MLDS-413 Introduction to Databases and Information Retrieval

Lecture 5
ER Modeling; Table Relationships; Intro to SQL

Instructor: Nikos Hardavellas

Slides adapted from Steve Tarzia

Last Lecture

- Defined a relational database
  - A set of tables where each table has:
    - Rows of data, columns defining the data stored in each row
    - A column whose value must be unique for each row (the primary key)
- Defined a database schema
- Primary and foreign keys
- Showed how tables are linked
- *Parent* tables must be filled before *child* tables
- Loosely classified tables as *objects*, *events*, and *relationships*
- Optional columns and *NULL* values
- Database normalization

Policies for deleting Foreign Keys

- Logically, you cannot delete a row from a parent table if a child table refers to it
- 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)

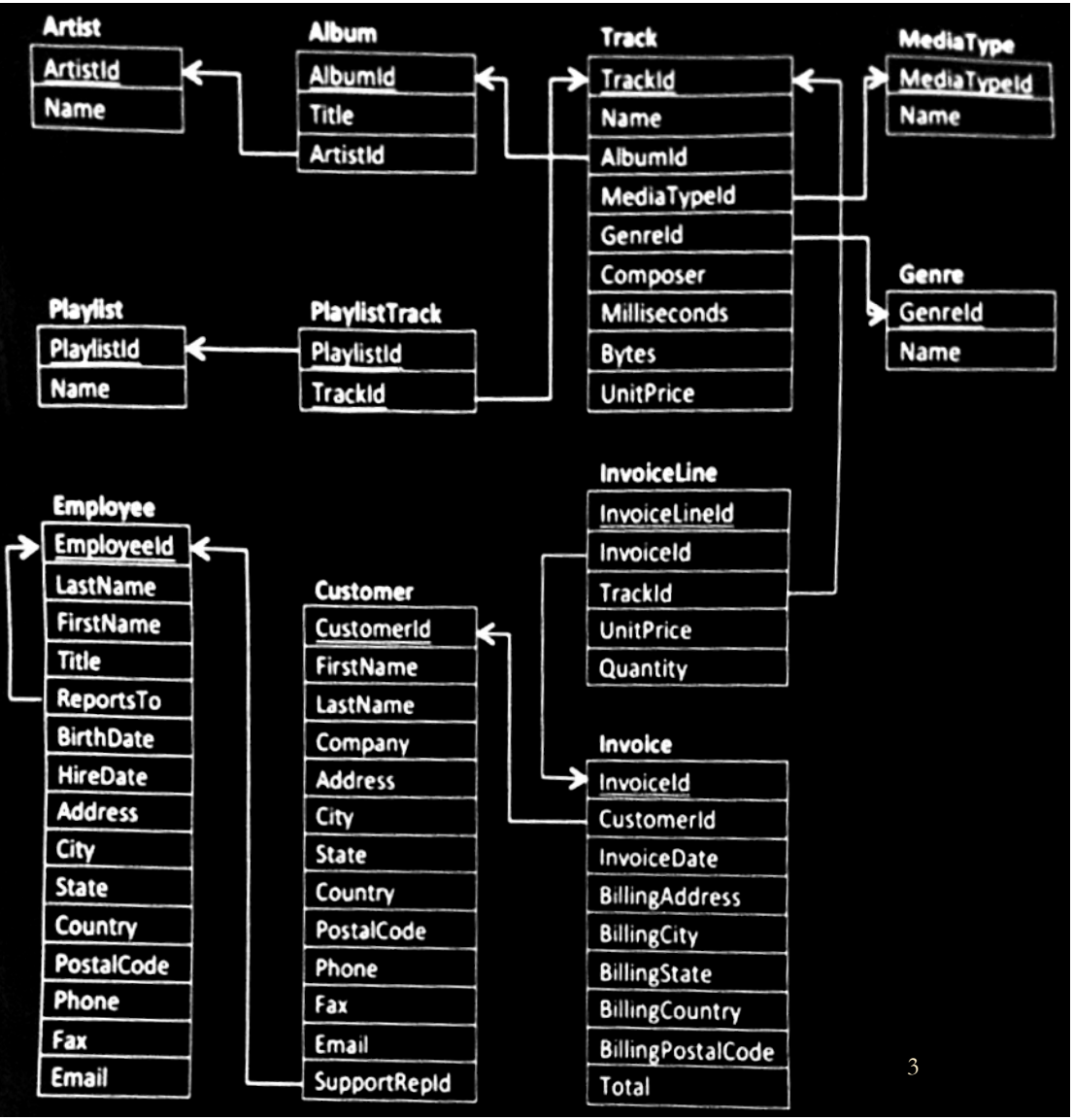


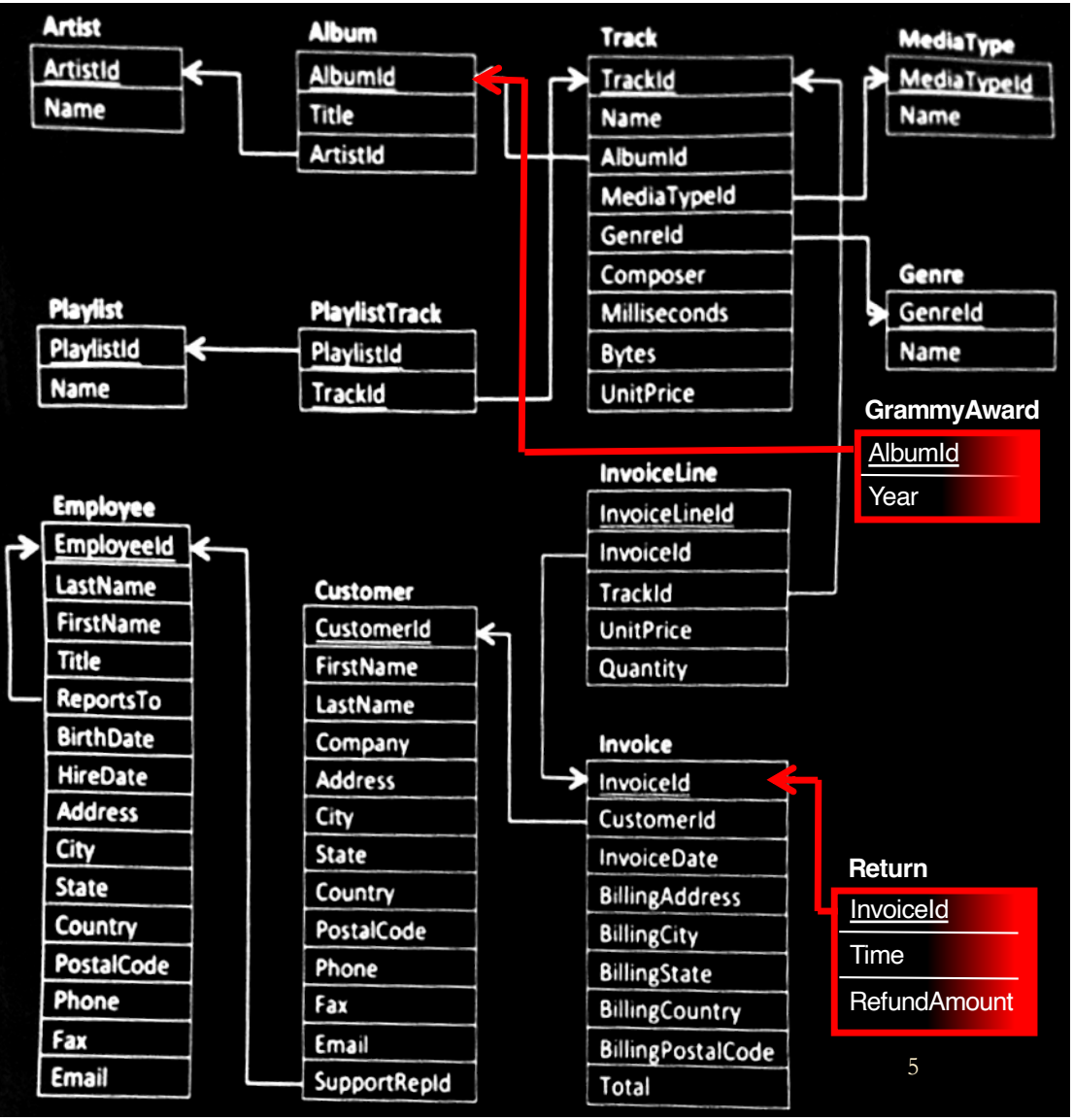
Table relationships in depth

Foreign keys can relate table rows in three ways:

- One-to-One
- One-to-Many (or Many-to-One)
- Many-to-Many

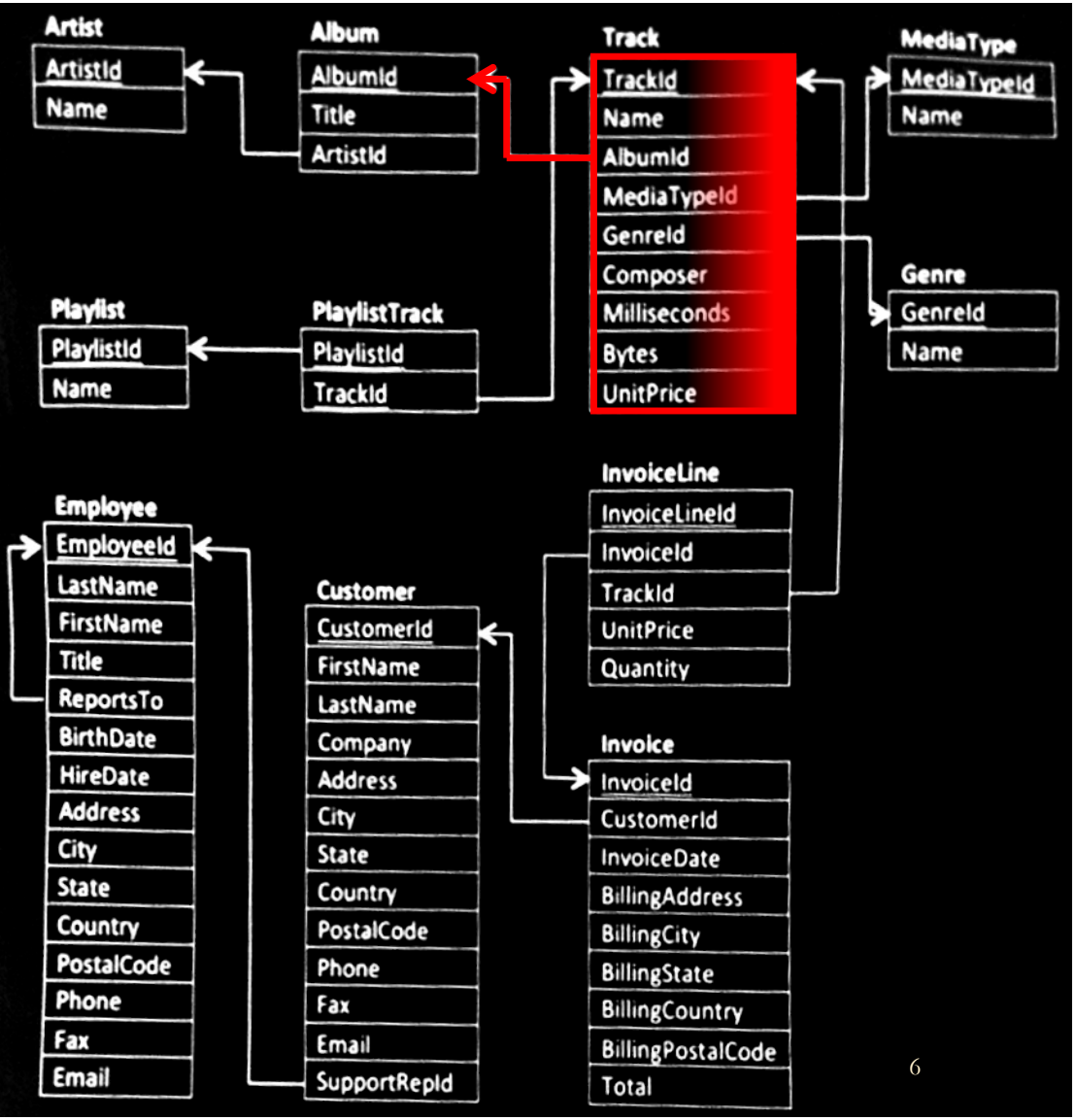
One to One

- One-to-one relationships exists **when a primary key is also a foreign key**.
- In other words, there is an arrow pointing from one primary key to another.
- The child table is a *subset table*.
- Subset tables are an alternative to having optional columns in the parent table.
- Note: the notation in this slide is **NOT** a good notation



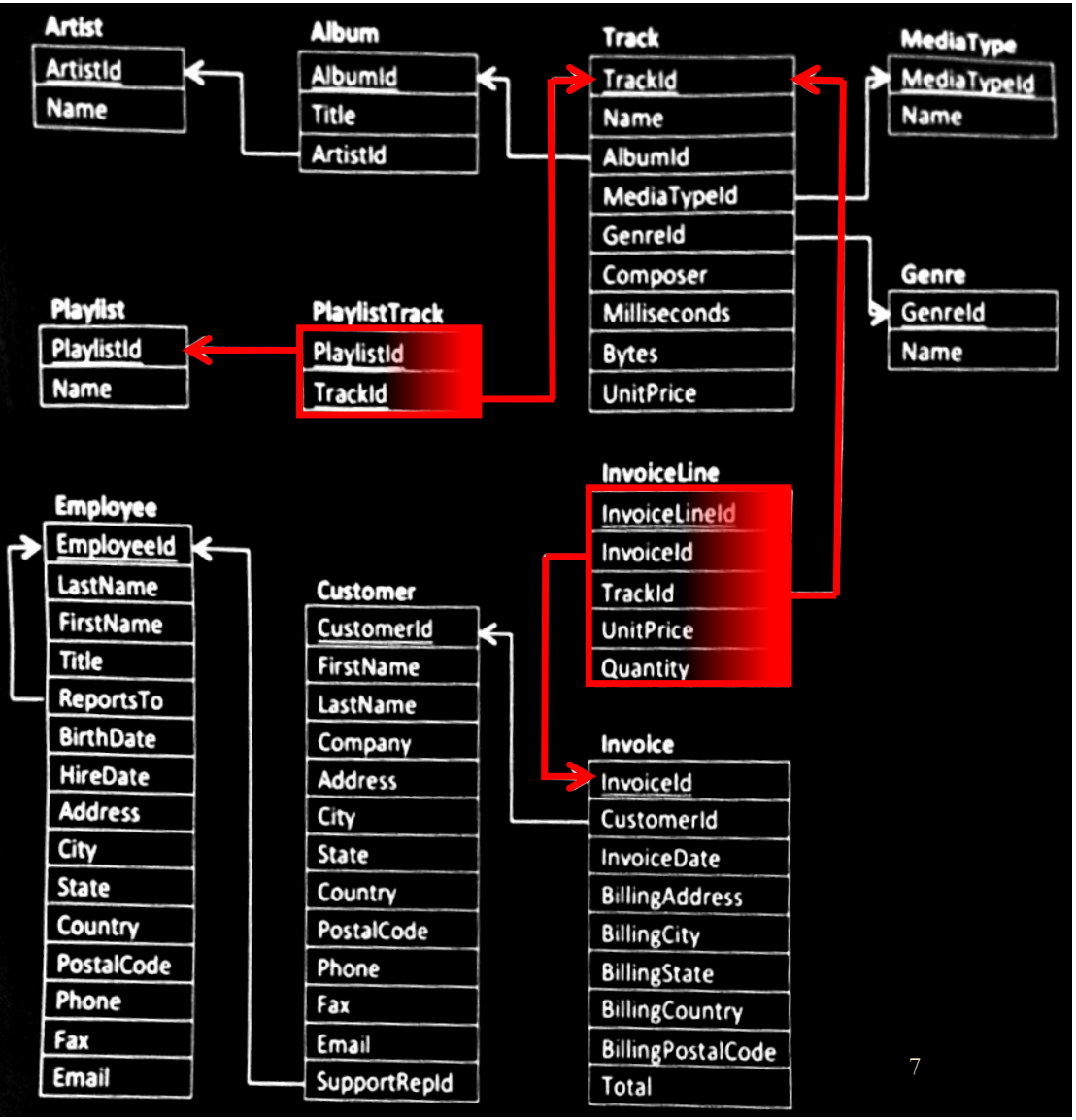
One to Many

- Most foreign keys create one-to-many relationships
- Created when a column that is **not a primary key is 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
- Note: the notation in this slide is **NOT** a good notation



Many to Many

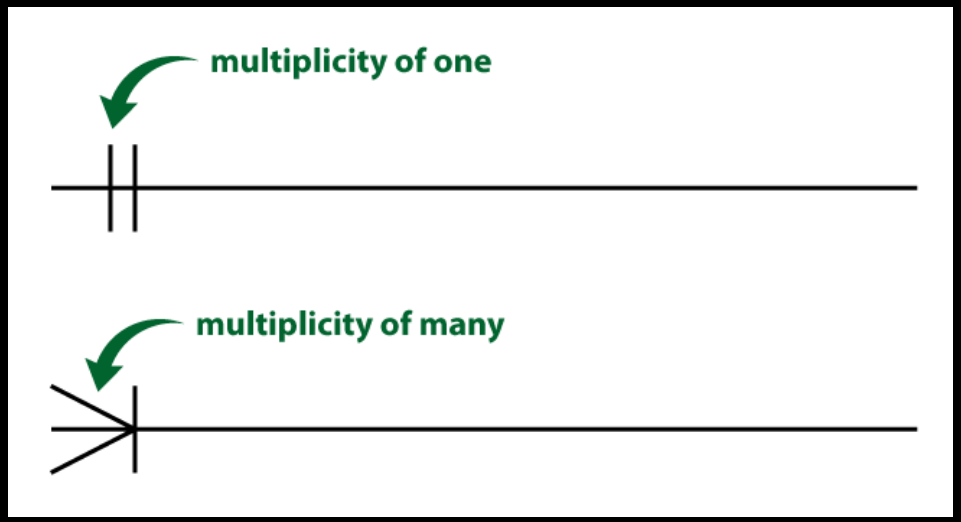
- Two one-to-many relationships starting at the same table can create a many-to-many relationship
- These are represented with *linking tables*.
- But, there are no strict rules.
  - We think of **Track** as a an *object* rather than a many-to-many relationship between albums and genres.
- Note: the notation in this slide is **NOT** a good notation



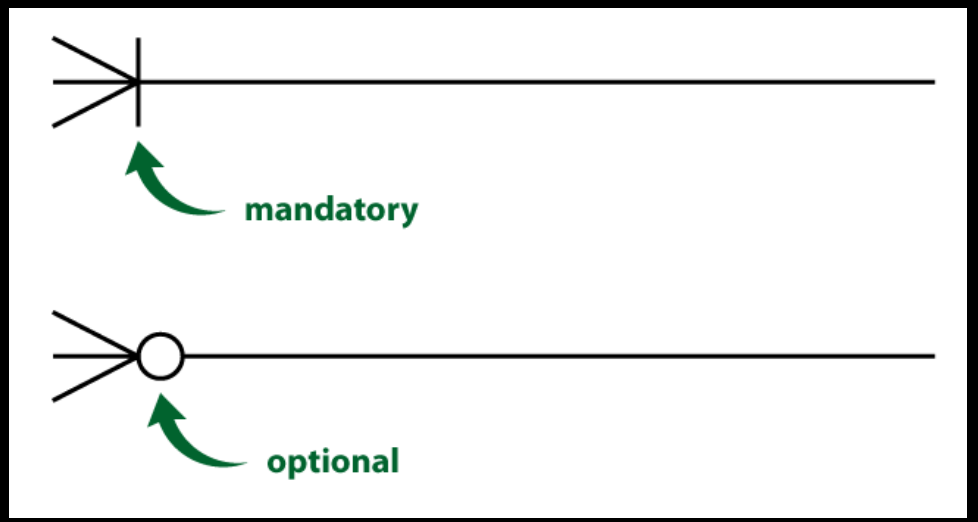
Crow's foot notation of ER diagrams

- Two positions on each line that links two tables
  - One position to denote cardinality (i.e., multiplicity)
  - One position to denote participation
  - Special symbols used for each possible value at each position
- Succinct and clean notation of table relationships, used by SQLite

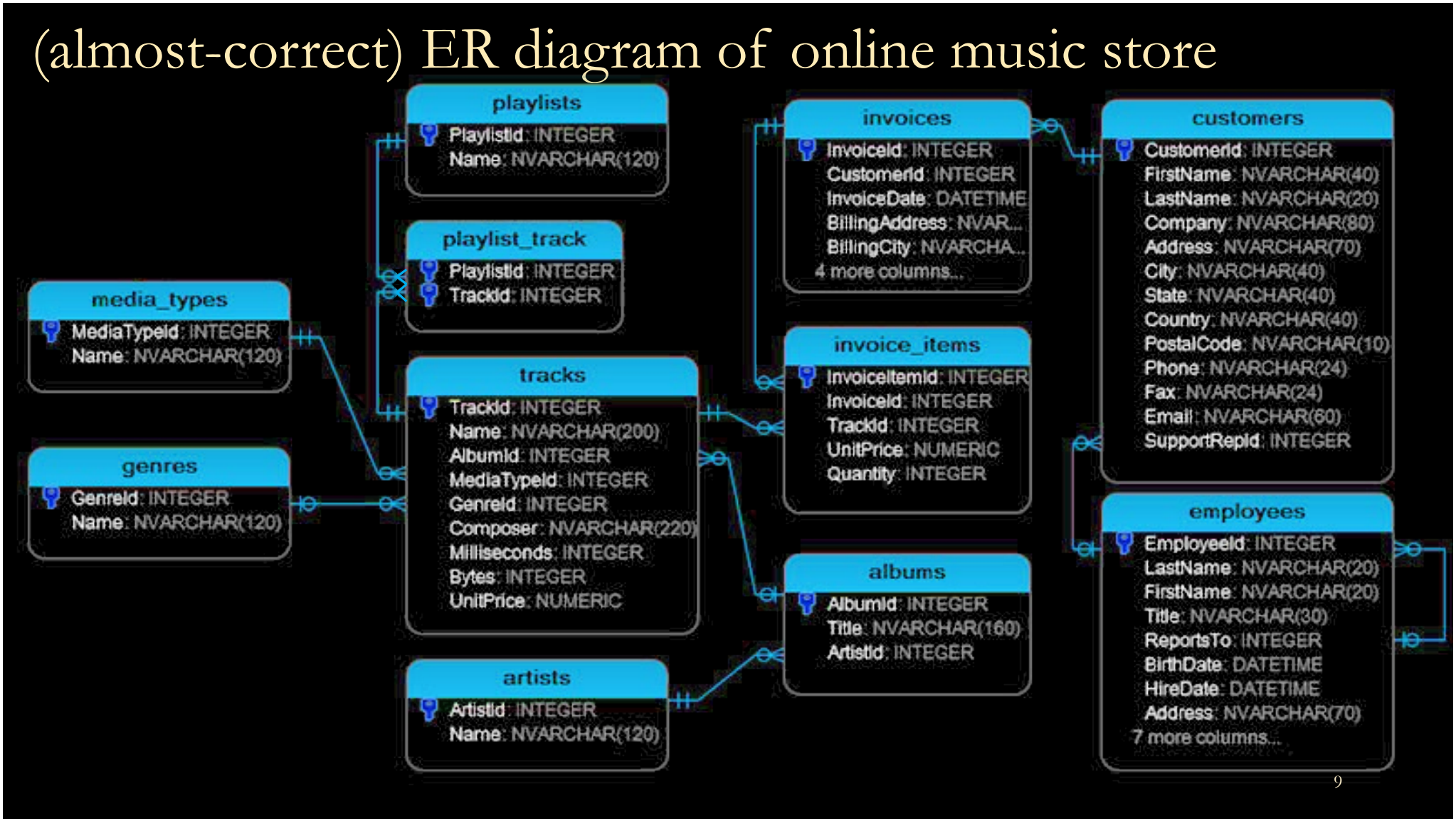
Cardinality constraints



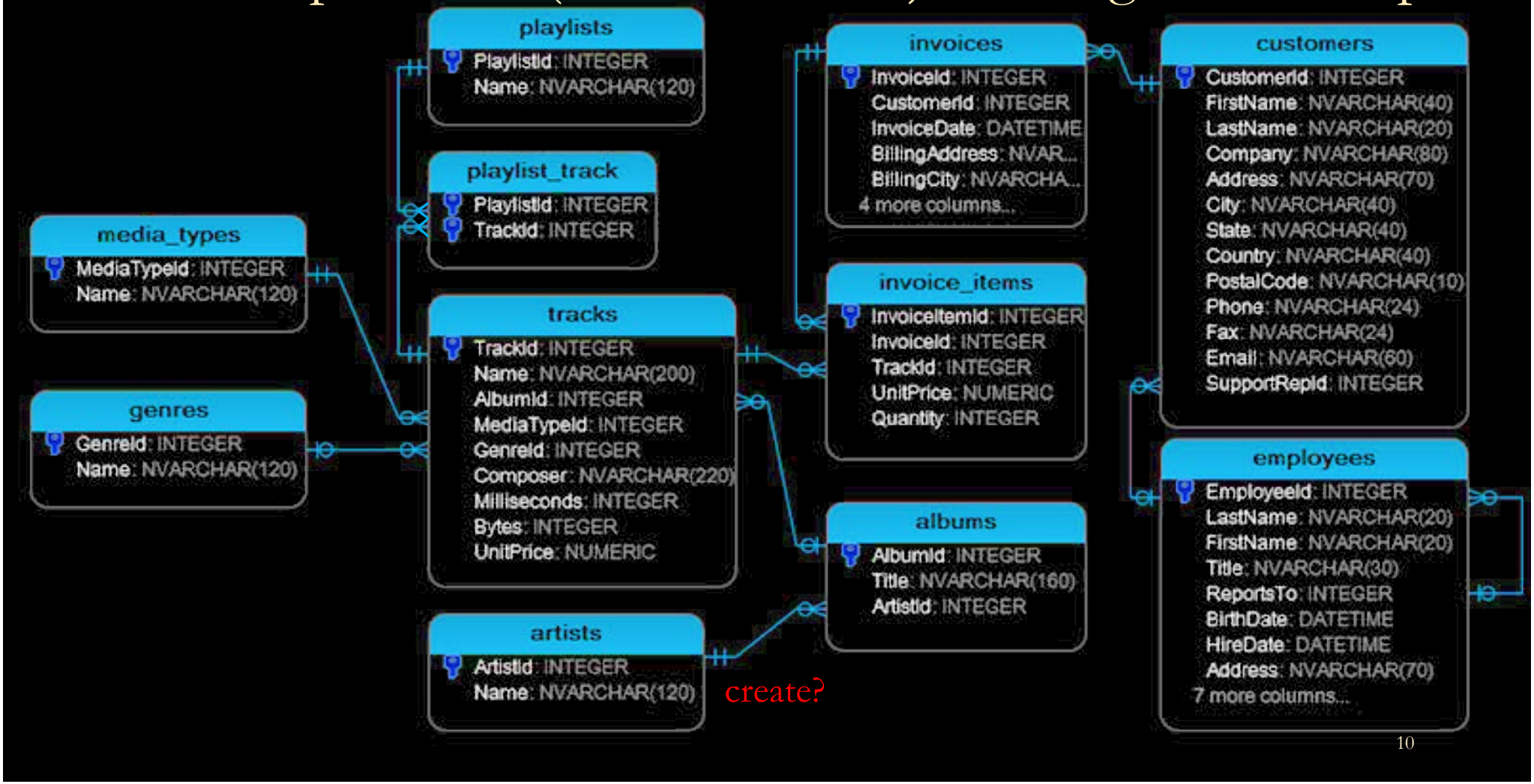
Participation constraints



(almost-correct) ER diagram of online music store

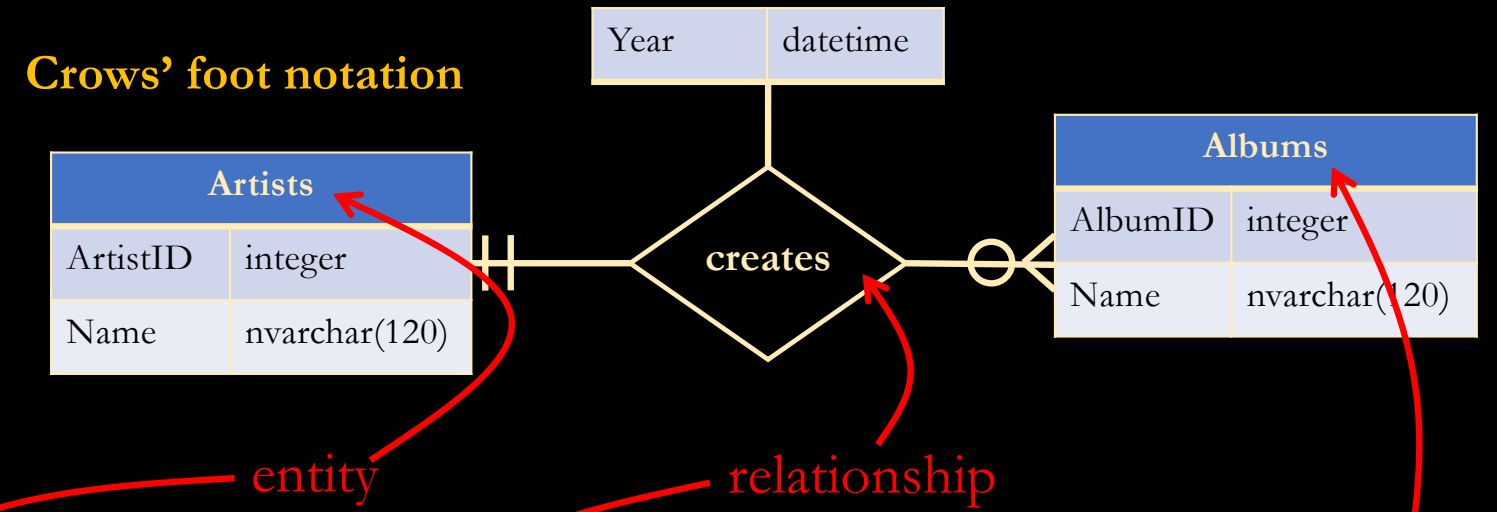


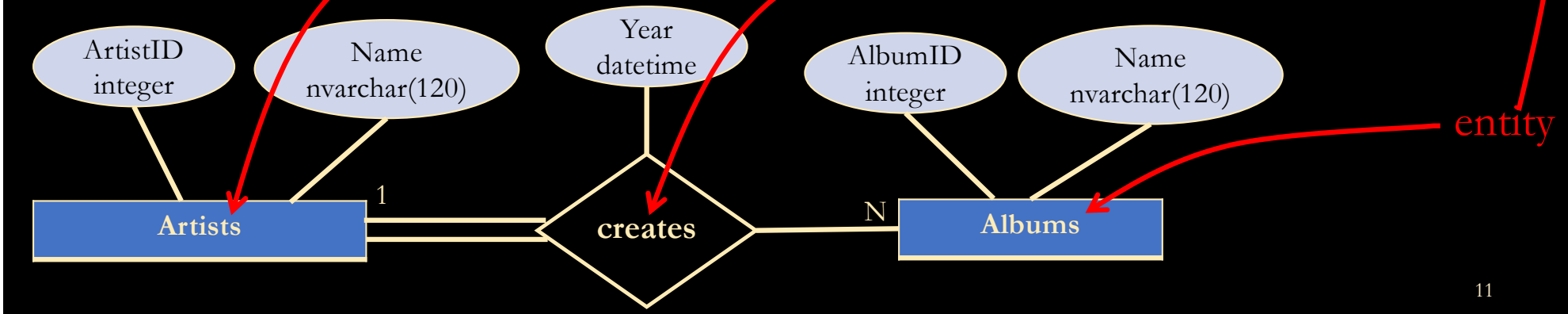
Relationships in this (almost-correct) ER diagram are implicit



Relationships in fully-correct ER diagrams are explicit

- Association of two or more entities
- Can have its own attributes
- Explicit is cleaner





Instance of a relationship

- Entities and relationships are just tables of a database
- The relationship table links two (or more) entities by using their primary keys as foreign keys
- The relationship's primary key is a composite key
- Rows are also called *records* or *tuples*

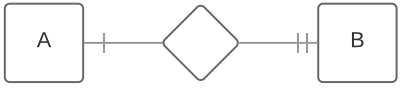
Artists	
ArtistID	**Name**
1	Bob
2	Kate
3	J.J.

Creates		
ArtistID	**Year**	**AlbumID**
1	1992	2
3	2011	1
2	2018	3

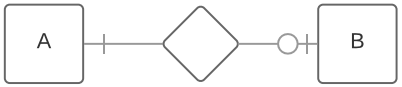
Albums	
AlbumID	**Name**
1	Sunset
2	Cloudy
3	Sunny

ER Diagrams cheat sheet

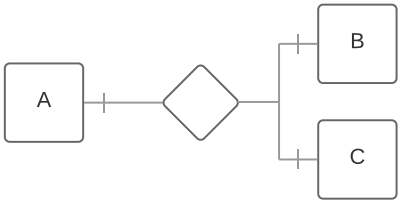
One A is associated with
one B



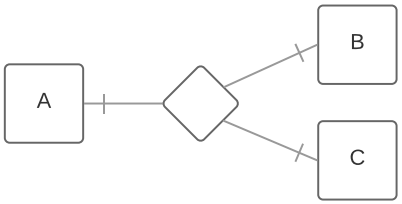
One A is associated with
one and only one B



One A is associated with
zero or one B

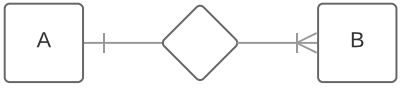


One A is associated with
one B and one C (both)
-- mutually inclusive association

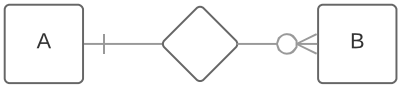


One A is associated with
one B and one C
-- N-ary relationship

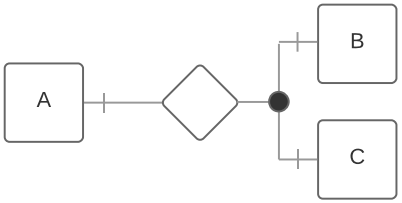
One A is associated with
many B



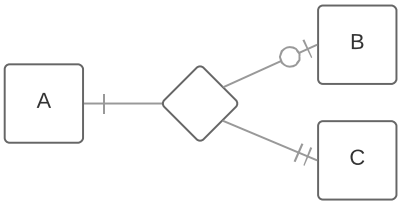
One A is associated with
one or many B



One A is associated with
zero, one or many B



One A is associated with
one B or one C (but not both)
-- mutually *exclusive association*



One A is associated with
zero or one B and exactly one C
-- N-ary relationship

Each DB management system has its own slightly different set of types

- Previous slide showed types used in SQLite
- Generally the following are always available:
  - Integer (32 and 64 bit)
  - Fixed point (usually called “numeric”)
  - Floating point (32 and 64 bit)
  - Text (usually called “char” or “varchar”) of various length
  - Binary (usually called “blob”) of various length

Structured Query Language (SQL)

- The standard programming language for relational databases
- Each DB Management System (DBMS) has its own dialect
  - In this course we will be using MySQL and SQLite's variants of SQL
- SQL is a *declarative* language (most other languages are imperative)
  - You describe the results you want to see
  - You do not describe the detailed steps necessary to gather those results
- We will be using a client program to connect to the DBMS and running SQL statements *interactively*:
  - run one statement and look at the results before running another one

SELECT gets data

```
SELECT FirstName, LastName FROM customers WHERE City = "Paris";
```

Columns to print — Table to examine — Filter

Result is a table with two rows:

FirstName	*LastName*
Camille	Bernard
Dominique	Lefebvre

Filtering, sorting, and limiting

We can use more complex filters:

```
SELECT FirstName, LastName FROM customers
     WHERE City = "Chicago"
                AND (State = "Illinois"
                      OR State = "IL");
```

Get all columns, sort the results (descending) and limit the results to just the first ten rows:

```
SELECT * FROM tracks ORDER BY UnitPrice DESC LIMIT 10;
```

Arithmetic

Your SELECT statements can include arithmetic

```
SELECT Name, UnitPrice / (Milliseconds/1000/60)
    AS PricePerMinute FROM tracks;
```

Check your DBMS's documentation for the specific math functions

Grouping

The GROUP BY clause combines multiple rows and lets you perform aggregation math functions

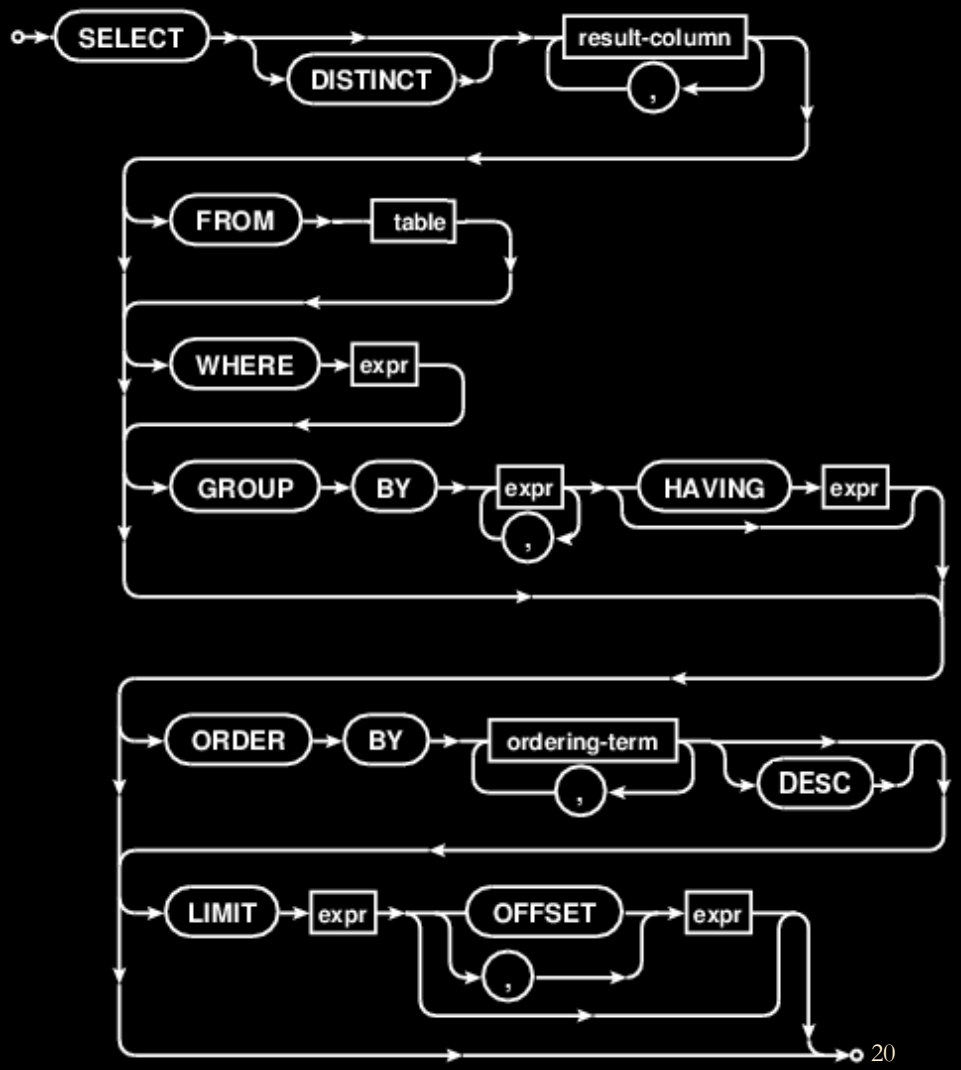
```
SELECT AlbumId,
    SUM(Milliseconds/1000/60) AS AlbumMinutes
  FROM tracks GROUP BY AlbumId ORDER BY AlbumMinutes;
```

Result:

AlbumId	*AlbumMinutes*
340	0.86300000
345	1.11065000
318	1.68821667
…	…

Syntax diagrams

- Any path from start to end is a valid SQL statement
- Choose which arrows to follow
- The rectangles refer to other diagrams
- Used by our SQL book
- Used by SQLite online docs: https://sqlite.org/lang.html



Syntax grammars

- A set of rules for building all possible statements
- Used by MySQL docs
- Optional items are in square braces: `[ ]`
- Pipe character for "or": `this | that`
- Curly braces for a required choice: `{one | two}`
- `...` for repetition
- Lowercase italics for things defined elsewhere.

```
SELECT
  [ALL | DISTINCT | DISTINCTROW ]
    [HIGH_PRIORITY]
    [STRAIGHT_JOIN]
    [SQL_SMALL_RESULT] [SQL_BIG_RESULT]
    [SQL_BUFFER_RESULT]
    [SQL_CACHE | SQL_NO_CACHE][SQL_CALC_FOUND_ROWS]
  select_expr [, select_expr ...]
  [FROM table_references
    [PARTITION partition_list]
  [WHERE where_condition]
  [GROUP BY {col_name | expr | position}
    [ASC | DESC], ... [WITH ROLLUP]]
  [HAVING where_condition]
  [ORDER BY {col_name | expr | position}
    [ASC | DESC], ...]
  [LIMIT {[offset,] row_count | row_count OFFSET offset}]
  [PROCEDURE procedure_name(argument_list)]
  [INTO OUTFILE 'file_name'
      [CHARACTER SET charset_name]
      export_options
    | INTO DUMPFILE 'file_name'
    | INTO var_name [, var_name]]
  [FOR UPDATE | LOCK IN SHARE MODE]]
```

Optional reading

- Database Design, Chapters 3, 7–10