EECS-317 Data Management and Information Processing

Lecture 7 – Relational Database Design

Steve Tarzia

Spring 2019

Northwestern

Announcements

- HW2 was due yesterday
- HW3 posted and due in one week.
- Midterm exam one week from Thursday (May 2nd)
  - Practice exams and additional practice HW questions are posted.

Last Lecture: Combining SELECTs, Adv. predicates

`UNION`, `INTERSECT`, and `EXCEPT`

- Used to combine two `SELECT` statements.
- Combines results table *vertically* (rather than horizontally for JOINs)
- Necessary when answer requires two different (virtual) tables.

- Discussed more advanced uses of predicates.
  - Summing an indicator variable.
- Introduced `CASE` statement which chooses between two different options depending on some condition in the row.

Next topic: Data Modeling

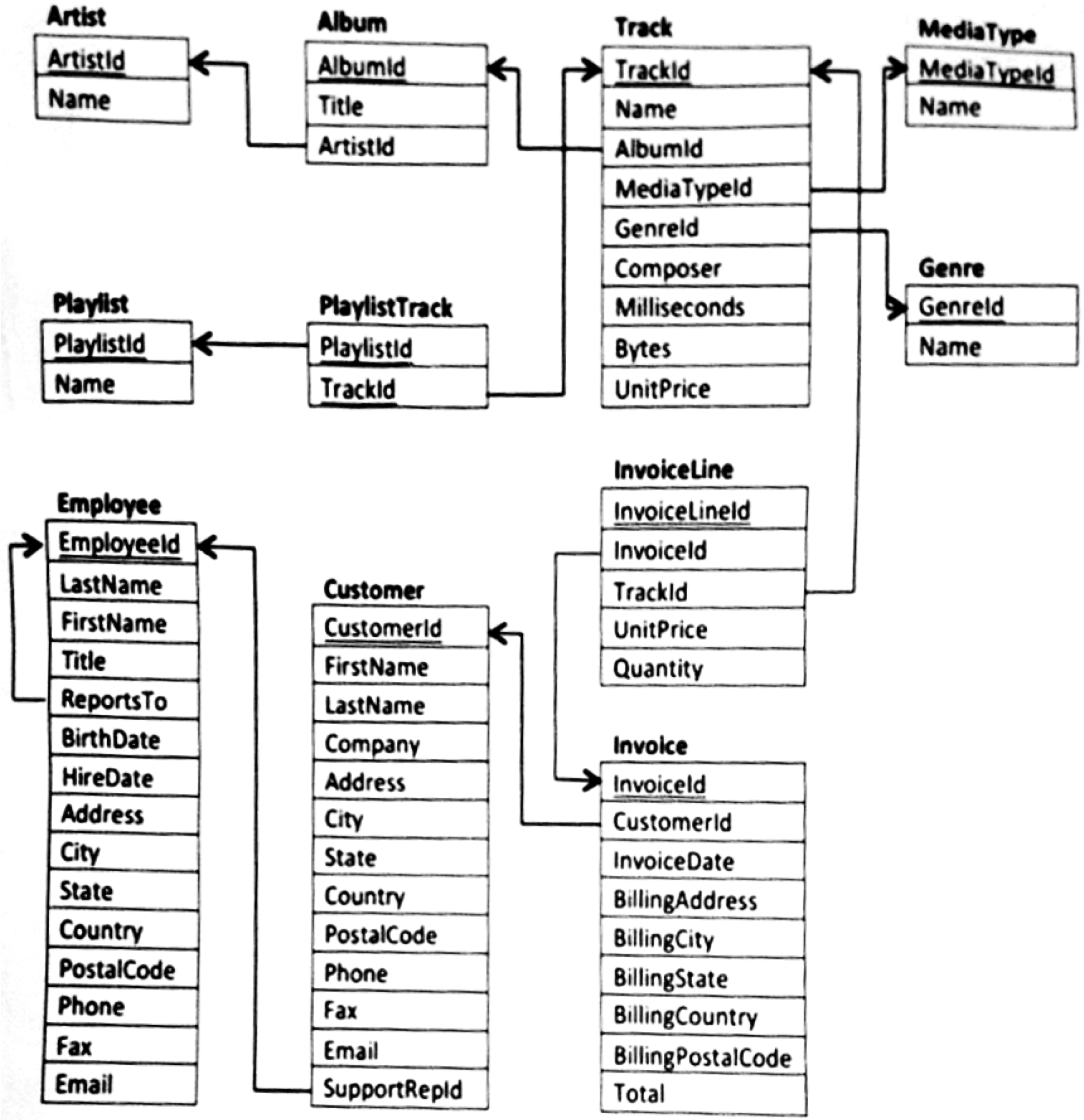
In other words, how to define a relational database schema

Database **Schema** defines the data’s structure

- Also called a data model
  - It’s *metadata* – data about data
- Defines the tables, including:
  - Columns in each table (both the name and *type*)
  - Primary Key for each table
  - Foreign Keys that link tables
  - Unique Keys, if any
  - Data type for each column (integer, floating point, text, date, time, etc.)
  - Whether columns required (default) or optional.
  - Default values for columns (optional) to be used when no value is supplied.
- Tables represent objects, events, or relationships

“Chinook” online music store

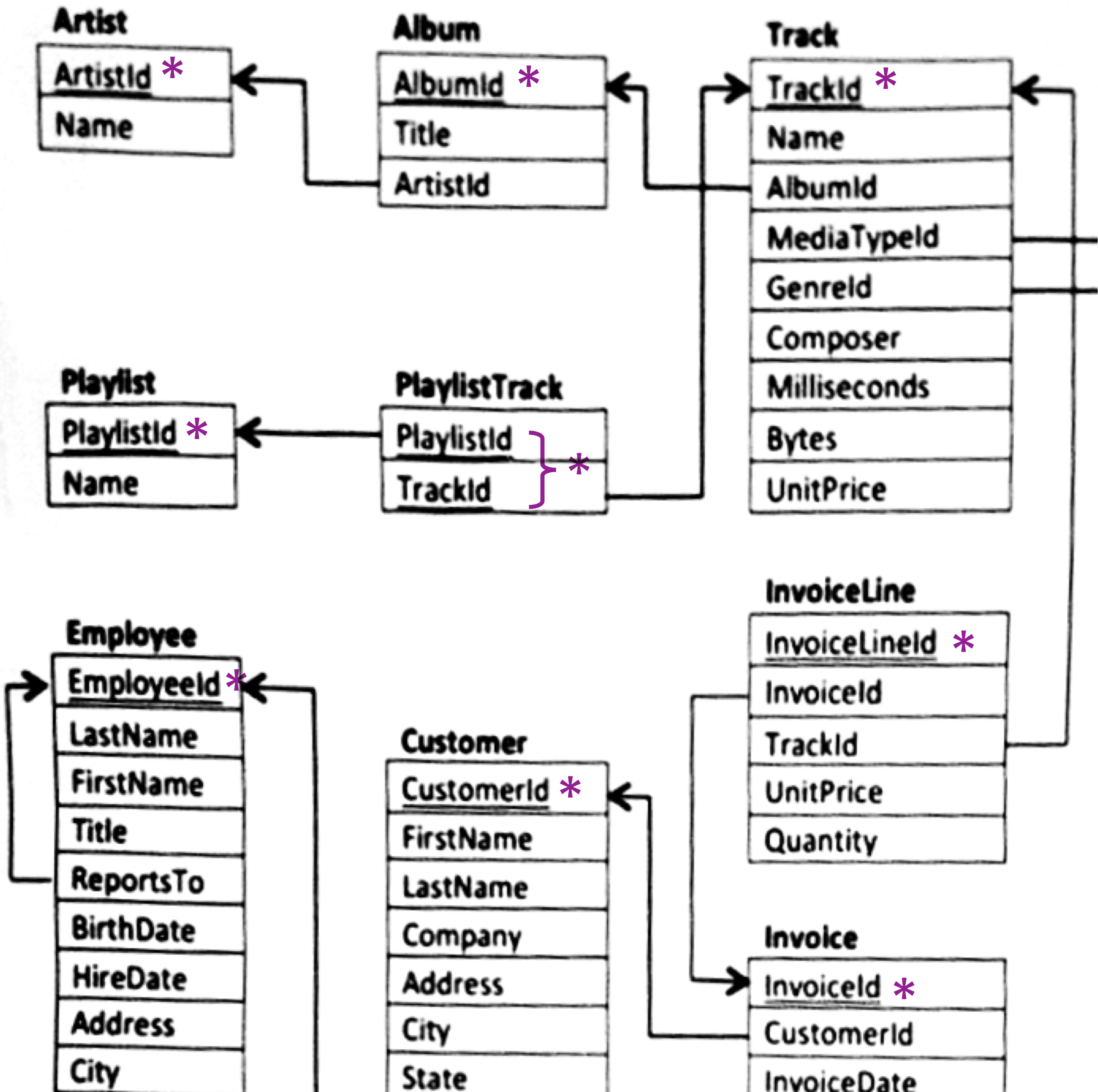
- 11 tables
- Let’s explore what this diagram means, and why this design was chosen.



Primary Keys

- Every table has a unique ***primary key*** – the column(s) that uniquely identify each row.
- No two rows can have the same primary key value.
- The primary key defines the principal feature of each row.
- Often it's an integer identifier
- **PlaylistTrack** table is different. It uses a *composite* primary key (made of two columns) and it lacks an integer identifier.

- In this class, we will **underline** primary keys in the diagrams.



Artist
ArtistId *
Name

Album
AlbumId *
Title
ArtistId

Track
TrackId *
Name
AlbumId
MediaTypeId
GenreId
Composer
Milliseconds
Bytes
UnitPrice

Playlist
PlaylistId *
Name

PlaylistTrack
PlaylistId
TrackId
*

InvoiceLine
InvoiceLineId *
InvoiceId
TrackId
UnitPrice
Quantity

Employee
EmployeeId *
LastName
FirstName
Title
ReportsTo
BirthDate
HireDate
Address
City

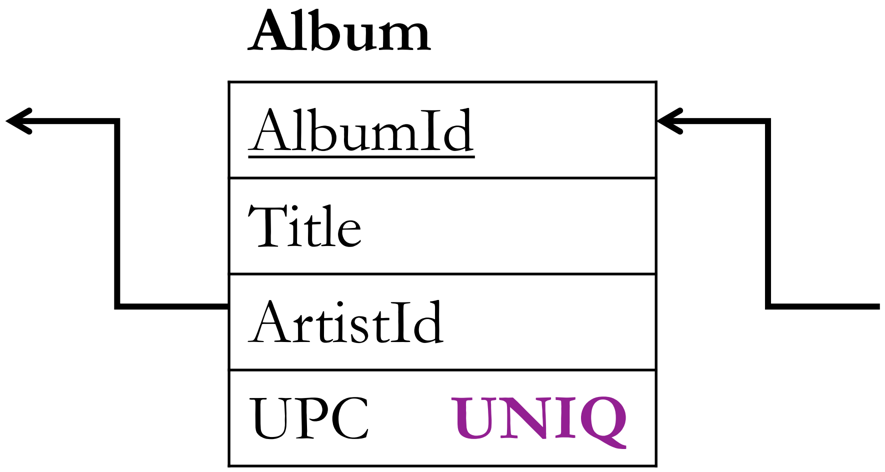
Customer
CustomerId *
FirstName
LastName
Company
Address
City
State

Invoice
InvoiceId *
CustomerId
InvoiceDate

Unique keys

- Unique keys are like additional (secondary) primary keys.
- No two rows can have the same value for a unique key.
- For example, we may wish to require that all Albums have both a unique AlbumId and a unique UPC (bar code):

- We write **UNIQ** next to columns with unique keys in the diagrams



Album

AlbumId

Title

ArtistId

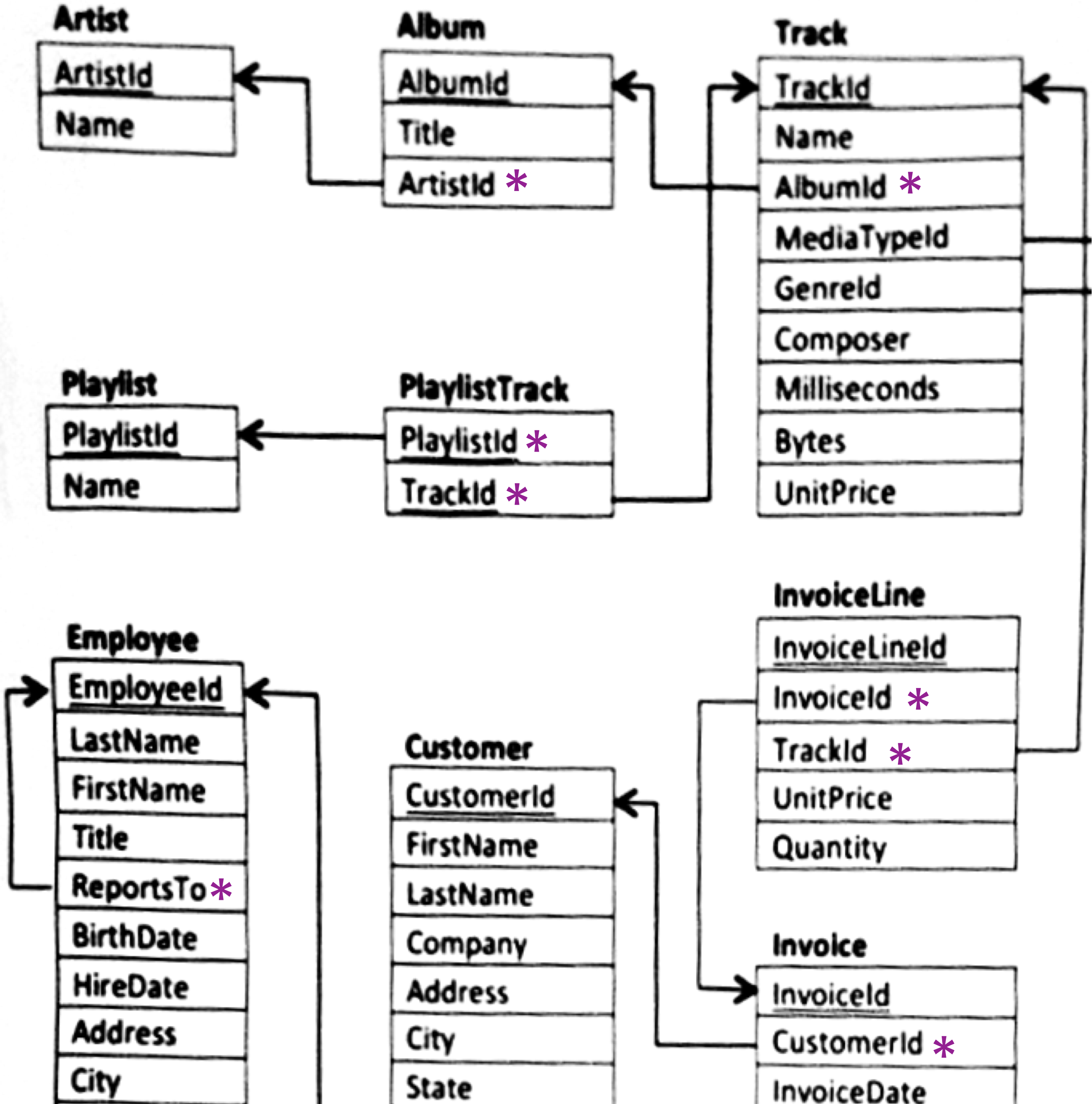
UPC **UNIQ**

- When inserting data into this table, the new row must have both a unique AlbumId and a unique UPC.

Foreign Keys

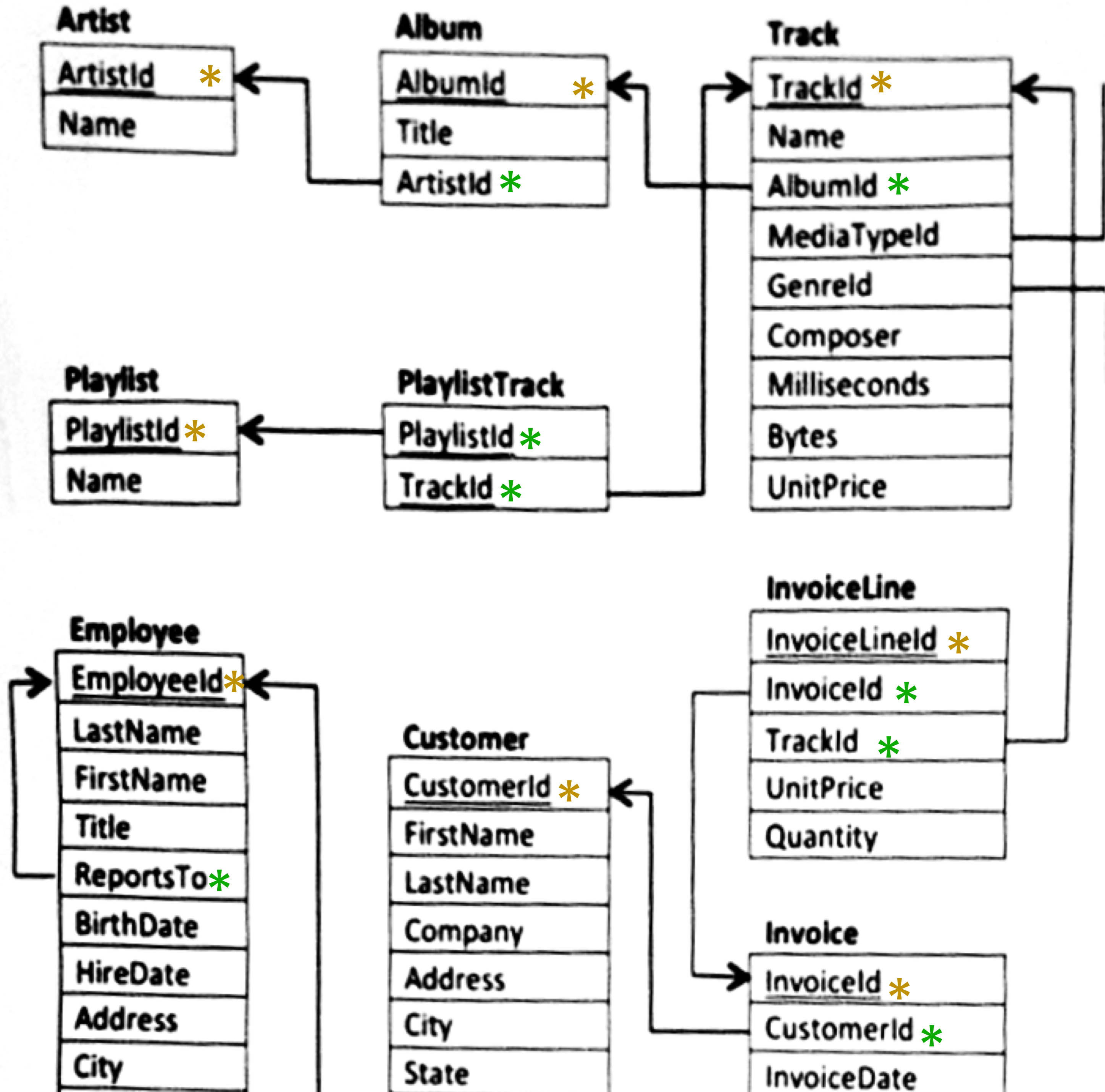
- Tables may be linked by *foreign keys* – columns that refer to keys in other tables.
- Usually these are integers ids, and should refer to a primary/unique key
- **PlaylistTrack** table is made entirely of foreign keys, so we call it a *linking table*.

- **Arrows** in these diagrams go from a foreign key to the column(s) they reference.



Parent and Child tables

- Foreign keys define a parent and child table.
    - Child points back to parent
    - Parent row must be created before child row
- A table can simultaneously be both a parent and child.
    - Album is a child to Artist, but a parent to Track.



Artist
ArtistId *
Name

Album
AlbumId *
Title
ArtistId *

Track
TrackId *
Name
AlbumId *
MediaTypeId
GenreId
Composer
Milliseconds
Bytes
UnitPrice

Playlist
PlaylistId *
Name

PlaylistTrack
PlaylistId *
TrackId *

InvoiceLine
InvoiceLineId *
InvoiceId *
TrackId *
UnitPrice
Quantity

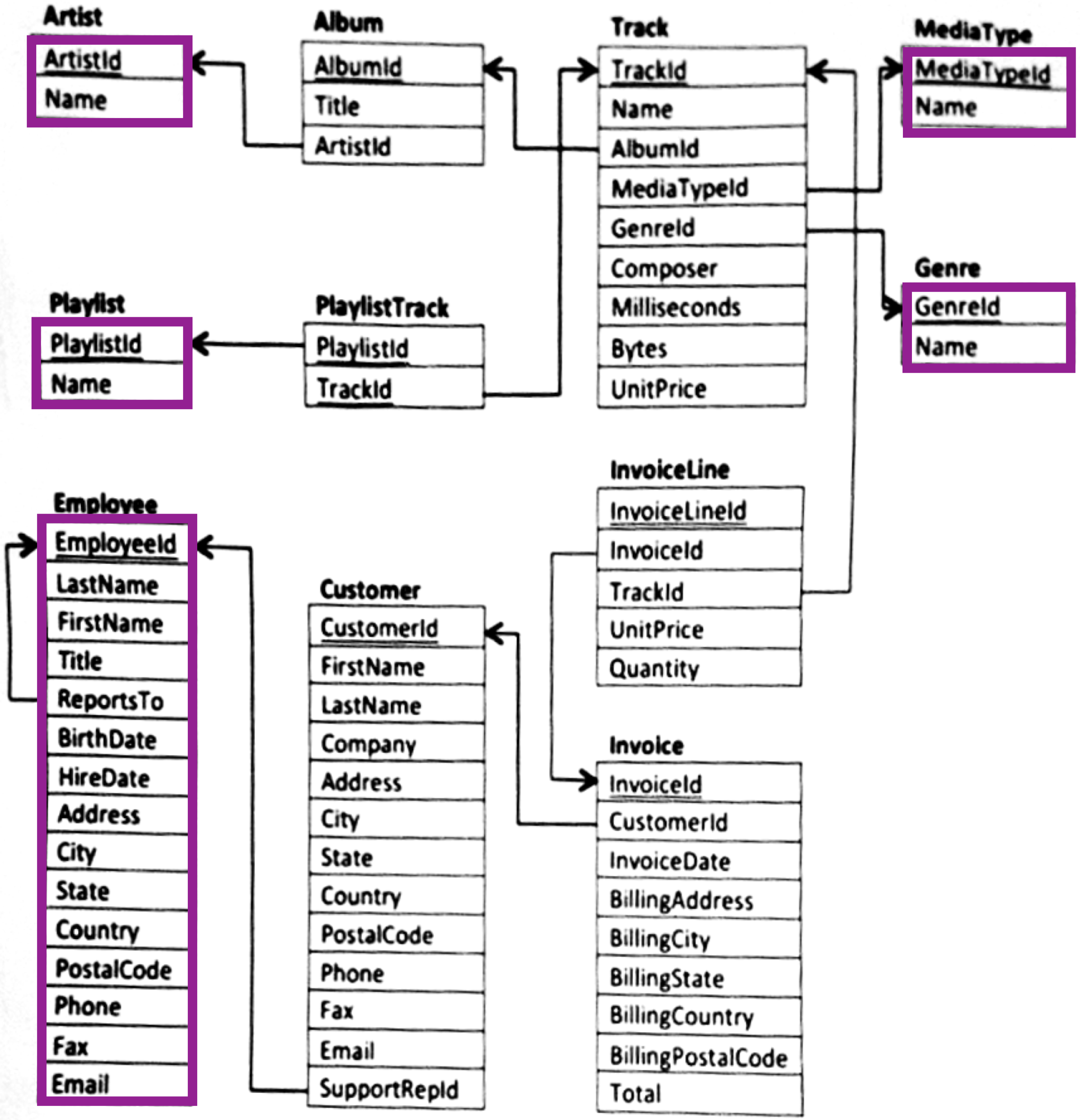
Employee
EmployeeId *
LastName
FirstName
Title
ReportsTo *
BirthDate
HireDate
Address
City

Customer
CustomerId *
FirstName
LastName
Company
Address
City
State

Invoice
InvoiceId *
CustomerId *
InvoiceDate

Highest-level parent tables

- In this example, you must create rows in these five tables before creating rows in the other tables.
- Just follow the arrows outward to determine all the rows that are necessary to fill a table.
- A Track requires MediaType, Genre, Album, and Artist.



Artist
ArtistId
Name

Album
AlbumId
Title
ArtistId

Track
TrackId
Name
AlbumId
MediaTypeId
GenreId
Composer
Milliseconds
Bytes
UnitPrice

MediaType
MediaTypeId
Name

Genre
GenreId
Name

Playlist
PlaylistId
Name

PlaylistTrack
PlaylistId
TrackId

InvoiceLine
InvoiceLineId
InvoiceId
TrackId
UnitPrice
Quantity

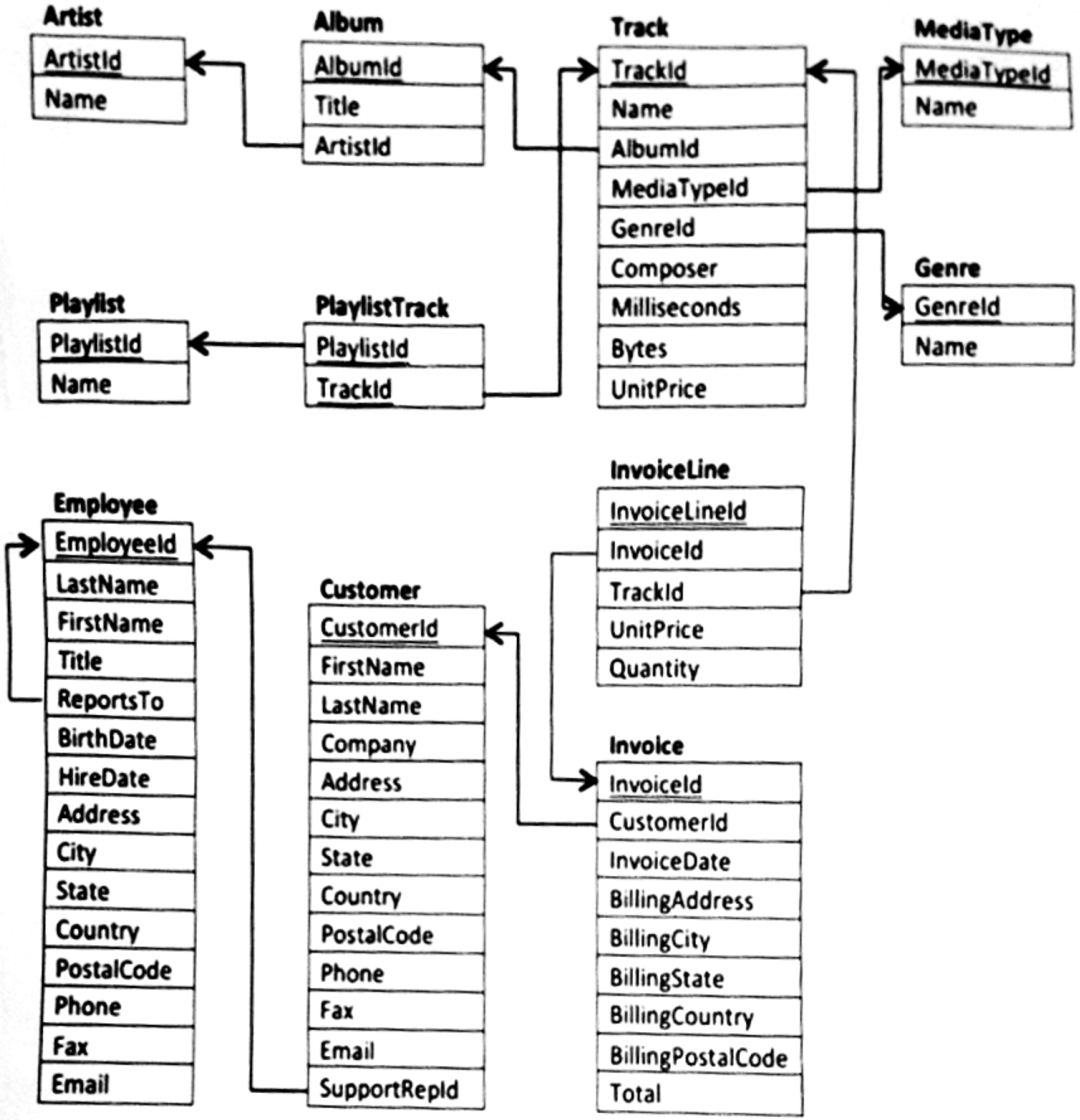
Employee
EmployeeId
LastName
FirstName
Title
ReportsTo
BirthDate
HireDate
Address
City
State
Country
PostalCode
Phone
Fax
Email

Customer
CustomerId
FirstName
LastName
Company
Address
City
State
Country
PostalCode
Phone
Fax
Email
SupportRepId

Invoice
InvoiceId
CustomerId
InvoiceDate
BillingAddress
BillingCity
BillingState
BillingCountry
BillingPostalCode
Total

Policies for deleting Foreign Keys

- Theoretically, you cannot delete a row from a parent table if a child table refers to it.
- For example, we cannot delete an artist if we still have one of her albums defined.
  - If the artist were deleted, the album row would have an invalid ArtistId.
- However, in practice, DB software is flexible.
- Three foreign key options for "ON DELETE":
  - **Restrict** (don't allow delete)
  - **Cascade** (delete children)
  - **Set NULL** (make orphan)



Artist
ArtistId
Name

Album
AlbumId
Title
ArtistId

Track
TrackId
Name
AlbumId
MediaTypeId
GenreId
Composer
Milliseconds
Bytes
UnitPrice

MediaType
MediaTypeId
Name

Genre
GenreId
Name

Playlist
PlaylistId
Name

PlaylistTrack
PlaylistId
TrackId

InvoiceLine
InvoiceLineId
InvoiceId
TrackId
UnitPrice
Quantity

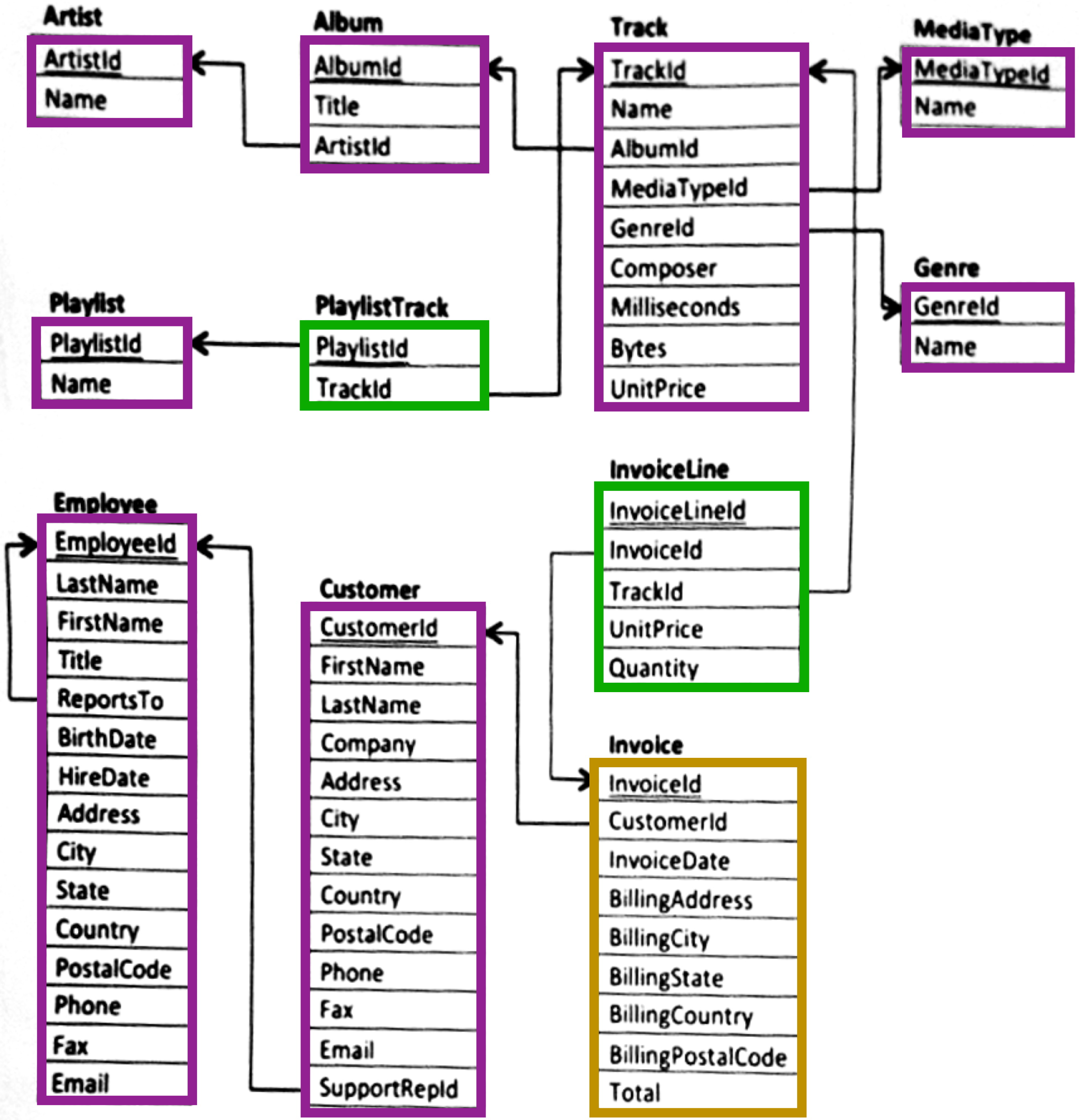
Employee
EmployeeId
LastName
FirstName
Title
ReportsTo
BirthDate
HireDate
Address
City
State
Country
PostalCode
Phone
Fax
Email

Customer
CustomerId
FirstName
LastName
Company
Address
City
State
Country
PostalCode
Phone
Fax
Email
SupportRepId

Invoice
InvoiceId
CustomerId
InvoiceDate
BillingAddress
BillingCity
BillingState
BillingCountry
BillingPostalCode
Total

Objects, Events, and Relationships

- These are not firm concepts and there are no strict definitions, but:
  - *Relationships* always have at least two foreign keys
  - *Events* always have a time and can repeat with a different id and time.



Artist
ArtistId
Name

Album
AlbumId
Title
ArtistId

Track
TrackId
Name
AlbumId
MediaTypeId
GenreId
Composer
Milliseconds
Bytes
UnitPrice

MediaType
MediaTypeId
Name

Genre
GenreId
Name

Playlist
PlaylistId
Name

PlaylistTrack
PlaylistId
TrackId

InvoiceLine
InvoiceLineId
InvoiceId
TrackId
UnitPrice
Quantity

Employee
EmployeeId
LastName
FirstName
Title
ReportsTo
BirthDate
HireDate
Address
City
State
Country
PostalCode
Phone
Fax
Email

Customer
CustomerId
FirstName
LastName
Company
Address
City
State
Country
PostalCode
Phone
Fax
Email
SupportRepId

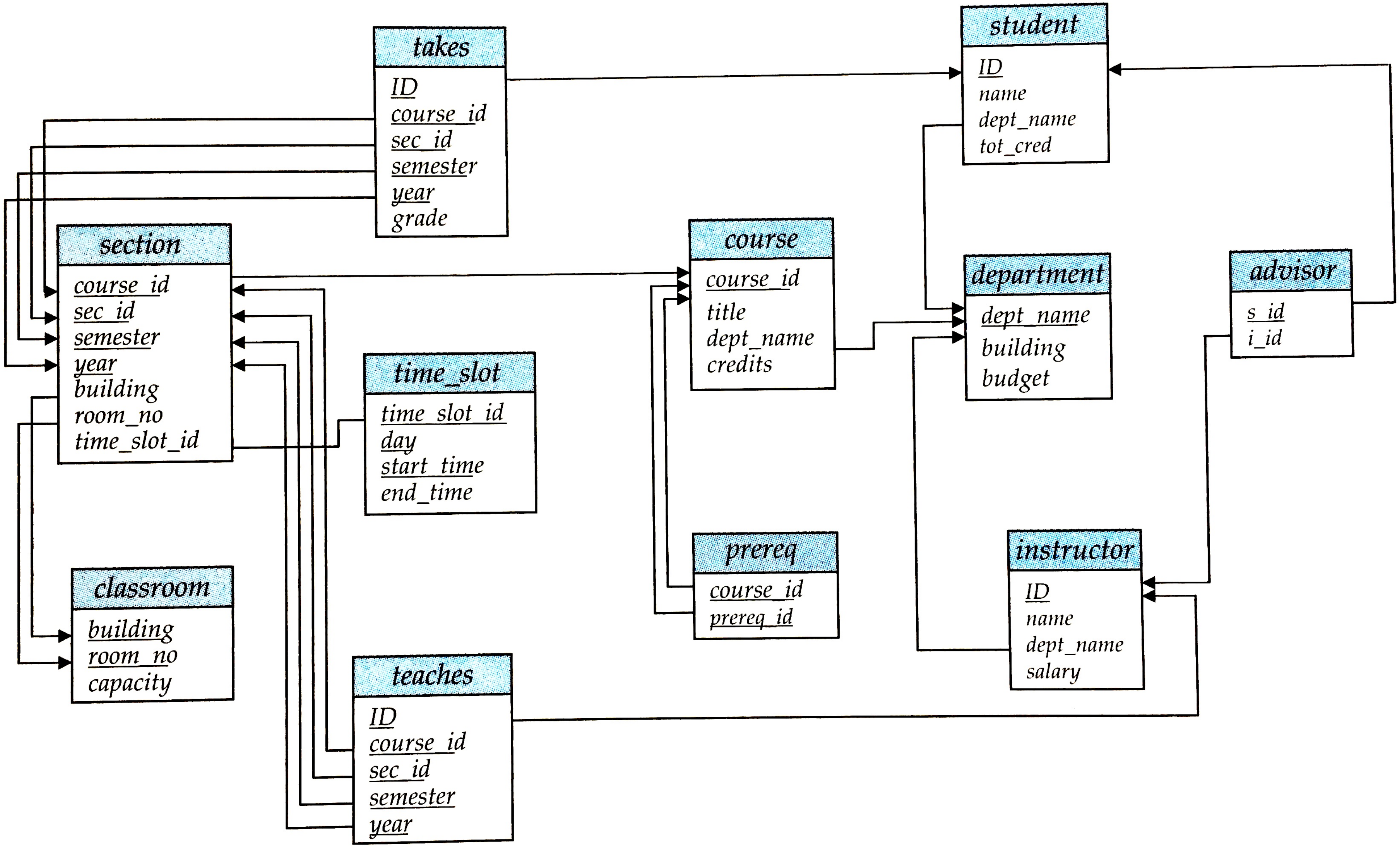
Invoice
InvoiceId
CustomerId
InvoiceDate
BillingAddress
BillingCity
BillingState
BillingCountry
BillingPostalCode
Total

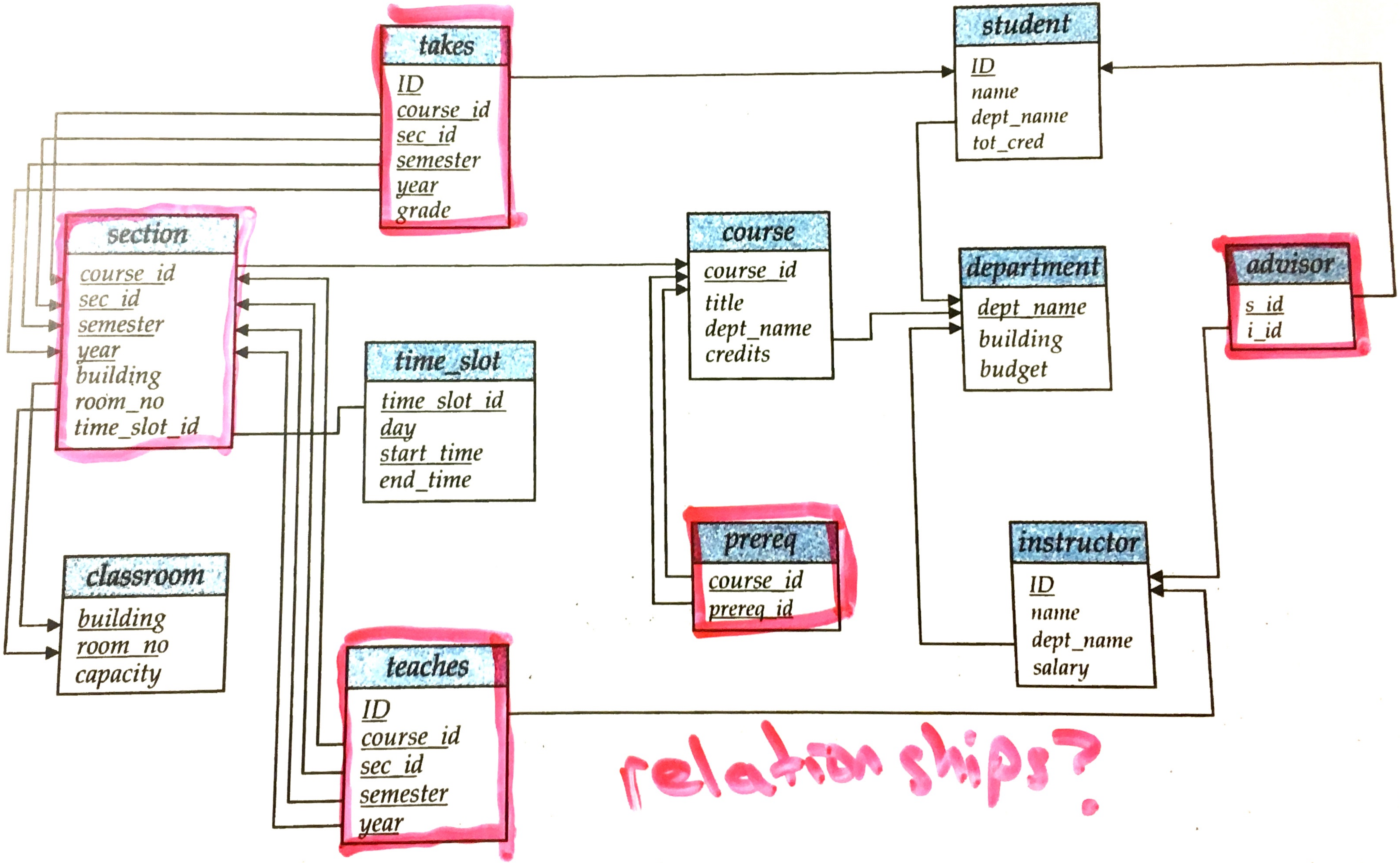
Another university database

Find:

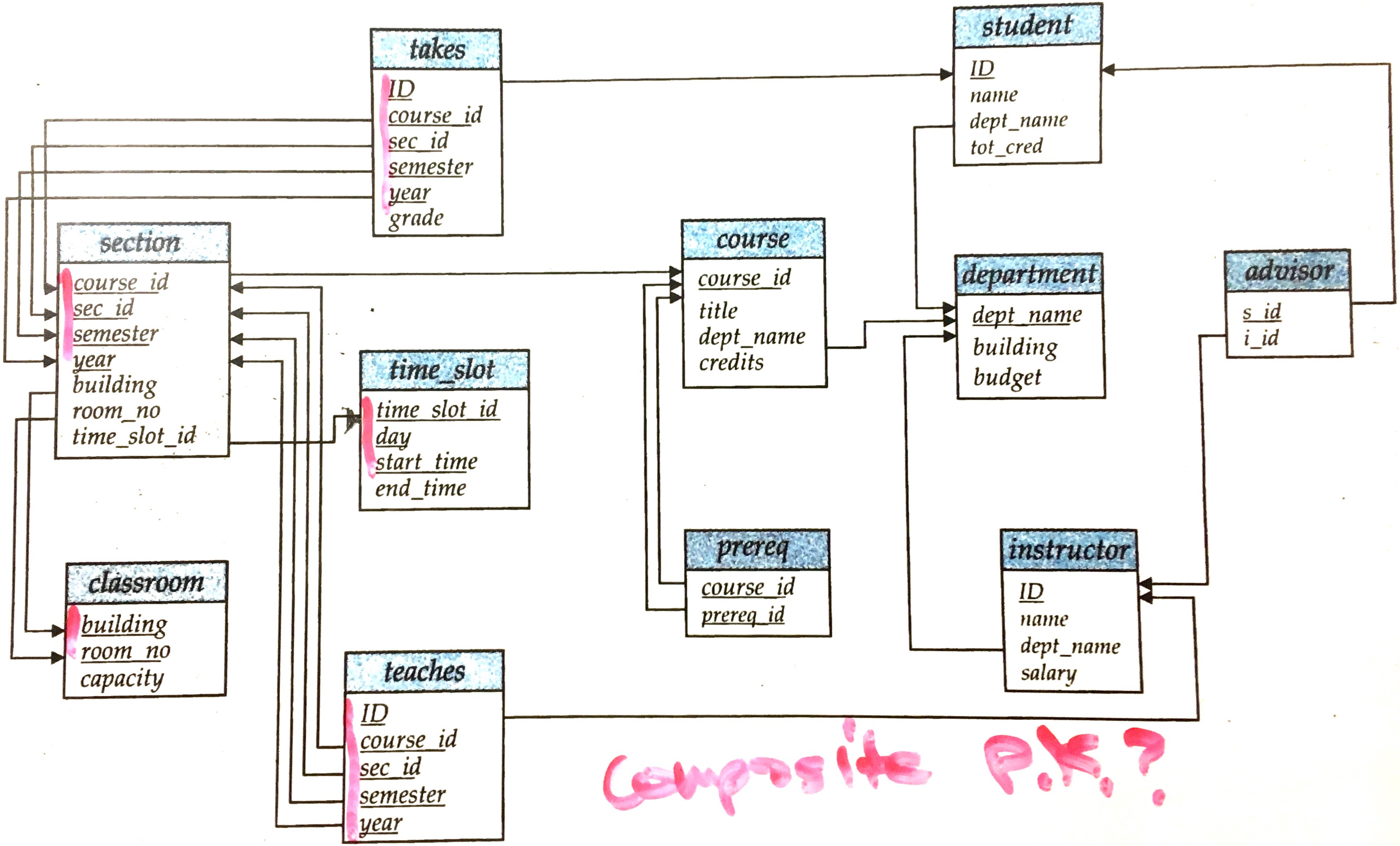
- Relationships
- Composite Primary Keys

- Change the design to allow multiple majors per student.

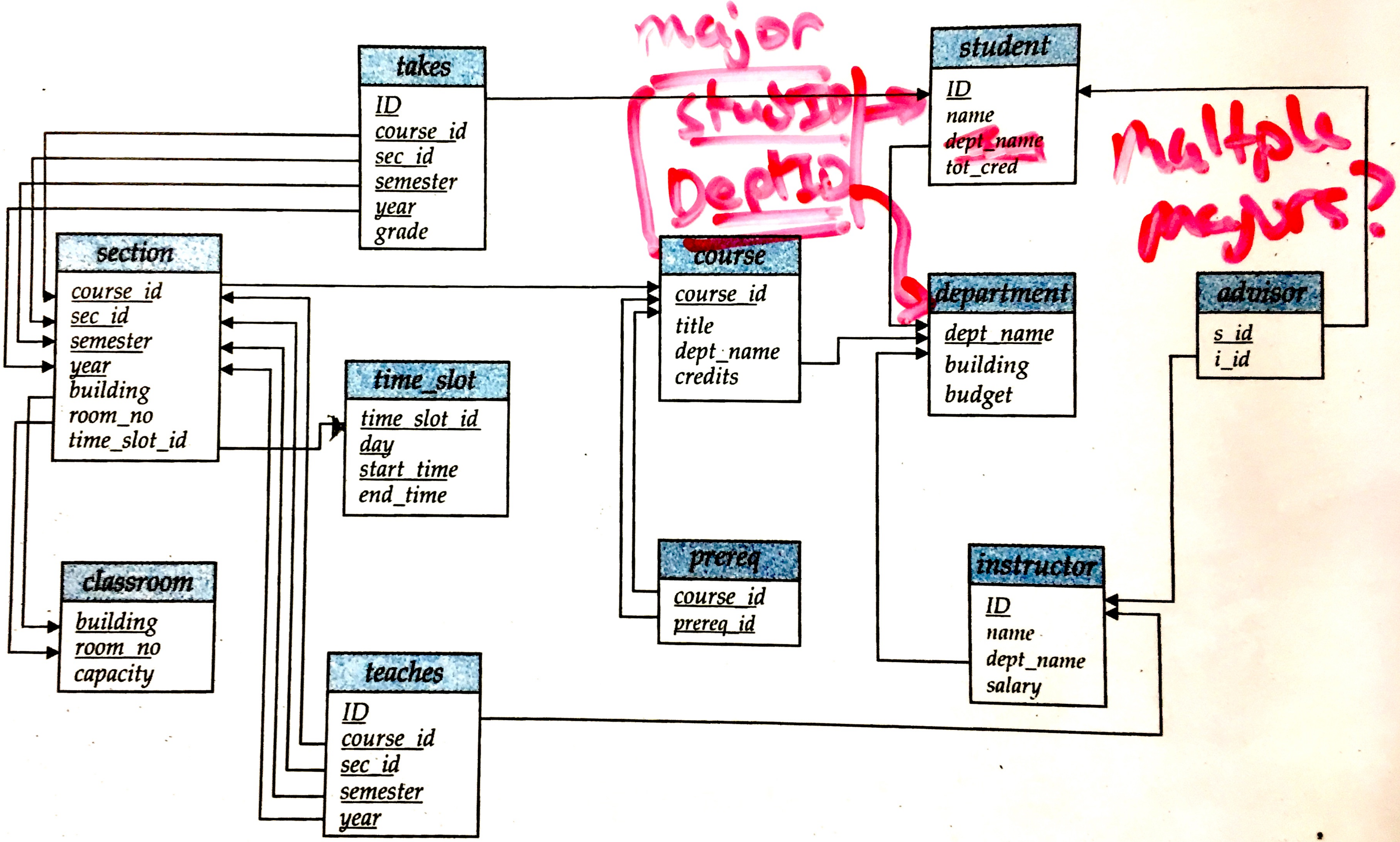




relationships?



Composite P.K.?



takes
ID
course_id
sec_id
semester
year
grade

student
ID
name
~~dept_name~~
tot_cred

section
course_id
sec_id
semester
year
building
room_no
time_slot_id

course
course_id
title
dept_name
credits

department
dept_name
building
budget

advisor
s_id
i_id

time_slot
time_slot_id
day
start_time
end_time

classroom
building
room_no
capacity

prereq
course_id
prereq_id

instructor
ID
name
dept_name
salary

teaches
ID
course_id
sec_id
semester
year

Table relationships in depth

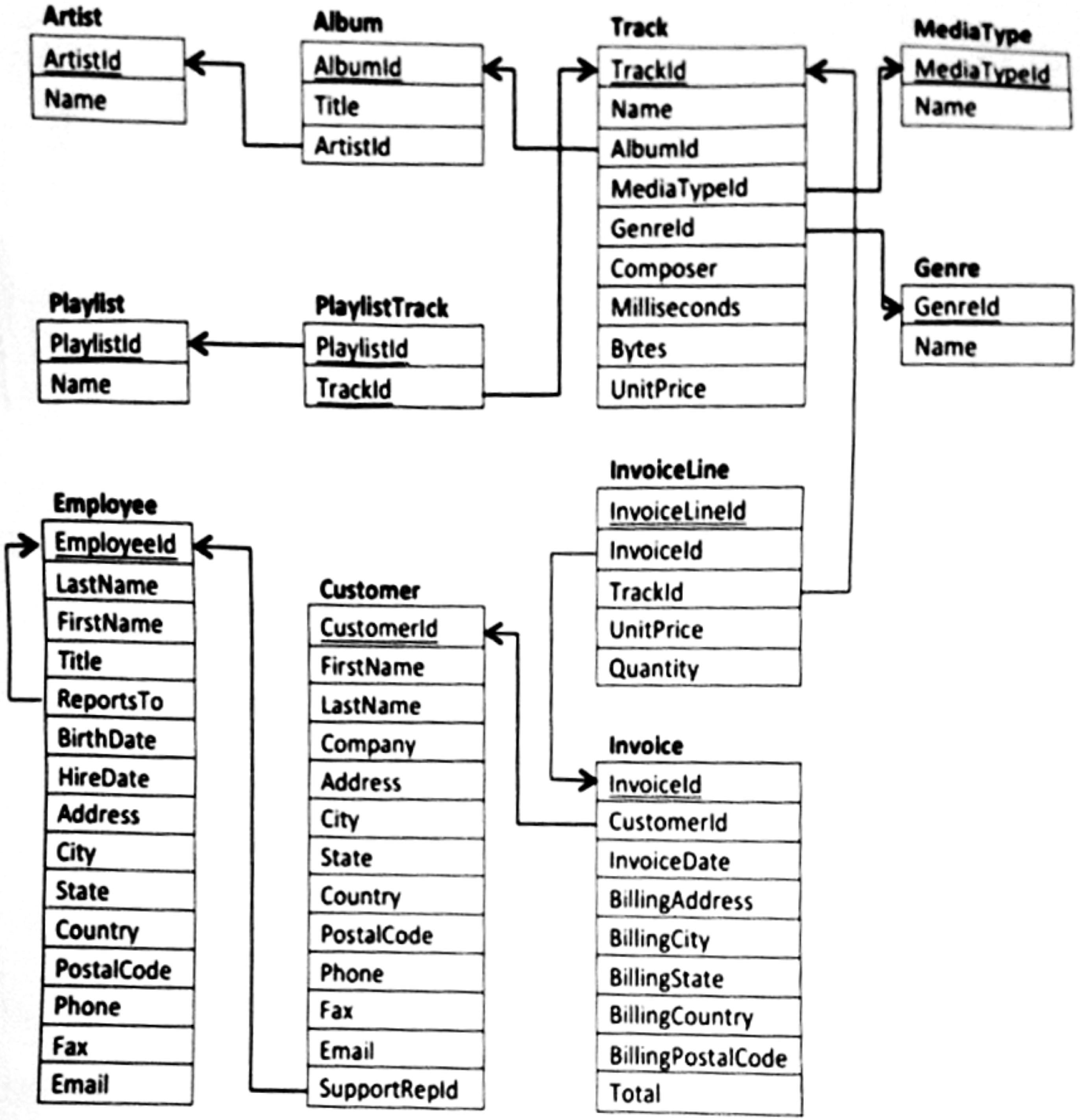
Foreign keys can relate table rows in three ways:

- One to One
- One to Many
- Many to Many

One to Many

(or equivalently "many to one")

- Most foreign keys create one-to-many relationships
- Created when a column that is **not a primary key** has a foreign key.
- All of the arrows in this diagram represent one-to-many relationships.
  - Many of the rows in the child table can be related one row in the parent table.



Artist
ArtistId
Name

Album
AlbumId
Title
ArtistId

Track
TrackId
Name
AlbumId
MediaTypeId
GenreId
Composer
Milliseconds
Bytes
UnitPrice

MediaType
MediaTypeId
Name

Genre
GenreId
Name

Playlist
PlaylistId
Name

PlaylistTrack
PlaylistId
TrackId

InvoiceLine
InvoiceLineId
InvoiceId
TrackId
UnitPrice
Quantity

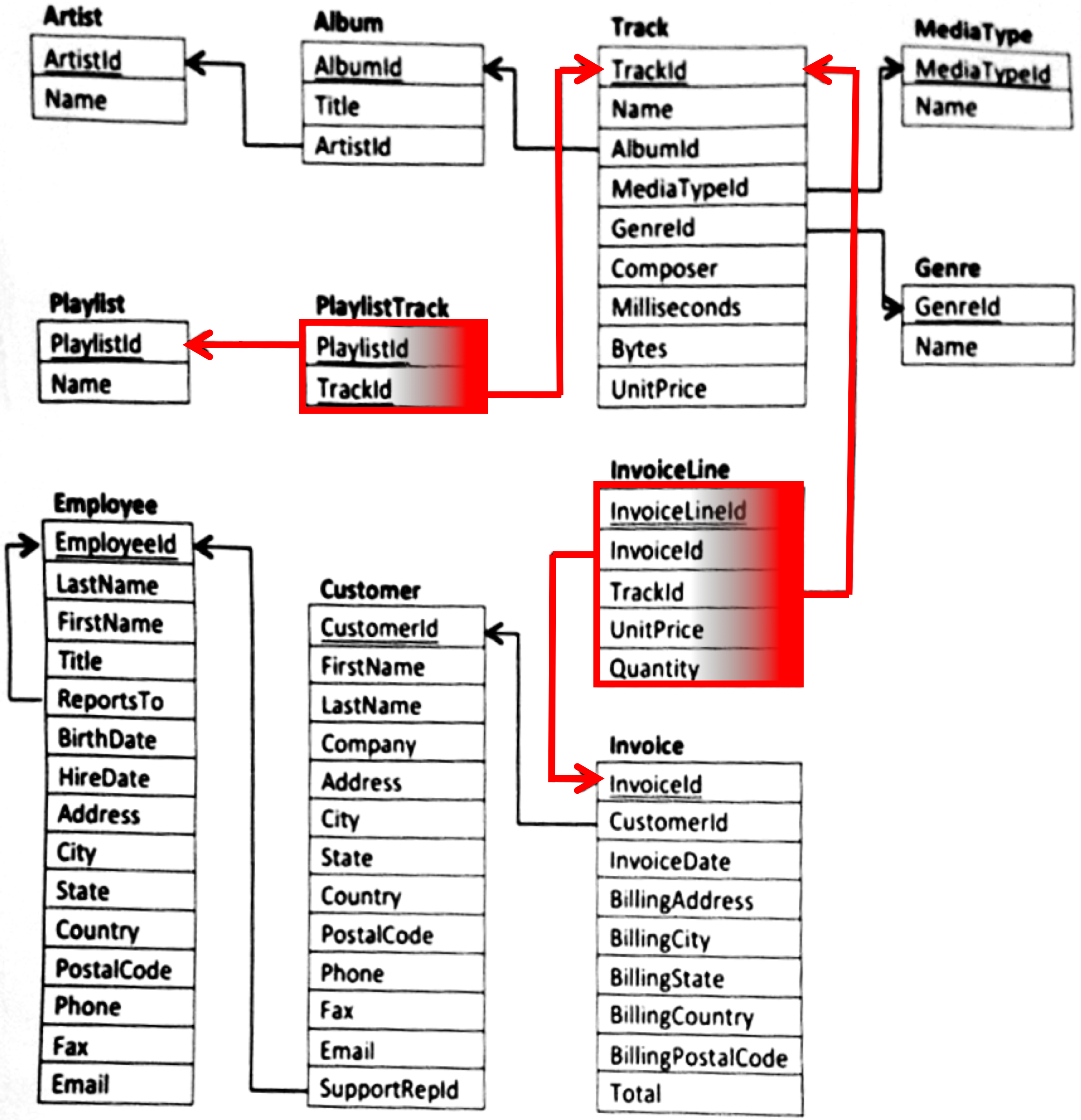
Employee
EmployeeId
LastName
FirstName
Title
ReportsTo
BirthDate
HireDate
Address
City
State
Country
PostalCode
Phone
Fax
Email

Customer
CustomerId
FirstName
LastName
Company
Address
City
State
Country
PostalCode
Phone
Fax
Email
SupportRepId

Invoice
InvoiceId
CustomerId
InvoiceDate
BillingAddress
BillingCity
BillingState
BillingCountry
BillingPostalCode
Total

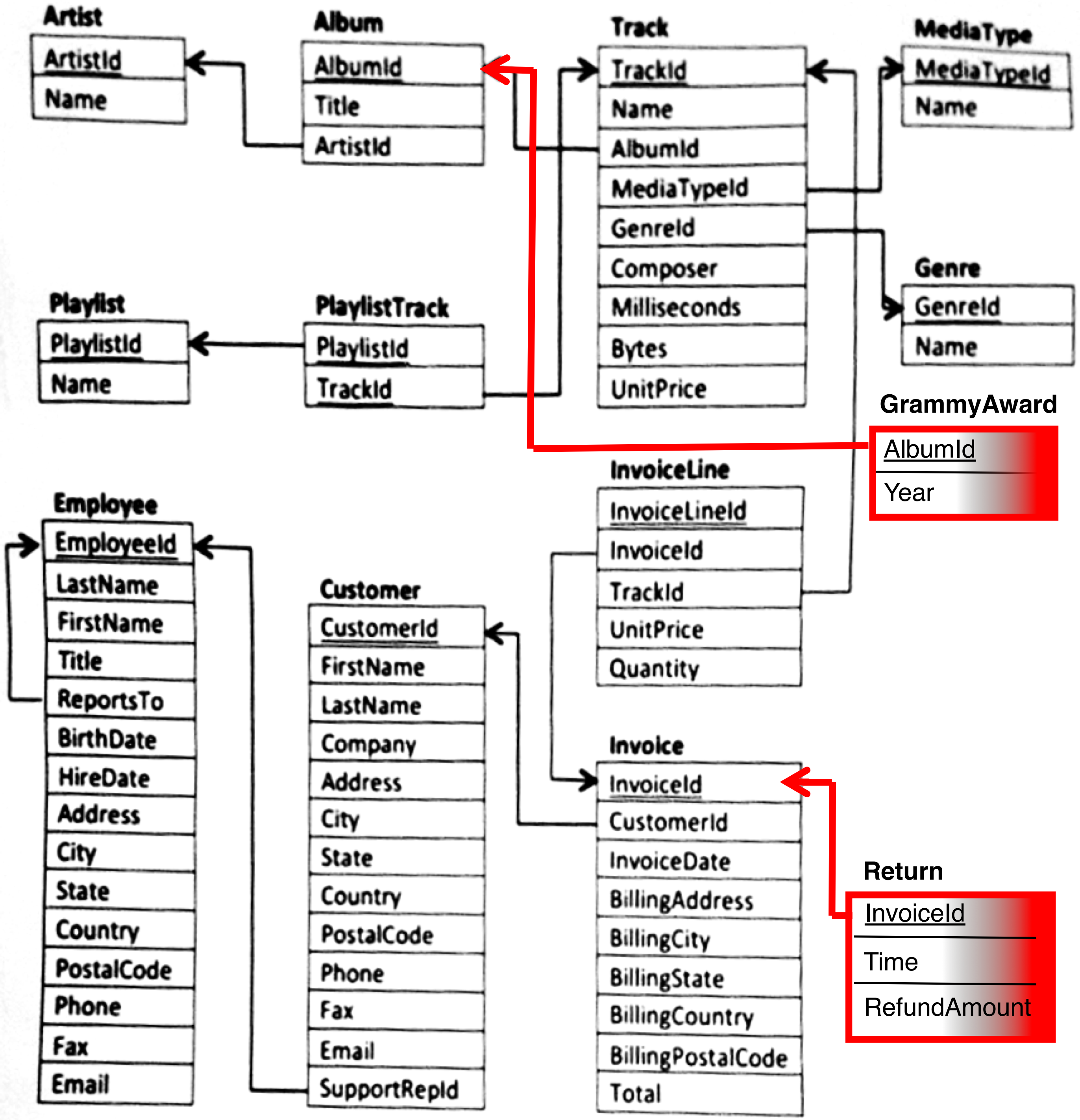
Many to Many

- Two one-to-many relationships starting at the same table can create a many-to-many relationship
- These are represented with ***linking tables.***
- But, some tables can be classified in multiple ways:
    - We think of **Track** as either an *object* or as a *many-to-many* relationship between albums and genres.



One to One

- One-to-one relationships exists when a primary (or unique) key is also a foreign key.
- In other words, there is an arrow pointing from one primary/unique key to another.
    - The fact that it's a unique key prevents it appearing multiple times (thus, not one-to-many).
- The child is a ***subset table***.
- Subset tables are an alternative to having optional columns in the parent table.



Artist
ArtistId
Name

Album
AlbumId
Title
ArtistId

Track
TrackId
Name
AlbumId
MediaTypeId
GenreId
Composer
Milliseconds
Bytes
UnitPrice

MediaType
MediaTypeId
Name

Genre
GenreId
Name

Playlist
PlaylistId
Name

PlaylistTrack
PlaylistId
TrackId

GrammyAward
AlbumId
Year

InvoiceLine
InvoiceLineId
InvoiceId
TrackId
UnitPrice
Quantity

Employee
EmployeeId
LastName
FirstName
Title
ReportsTo
BirthDate
HireDate
Address
City
State
Country
PostalCode
Phone
Fax
Email

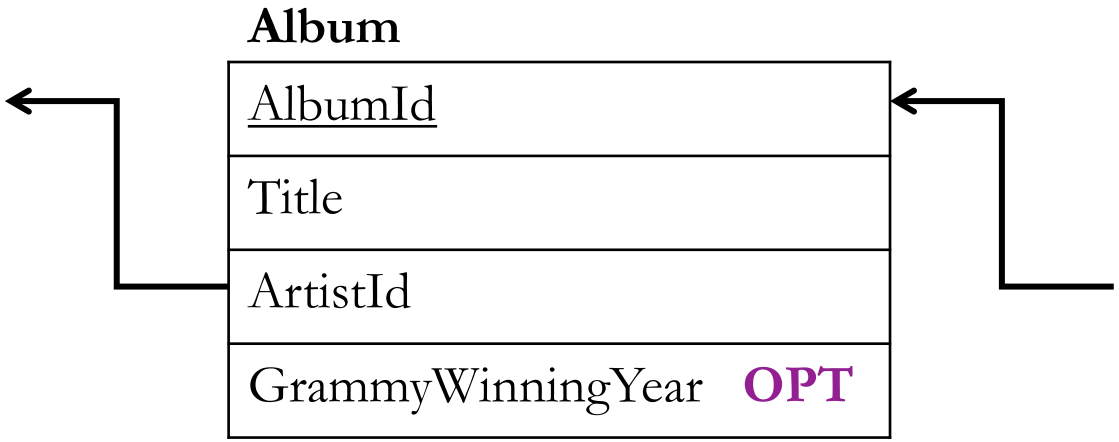
Customer
CustomerId
FirstName
LastName
Company
Address
City
State
Country
PostalCode
Phone
Fax
Email
SupportRepId

Invoice
InvoiceId
CustomerId
InvoiceDate
BillingAddress
BillingCity
BillingState
BillingCountry
BillingPostalCode
Total

Return
InvoiceId
Time
RefundAmount

Optional columns

- Strictly speaking, optional columns are not necessary
  - Just move the column to a new *subset table*
- But in practice, optional columns are common
  - Absent values have **NULL** value.
  - When defining the database tables you specify whether NULL is allowed.
- We write **OPT** next to optional columns in the diagrams



Album

AlbumId

Title

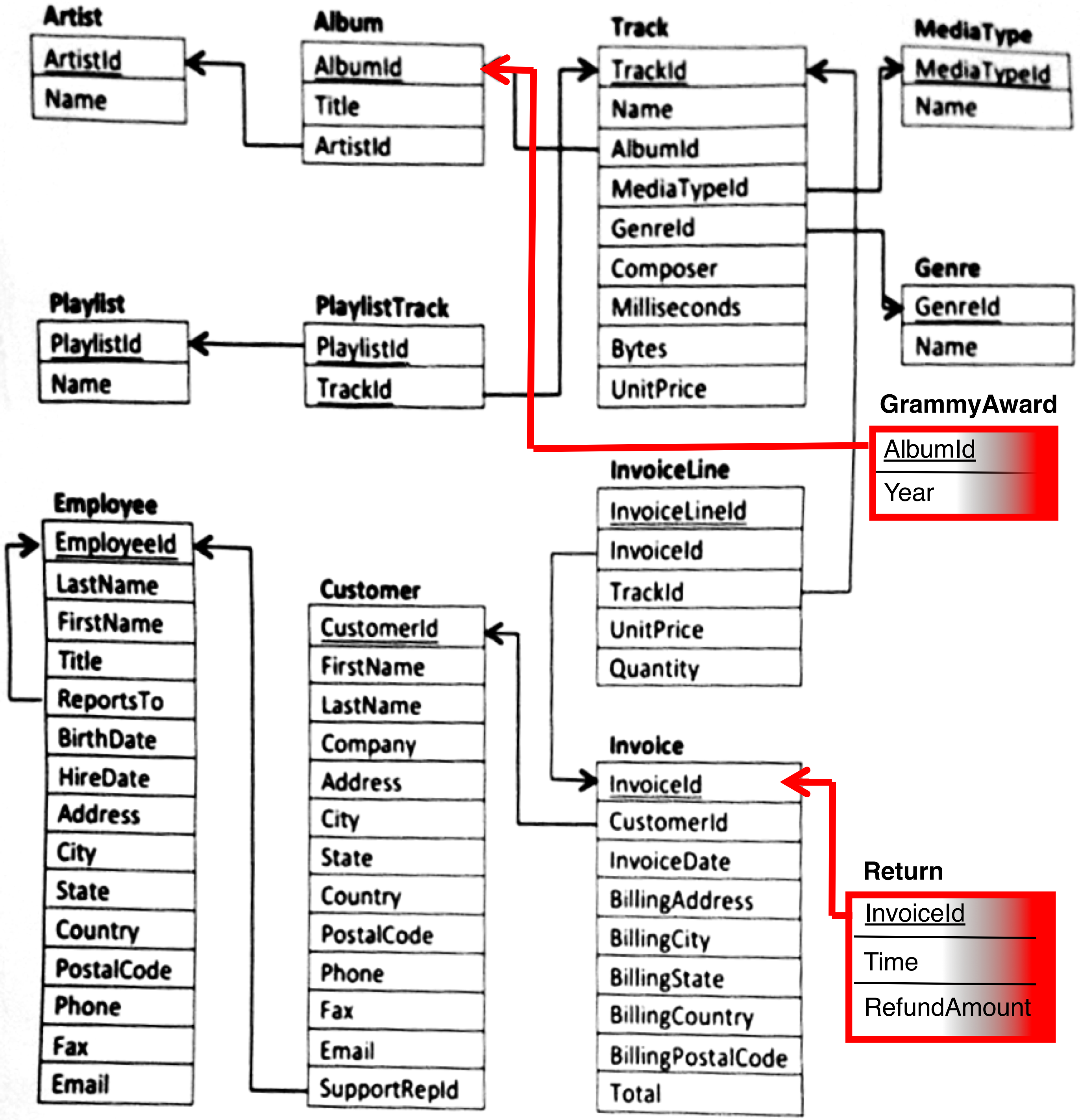
ArtistId

GrammyWinningYear **OPT**

- This is a good alternative design to the Grammy Award table on the previous slide.

Subset tables vs optional columns

- *Grammy Award* subset table supplies just one optional value, so it can be replaced by an optional column in Album.
- However, the *Return* table provides several related columns of optional information that must be provided "all or none."
  - If there is a return *Time*, then we must have a *RefundAmount*
- Thus, returns cannot be well modeled with optional columns in the Invoice table.



Artist
ArtistId
Name

Album
AlbumId
Title
ArtistId

Track
TrackId
Name
AlbumId
MediaTypeId
GenreId
Composer
Milliseconds
Bytes
UnitPrice

MediaType
MediaTypeId
Name

Genre
GenreId
Name

GrammyAward
AlbumId
Year

Playlist
PlaylistId
Name

PlaylistTrack
PlaylistId
TrackId

InvoiceLine
InvoiceLineId
InvoiceId
TrackId
UnitPrice
Quantity

Employee
EmployeeId
LastName
FirstName
Title
ReportsTo
BirthDate
HireDate
Address
City
State
Country
PostalCode
Phone
Fax
Email

Customer
CustomerId
FirstName
LastName
Company
Address
City
State
Country
PostalCode
Phone
Fax
Email
SupportRepId

Invoice
InvoiceId
CustomerId
InvoiceDate
BillingAddress
BillingCity
BillingState
BillingCountry
BillingPostalCode
Total

Return
InvoiceId
Time
RefundAmount

Data Modeling summary

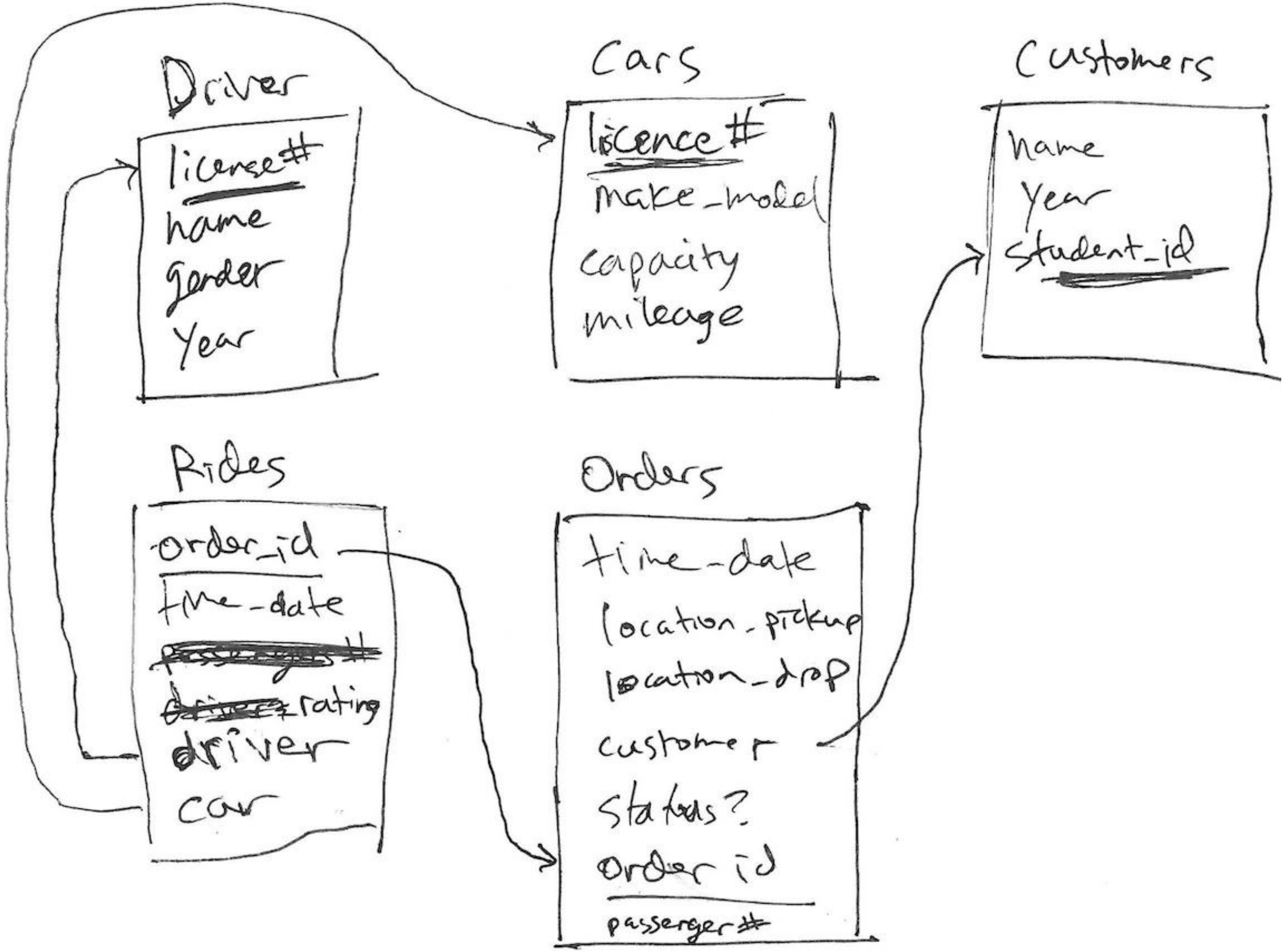
- ***Primary*** and ***unique keys*** prevent rows from repeating certain columns.
- ***Foreign keys*** link tables and point to primary/unique keys.
  - Create ***parent/child*** table relationships. Must fill in parent before child.
  - Parent rows cannot be deleted unless default foreign key behavior is changed.
    - Must kill children first!
- Tables can represent ***Objects***, ***Events*** (have time), and ***Relationships***:
  - ***One to many*** relationships allow multiple child rows referencing one parent row
    - Implemented with a single foreign key.
  - ***Many to many*** relationships link two or more rows
    - Implemented with a linking table
  - ***One to one*** relationships create subset tables
    - Implemented with a single foreign key that is also a unique key.

Database Schema Design steps

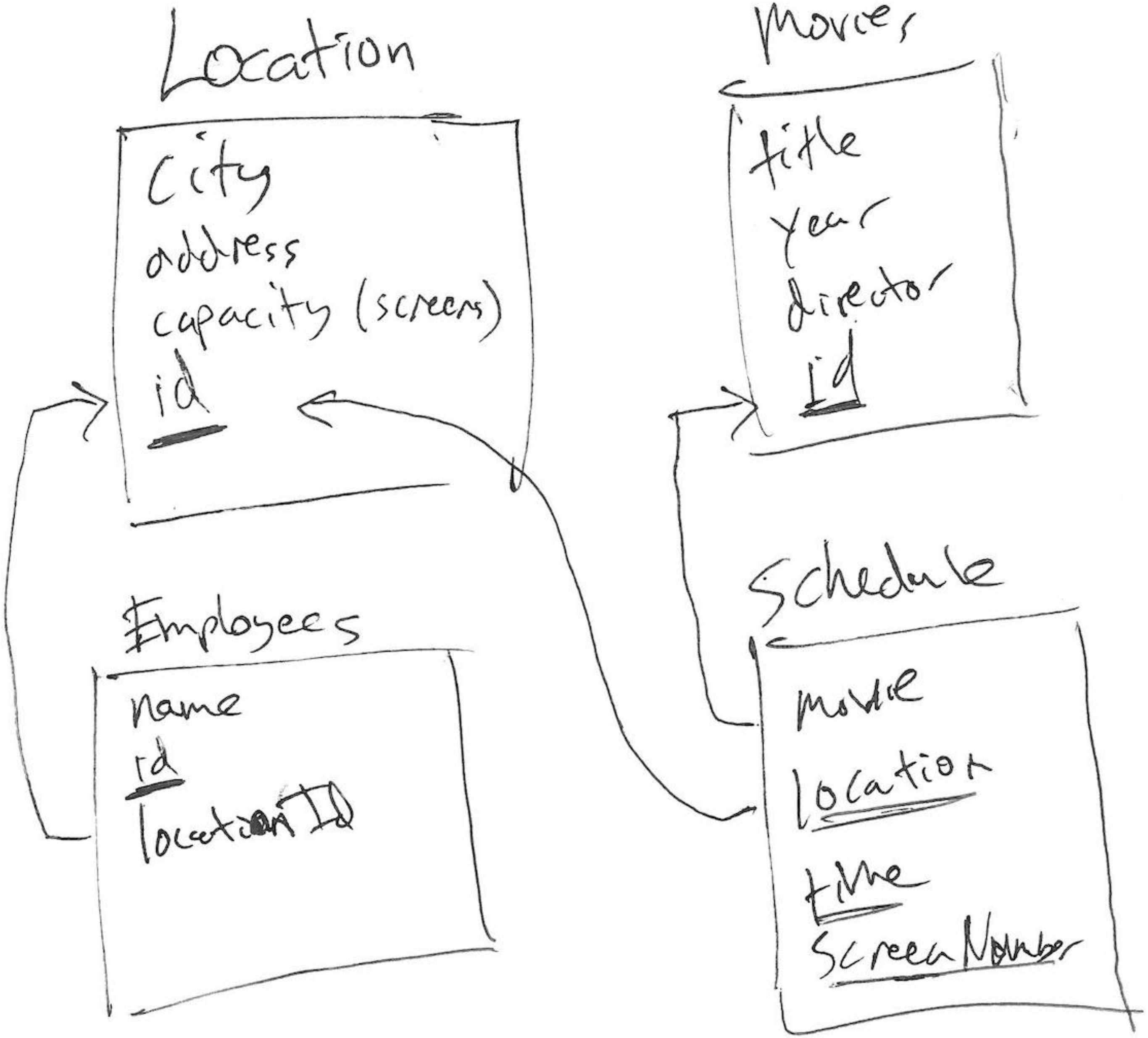
1. **List tables**
    - (Objects, events, relationships)
2. Choose **primary key** for each table
3. Choose **foreign keys** to link tables
4. Add uniq keys and/or optional columns
5. Refine the design, revisiting decisions made above

Data modeling examples

Safe Ride


Driver
license#
name
gender
Year
Cars
licence#
make_model
capacity
mileage
Customers
name
Year
student_id
Rides
order_id
time_date
~~Passengers~~ #
~~driver~~ rating
driver
car
Orders
time_date
location_pickup
location_drop
customer
status?
order id
passenger#

Movie theater Chain


Location
City
address
capacity (screens)
id
Movies
title
year
director
id
Employees
name
id
locationID
Schedule
movie
location
time
ScreenNumber