

# MSiA-413 Introduction to Databases and Information Retrieval

## Lecture 4 Data Modeling for Relational Databases Primary and Foreign Keys

Instructor: Nikos Hardavellas

Slides adapted from Steve Tarzia

# Last lecture we covered time and text encodings

- Times can be stored most easily as integer epoch seconds (seconds since 1970)
- The human-readable form of a given time depends on the time-zone.
- 8 bits is a byte, and hex encoding is a shorthand for binary representation
- Text is most often encoded in UTF-8 format
  - A variable length encoding using one to four bytes per character
  - Backward-compatible with ASCII: the older standard for basic U.S. characters.

# Why use a relational database?

- **Scalability** – work with data larger than computer's RAM
- **Persistence** – keep data around after your program finishes
- **Indexing** – efficiently sort & search along various dimensions
- **Integrity** – restrict data type, disallow duplicate entries
- **Deduplication** – save space, keep common data consistent
- **Concurrency** – multiple users or applications can read/write
- **Security** – different users can have access to specific data
- **“Researchability”** – SQL allows you to concisely express analysis

# Tables are the main concept in Relational DBs

Table (*relation*)

4 Columns (*attributes, fields*)

Primary key

3 Rows (*tuples, records*)

customer			
<i>id</i>	<i>name</i>	<i>address</i>	<i>city</i>
1	Becky G. Novick	1131 Poe Road	Houston
2	Pamela C. Tweed	3554 College View	Greenville
3	Danny C. Bost	1720 Gateway Ave	Brattleboro

The diagram illustrates a table named 'customer' with four columns: 'id', 'name', 'address', and 'city'. The 'id' column is highlighted as the primary key. The table contains three rows of data. Annotations with arrows and brackets identify the table as a 'relation', the columns as 'attributes' or 'fields', the 'id' column as the 'Primary key', and the three data rows as 'tuples' or 'records'.

# DB design process answers these questions:

- What tables do we need?
  - How to separate the data logically?
- What columns?
  - Data types for columns?
  - How will rows be uniquely identified (i.e., what should be this table's key)?
  - Are some columns optional?
- How will tables be linked?

customer			
<i>id</i>	<i>name</i>	<i>address</i>	<i>city</i>
1	Becky G. Novick	1131 Poe Road	Houston
2	Pamela C. Tweed	3554 College View	Greenville
3	Danny C. Bost	1720 Gateway Ave	Brattleboro

Sometimes we start with one redundant table and break it down to reflect the logical components

staff					
<i>id</i>	<i>name</i>	<i>department</i>	<i>building</i>	<i>room</i>	<i>facilities ext. #</i>
11	Bob	Industrial Eng.	Tech	100	1-1000
20	Betsy	Computer Sci.	Ford	100	1-5003
21	Fran	Industrial Eng.	Tech	101	1-1000
22	Frank	Chemistry	Tech	102	1-1000
35	Sarah	Physics	Mudd	200	1-2005
40	Sam	Materials Sci.	Cook	10	1-3004
54	Pat	Computer Sci.	Ford	102	1-5003

# This is called *Normalization*

staff			
<i>id</i>	<i>name</i>	<i>room</i>	<i>department ID</i>
11	Bob	100	1
20	Betsy	100	2
21	Fran	101	1
22	Frank	102	4
35	Sarah	200	5
40	Sam	10	7
54	Pat	102	2

department		
<i>id</i>	<i>name</i>	<i>building ID</i>
1	Industrial Eng.	1
2	Computer Sci.	2
4	Chemistry	1
5	Physics	4
7	Materials Sci.	5

building		
<i>id</i>	<i>name</i>	<i>facilities ext. #</i>
1	Tech	1-1000
2	Ford	1-5003
4	Mudd	1-2005
5	Cook	1-3004
6	Garage	1-6001

- Removes redundancy
  - Save space
  - Edit values in one place, so duplicates don't get out of date
- Tables can be populated separately
- **But**, you are adding a new *id* column for each table

# Tables

- Represent objects, events, or relationships
  - Must be uniquely identifiable
  - Have attributes that the DB will store in columns
  - Can refer to rows in other tables
- ***Objects***: people, places, things
- ***Events***: usually associated with a specific time. Can recur.
- ***Relationships***: associate things

The best way to learn data modelling is by example



# Database **Schema** defines the data's structure

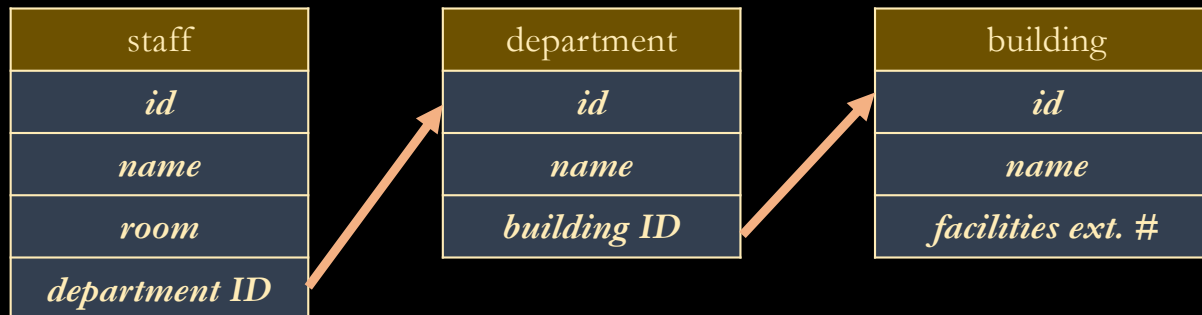
- Also called a data model
- It's *metadata* – data about data
- Defines:
  - Tables (their names)
  - Columns in each table (both the name and ***data type***)
  - Primary Key for each table (show it by underlining the column name)
  - Foreign Keys that link tables (they appear as additional columns)

staff			
<i>id</i>	<i>name</i>	<i>room</i>	<i>department ID</i>
11	Bob	100	1
20	Betsy	100	2
21	Fran	101	1
22	Frank	102	4
35	Sarah	200	5
40	Sam	10	7
54	Pat	102	2

department		
<i>id</i>	<i>name</i>	<i>building ID</i>
1	Industrial Eng.	1
2	Computer Sci.	2
4	Chemistry	1
5	Physics	4
7	Materials Sci.	5

building		
<i>id</i>	<i>name</i>	<i>facilities ext. #</i>
1	Tech	1-1000
2	Ford	1-5003
4	Mudd	1-2005
5	Cook	1-3004
6	Garage	1-6001

DB design diagram:



staff			
<i>id</i>	<i>name</i>	<i>room</i>	<i>department ID</i>
11	Bob	100	1
20	Betsy	100	2
21	Fran	101	1
22	Frank	102	4
35	Sarah	200	5
40	Sam	10	7
54	Pat	102	2

department		
<i>id</i>	<i>name</i>	<i>building ID</i>
1	Industrial Eng.	1
2	Computer Sci.	2
4	Chemistry	1
5	Physics	4
7	Materials Sci.	5

building		
<i>id</i>	<i>name</i>	<i>facilities ext. #</i>
1	Tech	1-1000
2	Ford	1-5003
4	Mudd	1-2005
5	Cook	1-3004
6	Garage	1-6001

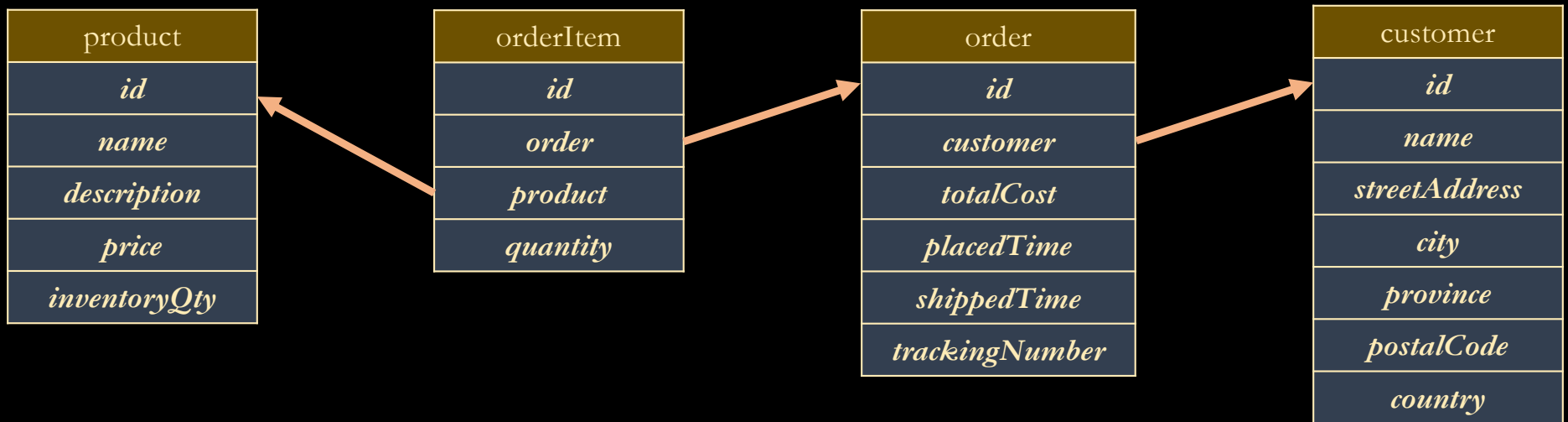
## DB schema:

staff
<i><u>id</u>: integer</i>
<i>name: varchar(128)</i>
<i>room: varchar(64)</i>
<i>department ID: integer</i>

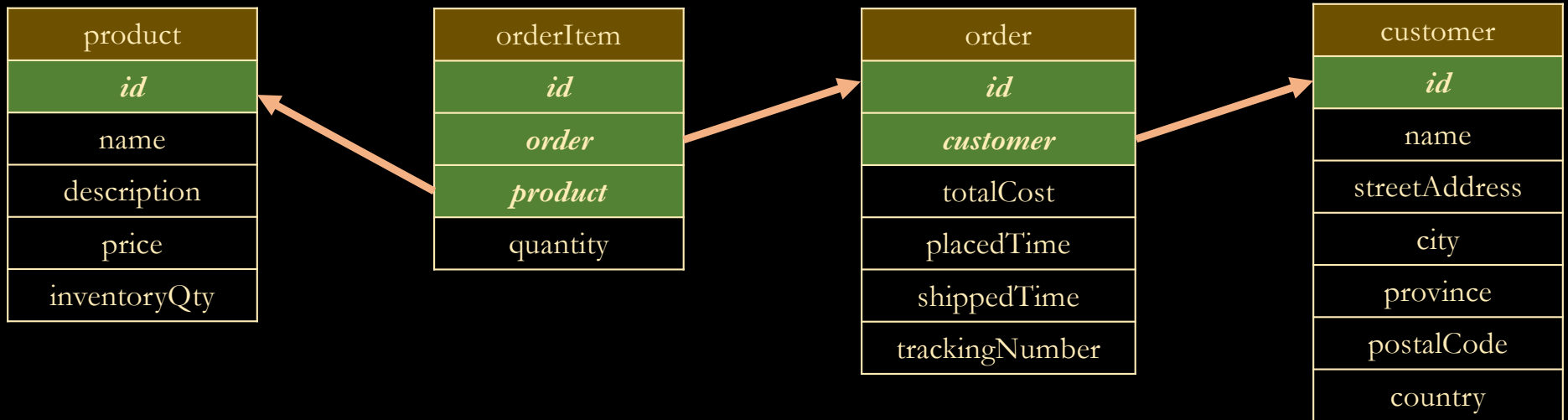
department
<i><u>id</u>: integer</i>
<i>name: varchar(128)</i>
<i>building ID: integer</i>

building
<i><u>id</u>: integer</i>
<i>name: varchar(128)</i>
<i>facilities ext. #: varchar(8)</i>

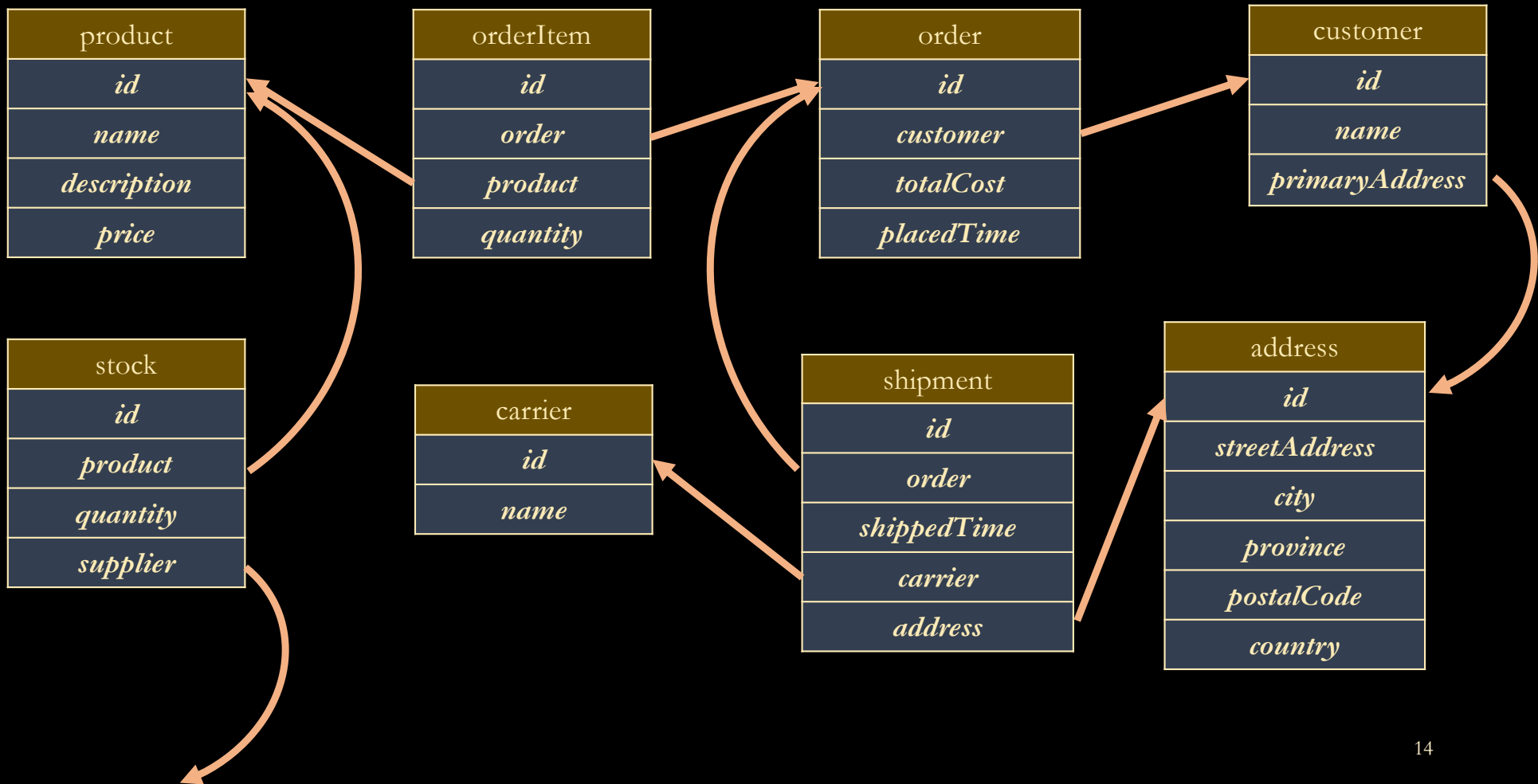
# Online retail example



# Some columns are just *internal references*



# Can make the model more complex



## Part 2 : Primary and Foreign Keys

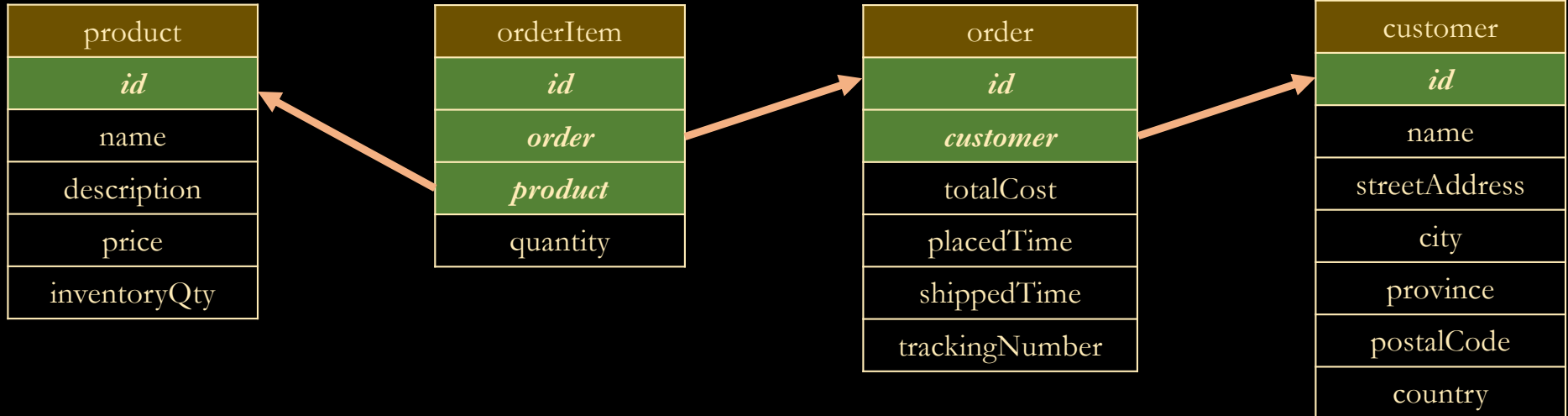
In the previous part, we introduced Relational Databases, defined as:

- A set of tables where each table has:
  - Rows of data (aka *tuples*, *records*)
  - Columns (aka *attributes*, *fields*) defining the data stored in each row
  - A column whose value must be unique for each row (the *primary key*)
- All of the values in a given column must be of the same type, and within a row each column may store only one value
- A column may refer to a row in another table (tables may be linked with *foreign keys*)



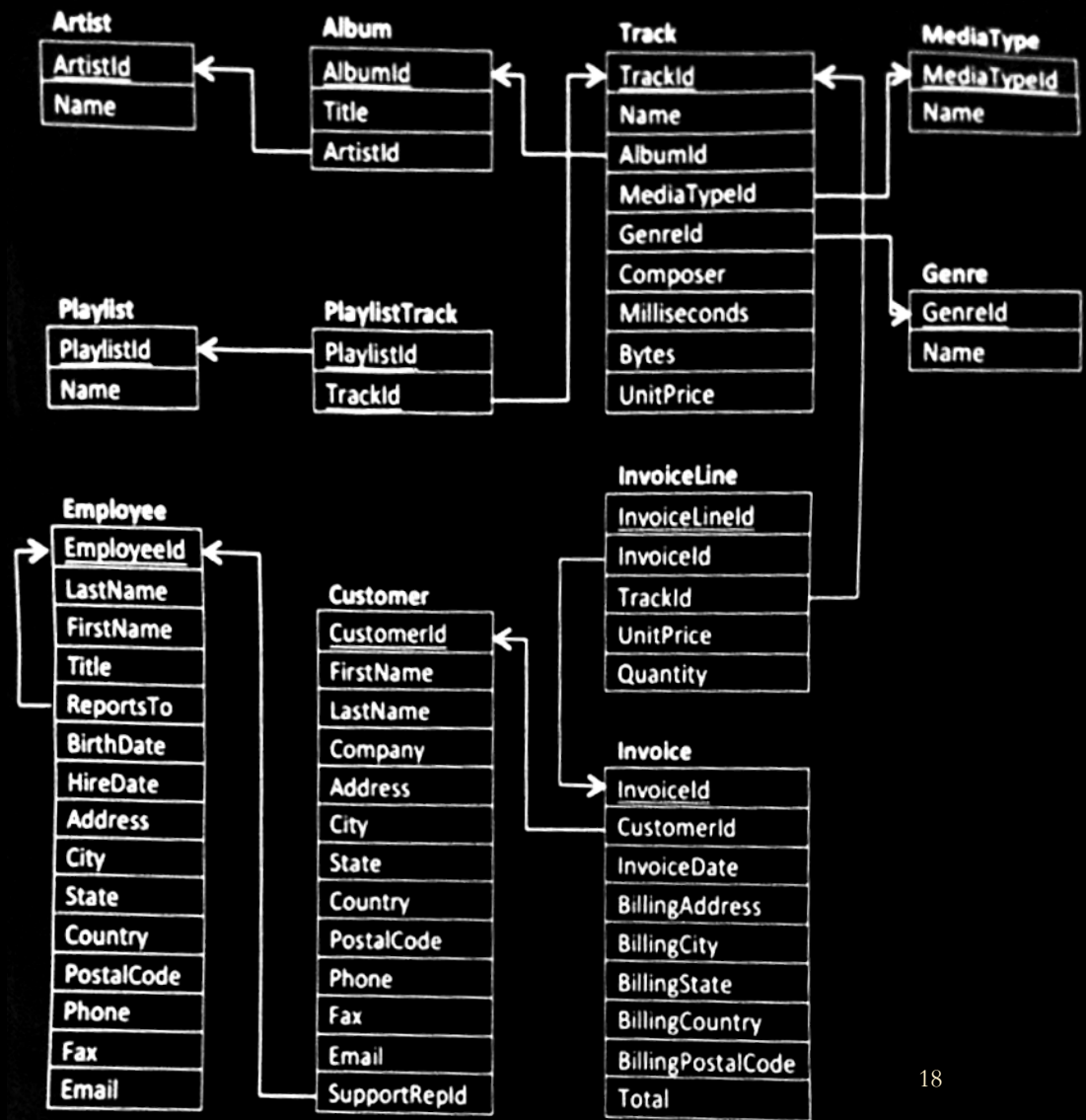
## Previous Part (continued)

- Showed benefits of using multiple tables:
  - Eliminate data redundancy:  
Saves space and allows updates to happen in one place
  - Allows objects, events, and relationships to be added separately
- Showed data model diagrams:



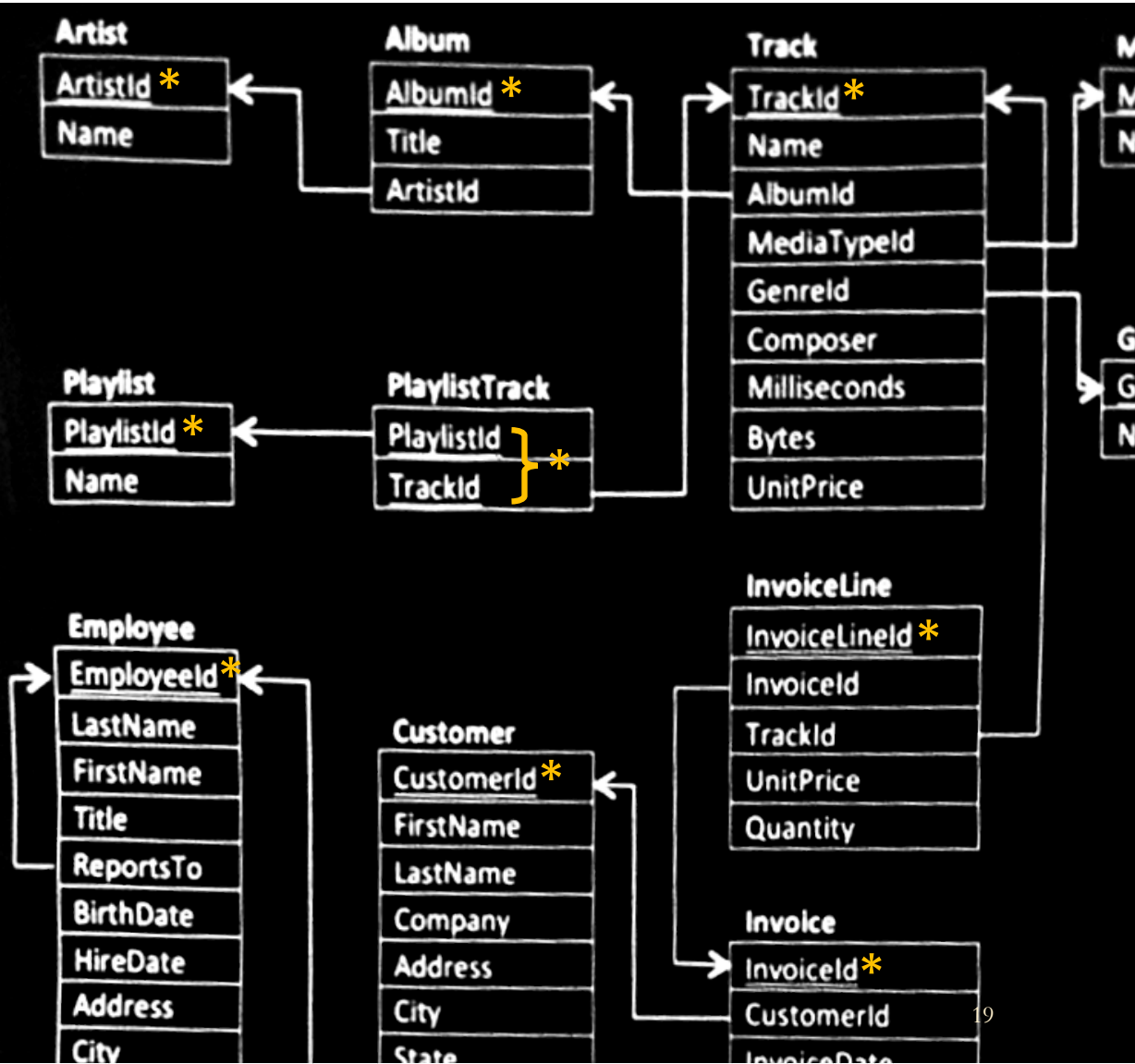
# A more detailed example

- “Chinook” online music store
- We will download and use this database later in the quarter
- 11 tables



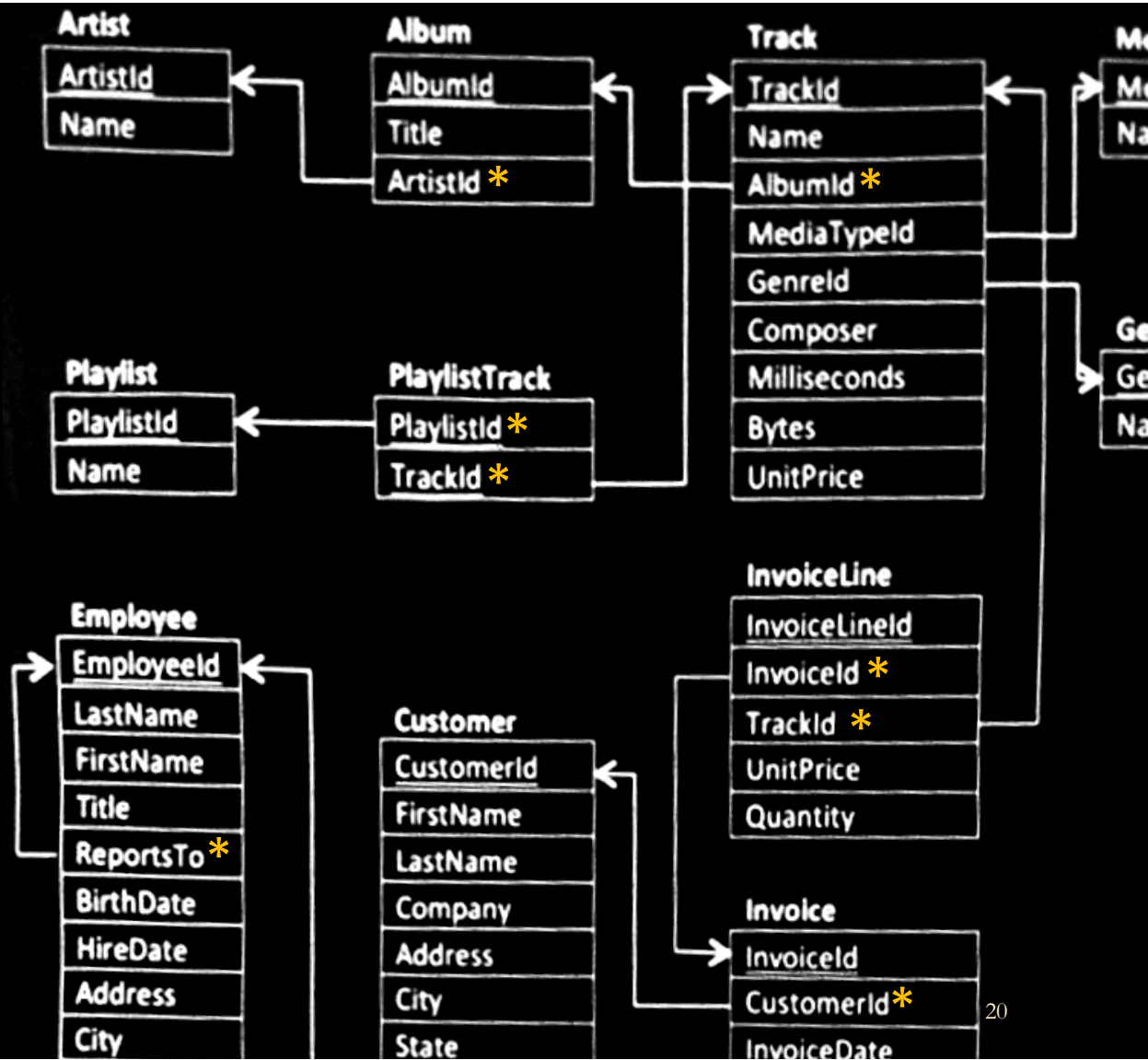
# Primary Keys

- 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
- Usually 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
- Primary key(s) are underlined



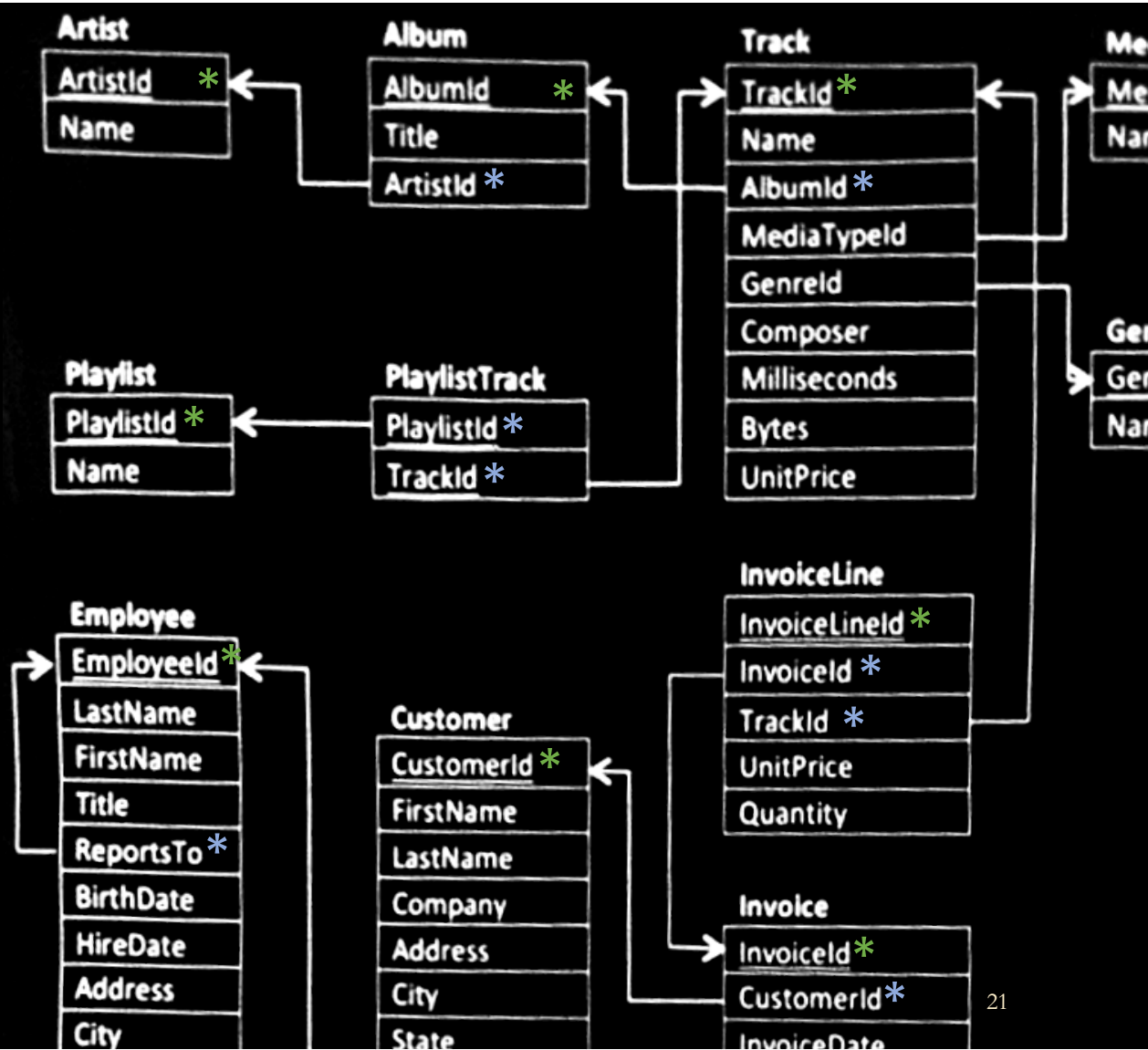
# Foreign Keys

- Tables may be linked by *foreign keys* – columns that refer to keys in other tables
- Usually these are integers ids, and usually refer to primary keys, but not necessarily
- **PlaylistTrack** table is made entirely of foreign keys, so we call it a *linking table*



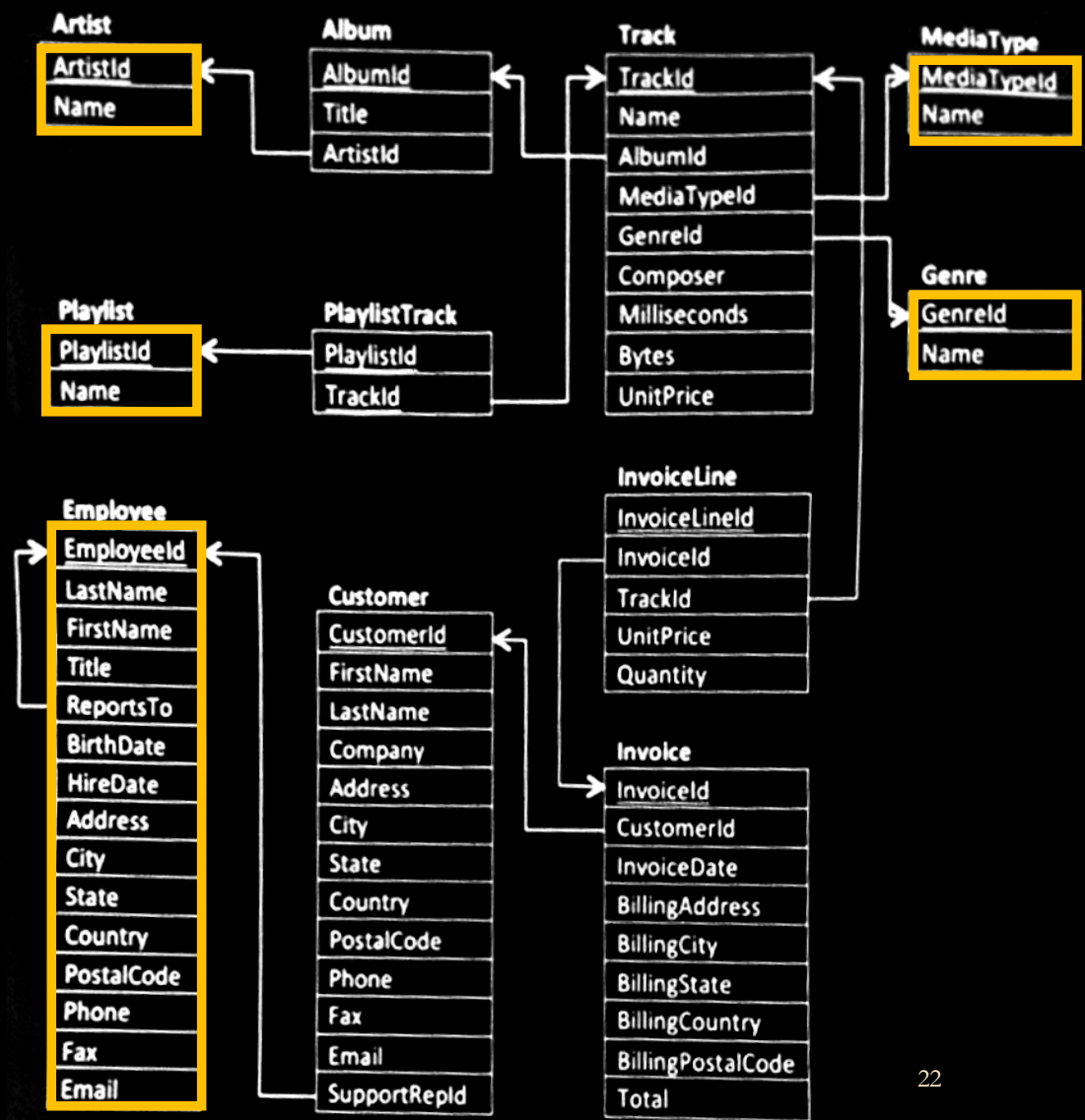
# Parent and Child tables

- Foreign keys define a parent and child table.
  - Child points back to parent
  - Parent row must be created before child row



# 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



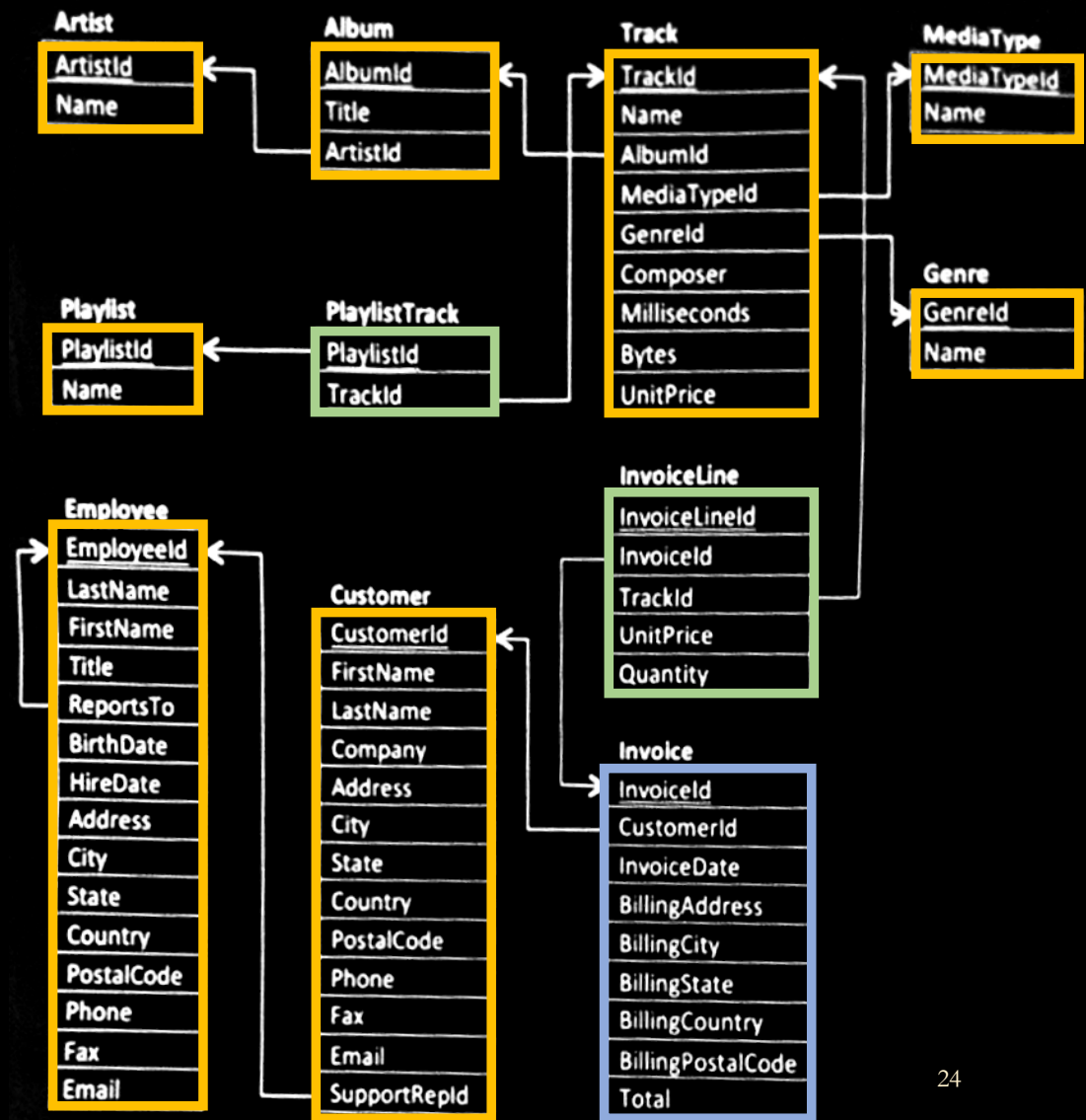
# Optional columns

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



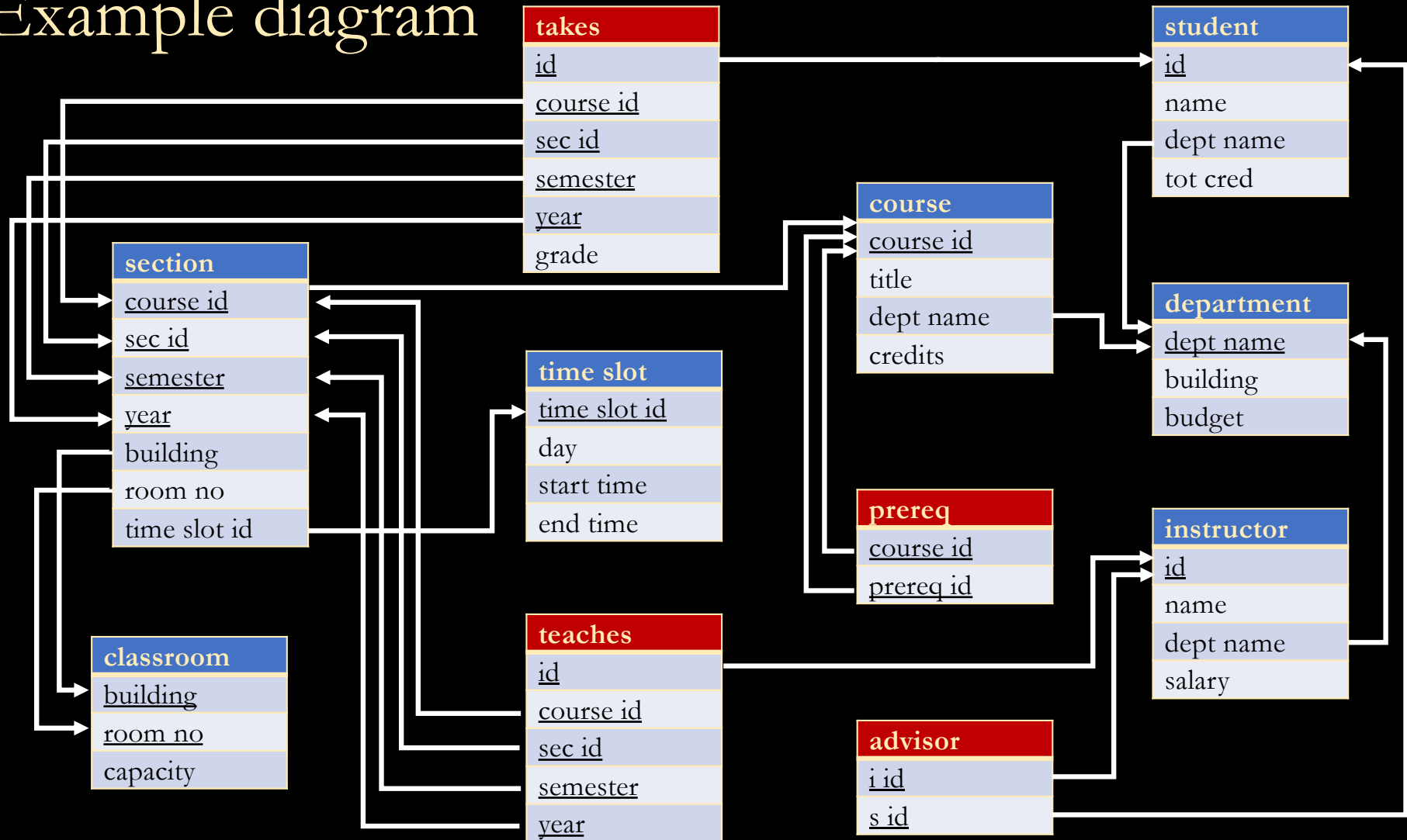
# Objects, Events, and Relationships

- 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





# Example diagram



# Example questions on the previous diagram

- What are the top-level tables?
- Why is prereq in a table instead of a column in “course”?
- Are there any relationship tables here?
- Event tables?
- Composite primary keys?
- Why does “section” have two arrows pointing to “classroom”?
- Is this different than the two arrows pointing from prereq to course?
- In student table: can we remove tot\_cred column without losing data? If dept\_name is optional, how can we move it to an outside table? If student can have multiple majors, what do we do?
- What questions can you answer with this database?

# Optional reading

- Database Design, Chapters 1-3