or “Tullamarine.”. The name can be used to differentiate between airports, but most passengers will just use the name of the town or city. This can lead to confusion, when, for example, a passenger could book a flight to Melbourne, Florida, USA, instead of Melbourne, Victoria, Australia. To avoid such problems, the International Air Transport Association (IATA) assigns a unique airport code to each airport; the airport code for Melbourne, Florida, USA is MLB, while the code for Melbourne, Victoria, Australia is MEL. If we were to model the airport as a separate entity, we could use the IATA-assigned airport code as the primary key. Incidentally, there’s an alternative set of airport codes assigned by the International Civil Aviation Organization (ICAO); under this code, Melbourne, Florida is KMLB, and Melbourne, Australia is YMML.

Yêu cầu này muốn chúng ta phải phân biệt sân bay với nhau vì khách hàng thường sẽ gọi sân bay bằng tên của thị trấn hay thành phố của sân bay đó. Có một vài trường hợp hành khách nhầm lẫn sân bay của thành phố này với thành phố khách chỉ vì nó trùng tên với nhau.

Để giải quyết vấn đề này chúng ta thêm một thực thể mới là **Airport** với hai thuộc tinh là **AirportID** và **AirportName**. Trong đó **AirportID** là khoá chính phân biệt giữa các đối tượng dữ liệu khác nhau. VD: sân bay Melbourne (Florida, USA) có mã là MLB trong khi ở Victoria, Australia cũng có sân bay tên Melbourne nhưng lại có mã là MEL.



Thực thể Airport với 2 thuộc tính AirportID và AirportName

Thực thể Airport này có quan hệ 1 – n với thực thể Booking là vì khi ta đặt vé chuyến bay thì ta chỉ có thể chọn 1 sân bay đến trong lượt bay đó và một sân bay thì có thể chọn đặt vé đưược nhiều lần.

Sau khi thực hiện tất cả các yêu cầu, ta được mô hình ERD mới:

