Lecture 3 ER/Class Diagram
SQL: CRUD

CPE241 Database Systems

27 January 2026

Dr. Jaturon Harnsomburana
Dr. Piyanit Ua-areemitr

Department of Computer Engineering
King Mongkut's University of Technology Thonburi

Today Goals

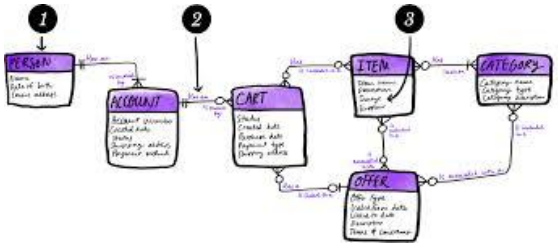
- ERD concepts
- Notations
- Basics ERDs
- Key Constraints
- DB Schemas
- SQL:CRUD
- Exercise: from requirements to tables

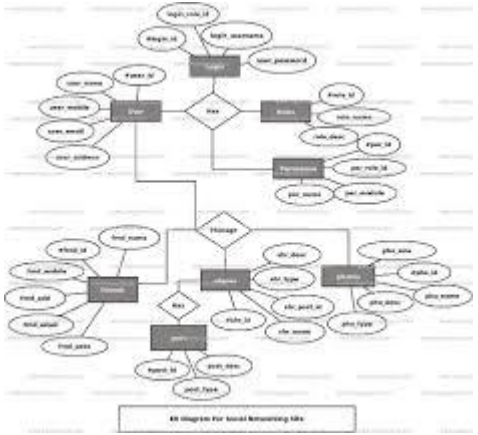
What is Entity-Relationship Diagram

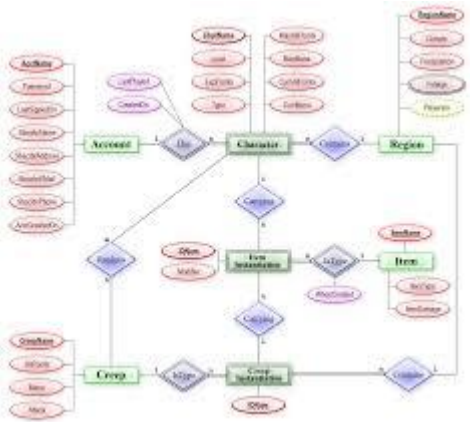
A diagram that shows relationships among entities

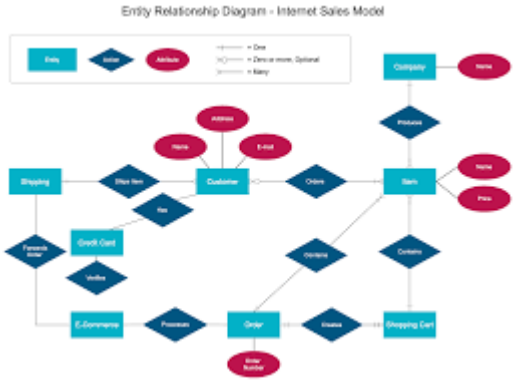
What is :-

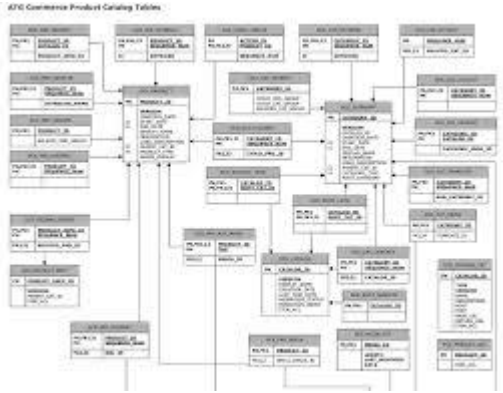
- Entity
- Relationship











Why do we need ER diagram?

The problem with relational schemas

Artists(ArtistId, Name)
Albums(AlbumId, Title, ArtistId)
Tracks(TrackId, Name, AlbumId, MediaTypeId, GenreId, Composer,
Milliseconds, Bytes, UnitPrice)
Genres(GenreId, Name)
MediaTypes(MediaTypeId, Name)
Playlists(PlaylistId, Name)
PlaylistTrack(PlaylistId, TrackId)
Customers(CustomerId, FirstName, LastName, Company, Address, City,
State, Country, PostalCode, Phone, Fax, Email, SupportRepId)
Invoices(InvoiceId, CustomerId, InvoiceDate, BillingAddress, BillingCity,
BillingState, BillingCountry, BillingPostalCode, Total)
InvoiceLines(InvoiceLineId, InvoiceId, TrackId, UnitPrice, Quantity)

The Purpose of ERDs

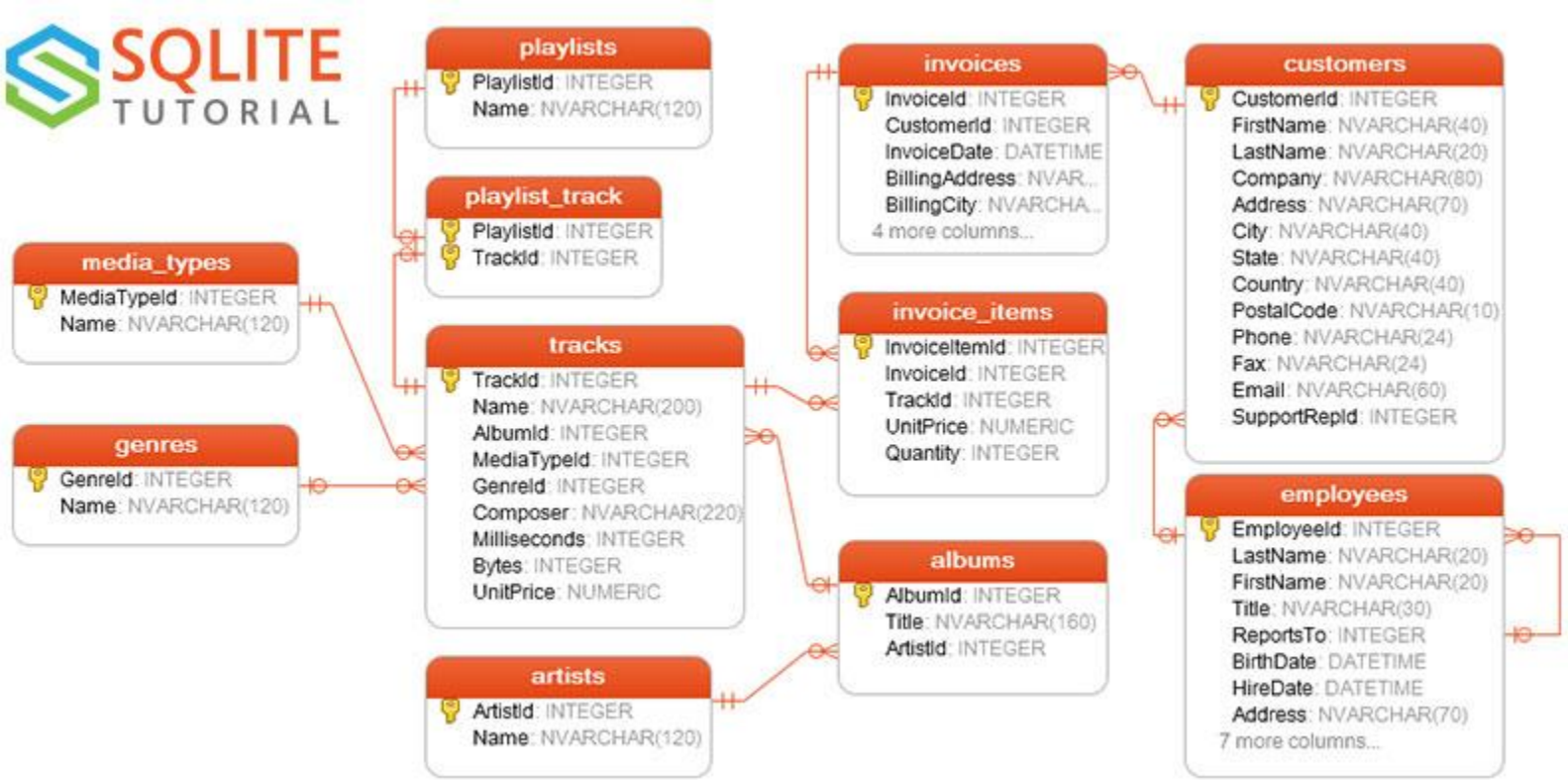
The primary purpose

a visual way to

- Understand,
- Design
- Communicate

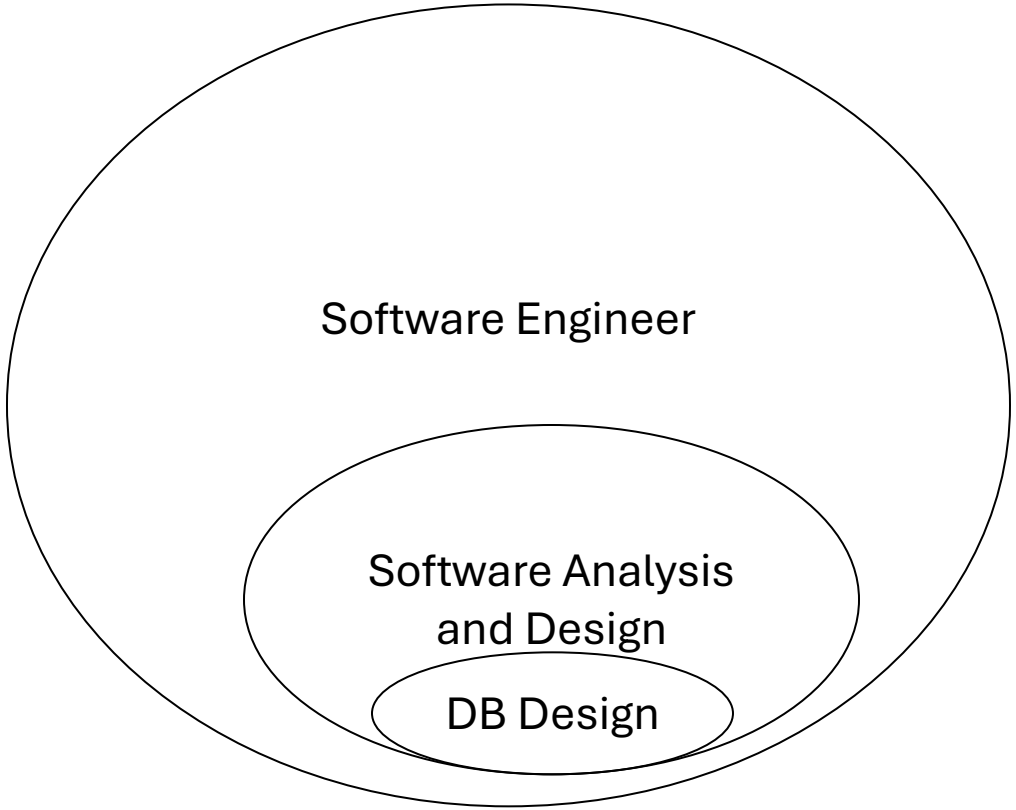
the database structure.

Chinook ERD


SQLITE TUTORIAL
playlists
PlaylistId: INTEGER
Name: NVARCHAR(120)
playlist_track
PlaylistId: INTEGER
TrackId: INTEGER
media_types
MediaTypeId: INTEGER
Name: NVARCHAR(120)
genres
GenreId: INTEGER
Name: NVARCHAR(120)
tracks
TrackId: INTEGER
Name: NVARCHAR(200)
AlbumId: INTEGER
MediaTypeId: INTEGER
GenreId: INTEGER
Composer: NVARCHAR(220)
Milliseconds: INTEGER
Bytes: INTEGER
UnitPrice: NUMERIC
artists
ArtistId: INTEGER
Name: NVARCHAR(120)
invoices
InvoiceId: INTEGER
CustomerId: INTEGER
InvoiceDate: DATETIME
BillingAddress: NVAR...
BillingCity: NVARCHA...
4 more columns...
invoice_items
InvoiceItemId: INTEGER
InvoiceId: INTEGER
TrackId: INTEGER
UnitPrice: NUMERIC
Quantity: INTEGER
albums
AlbumId: INTEGER
Title: NVARCHAR(160)
ArtistId: INTEGER
customers
CustomerId: INTEGER
FirstName: NVARCHAR(40)
LastName: NVARCHAR(20)
Company: NVARCHAR(80)
Address: NVARCHAR(70)
City: NVARCHAR(40)
State: NVARCHAR(40)
Country: NVARCHAR(40)
PostalCode: NVARCHAR(10)
Phone: NVARCHAR(24)
Fax: NVARCHAR(24)
Email: NVARCHAR(60)
SupportRepId: INTEGER
employees
EmployeeId: INTEGER
LastName: NVARCHAR(20)
FirstName: NVARCHAR(20)
Title: NVARCHAR(30)
ReportsTo: INTEGER
BirthDate: DATETIME
HireDate: DATETIME
Address: NVARCHAR(70)
7 more columns...

Process of DB design

1. Requirement Analysis
2. Conceptual Design
3. Logical Design
4. Physical Design
5. Implementation
6. ~~Testing~~
7. Deployment
8. Maintenance
9. Documentation
10. Review and Iteration



Conceptual/Logical/Physical Data Model

Conceptual

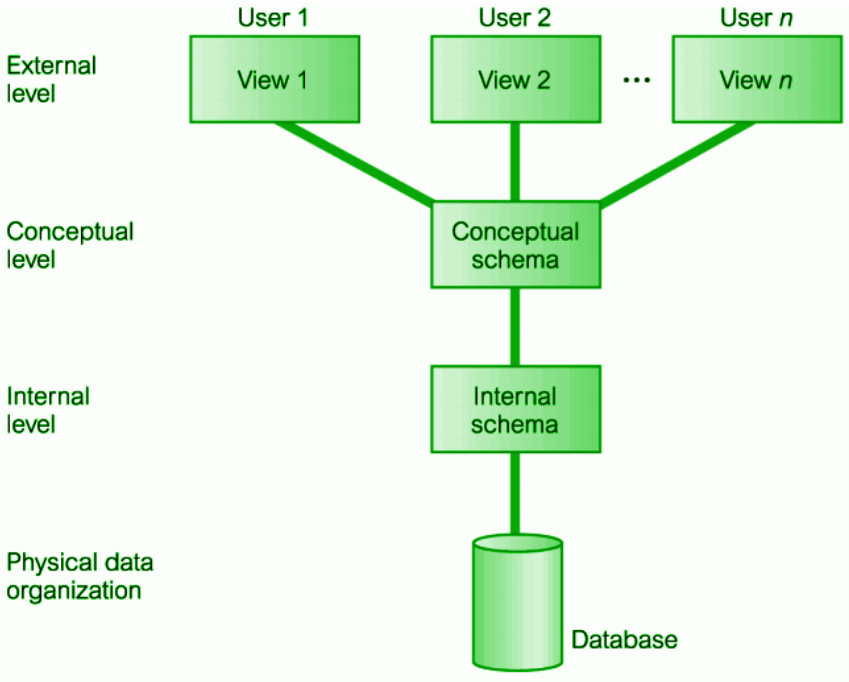
High level for communication

Logical

More detail for correctness

Physical

For implementation

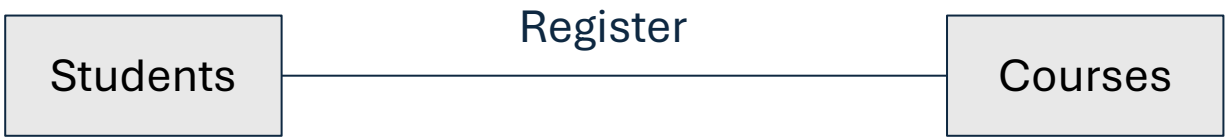


Conceptual Design

Identify Entities and Attributes: Determine the main objects (entities) and their characteristics (attributes) that the database must store.

Define Relationships: Specify how entities are related to each other, including cardinality (e.g., one-to-many, many-to-many).

Create an ER Diagram: Use tools to visually represent entities, attributes, and relationships using an **Entity-Relationship Diagram (ERD)**.



Logical Design

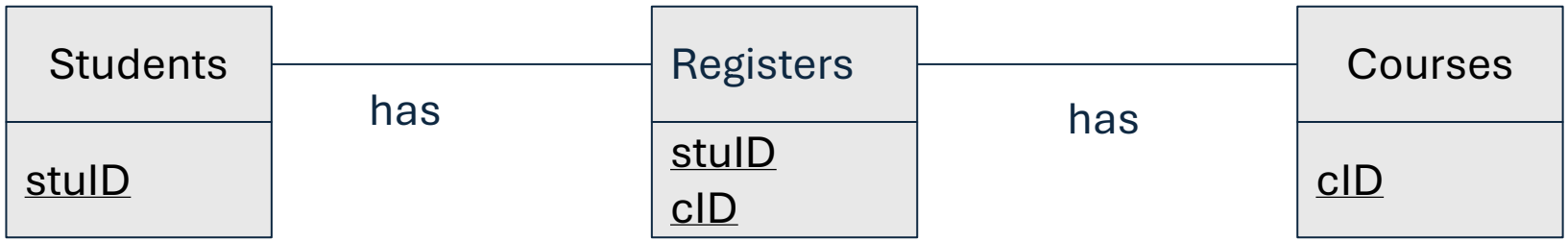
Convert ERD to Schema: Translate the conceptual model into a logical schema that adheres to the chosen database model (e.g., relational, NoSQL).

Normalize Data:

- Apply normalization rules to eliminate data redundancy.
- Ensure data integrity by dividing data into smaller, related tables (e.g., 1NF, 2NF, 3NF).

Define Data Types and Keys:

- Specify data types for attributes.
- Identify primary and foreign keys for each table.



Physical Design

Choose Database Management System (DBMS): Select the appropriate DBMS (e.g., MySQL, PostgreSQL, MongoDB) based on requirements.

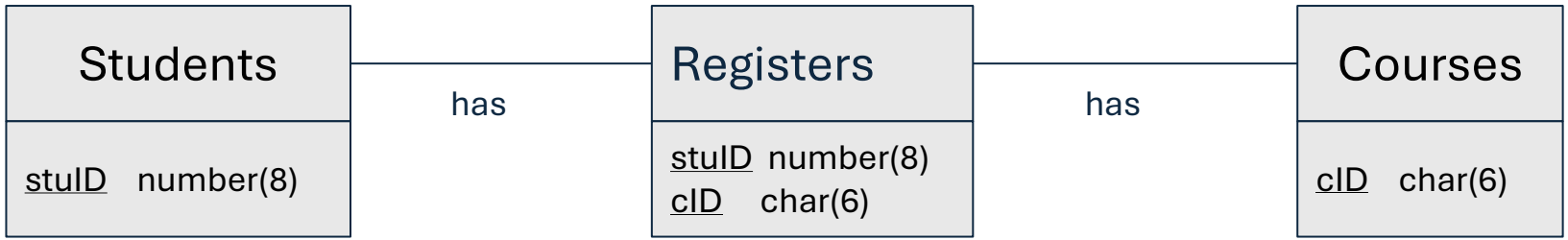
Optimize Storage:

- Decide on indexing strategies for faster queries.
- Partition data if necessary for scalability.

Define Constraints:

- Enforce constraints like uniqueness, not null, and foreign key references.

Set Up Security: Plan user roles, permissions, and access controls.



Documentation

Create Design Documentation: Include details of the schema, relationships, and business rules.

Update Documentation: Ensure all changes to the database are reflected in the documentation for future reference.

ERD Components

Entities: Objects or concepts (rectangles).

Attributes: Properties of entities (ovals or listed inside entities).

Relationships: Connections between entities (diamonds or lines).

Primary Keys: Unique identifiers (underlined or marked "PK").

Foreign Keys: Links between entities (marked "FK").

Cardinality: Specifies occurrences (1:1, 1:N, M:N).

Weak Entities: Depend on other entities (double rectangles).

Cardinality and Modality

Cardinality

- **Definition**: Cardinality specifies the **number of instances** of one entity that can or must be associated with instances of another entity in a relationship.
- **Types**:
    1. **One-to-One (1:1)**: One instance of Entity A is related to one instance of Entity B.
    2. **One-to-Many (1:N)**: One instance of Entity A is related to many instances of Entity B.
    3. **Many-to-Many (M:N)**: Many instances of Entity A are related to many instances of Entity B.

Modality

- **Definition**: Modality defines whether a relationship is **mandatory or optional**, describing the **minimum participation** of an entity in a relationship.
- **Types**:
    1. **Mandatory (Modality = 1)**: An entity must participate in the relationship.
    2. **Optional (Modality = 0)**: An entity can participate, but it is not required.

Notations

Chen

Easy to draw, but some say not intuitive.

Crow’s foot

Need learning

UML Class Diagram

Used in software design and development

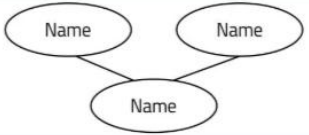
https://medium.com/@ericgcc

Chen and Crow’s foot are equivalent, they show the same information. The different is just drawing

Class diagram is different. It is used for software development. But people use it in NoSQL. It has similar information ERD has, but some are missing.

Chen Notation : Entity & Attribute

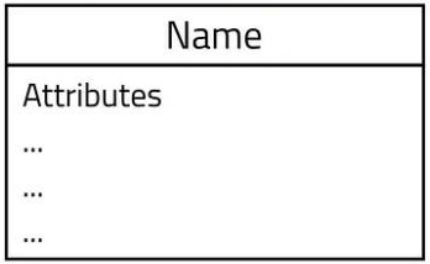
Entity	
Weak Entity	
Attribute (mandatory)	
Pimary identifier attribute	
Partial identifier attribute	

Multi-valued attribute	
Derived attribute	
Composite attribute	

Chen Notation : Relationship

Relationship	
Weak Relationship	
Optional symbol	0
One symbol	1
Many symbol	M

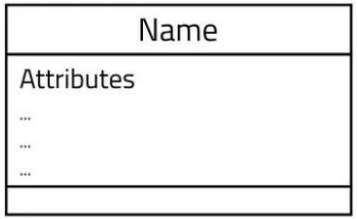
Crow’s Foot Notation : Entity & Attribute

Entity	
Pimary identifier attribute	Attribute name <pi> <M>
Alternate identifier attribute	Attribute name <ai> <M>
Mandatory attribute	Attribute name <M>
Optional attribute	Attribute name

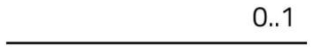
Crow’s Foot Notation : Relation

Relationship	
Optional symbol	
One symbol	
Many symbol	
Zero or one	
Only one	
Zero or more	
One or more	

UML Class Diagram : Class & Attribute

Entity	
Attribute (mandatory)	Attribute name [1]
Pimary identifier attribute	Attribute name {id}
Alternate identifier attribute	«ai» Attribute name [1]
Multi-valued attribute	Attribute name [x..y]
Derived attribute	/Attribute name

UML Class Diagram : Relation

Relationship	
Optional symbol	0
One symbol	1
Many symbol	*
Zero or one	
Only one	
Zero or more	
One or more	

Key constraints

- Key
  - Previous lecture, something that can identify a record
  - There are several candidates, the winner is primary key
  - The rest can be candidate keys
- Types of keys
  - Super key – any combination that can identify a record
  - Primary key (PK) – a selected set of minimal attributes that can identify a record
  - Candidate Key – a set of attributes that can identify a record but is not a PK
  - Foreign Key (FK) – a link from a table to PK of the other shows relationship
  - Composite Key – a key that composed from more than one attributes

steps for drawing an ERD

Gather Requirements: Identify data needs and rules.

Identify Entities: List main objects (e.g., Customers, Orders).

Add Attributes: Define key details for each entity.

Set Relationships: Connect entities and specify cardinality.

Add Keys: Assign primary and foreign keys. Determine cardinality and modality.

Draw Diagram: Use appropriate notation (Chen or Crow’s Foot).

Review: Validate with stakeholders.

Practice Makes Perfect

Before you leap from the plane!

- Normalization
- Real-world constraint
- Naming Conventions

Tools

- Miro
- Canva
- pgAdmin 4
- pgModeler
- DBeaver

Case studies (in Relational Schemas)

Artists(ArtistId, Name)
Albums(AlbumId, Title, ArtistId)
Tracks(TrackId, Name, AlbumId, MediaTypeId, GenreId, Composer, Milliseconds, Bytes, UnitPrice)
Genres(GenreId, Name)
MediaTypes(MediaTypeId, Name)
Playlists(PlaylistId, Name)
PlaylistTrack(PlaylistId, TrackId)
Customers(CustomerId, FirstName, LastName, Company, Address, City, State, Country, PostalCode, Phone, Fax, Email, SupportRepId)
Invoices(InvoiceId, CustomerId, InvoiceDate, BillingAddress, BillingCity, BillingState, BillingCountry, BillingPostalCode, Total)
InvoiceLines(InvoiceLineId, InvoiceId, TrackId, UnitPrice, Quantity)

Pair them up

Artists and Albums

Albums and Tracks

Tracks and Genres

Tracks and Media

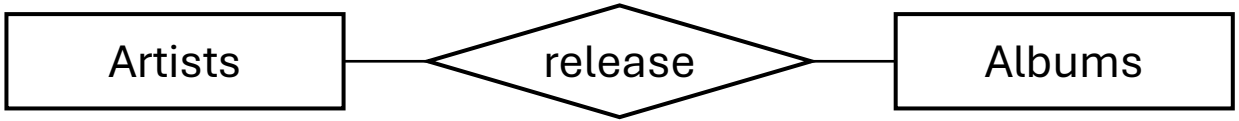
Tracks and invoices and Customers

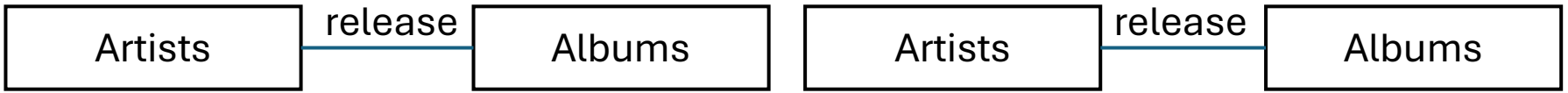
invoices and Customers

Tracks and Customers

Chen/Crow's Foot/Class Diagram

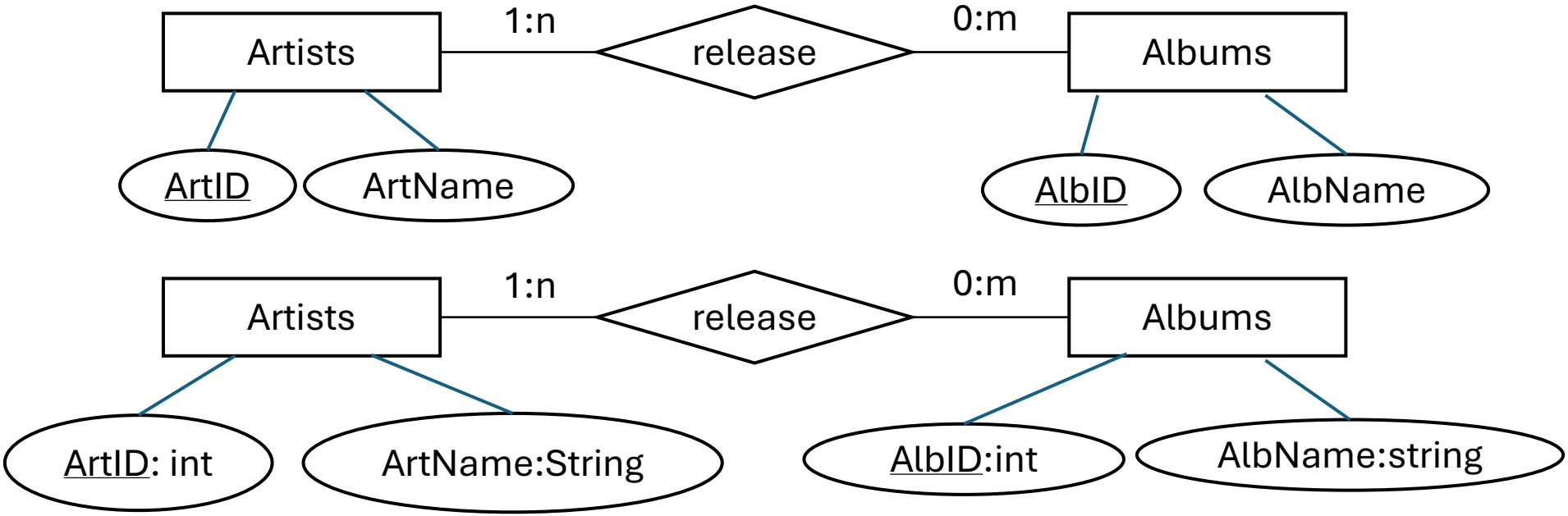
Artists and Albums





Chen ERD

Artists and Albums



Crow’s Foot

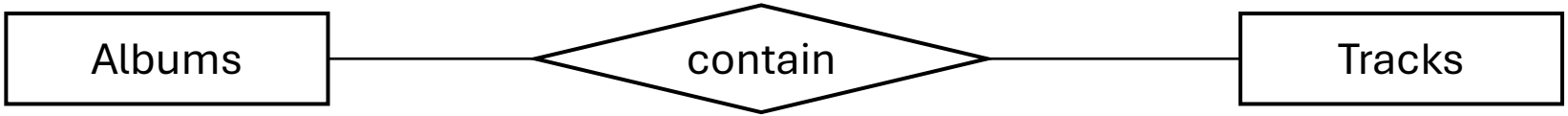
Artists and Albums

Class Diagram

Artists and Albums

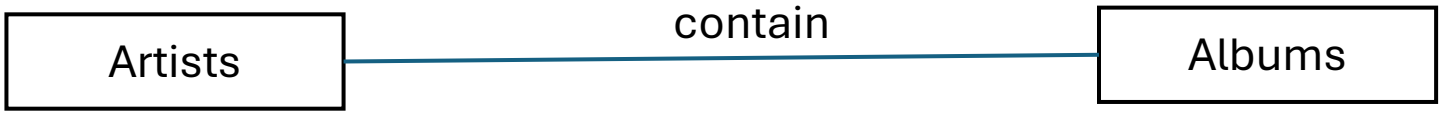
Chen

Albums and Tracks

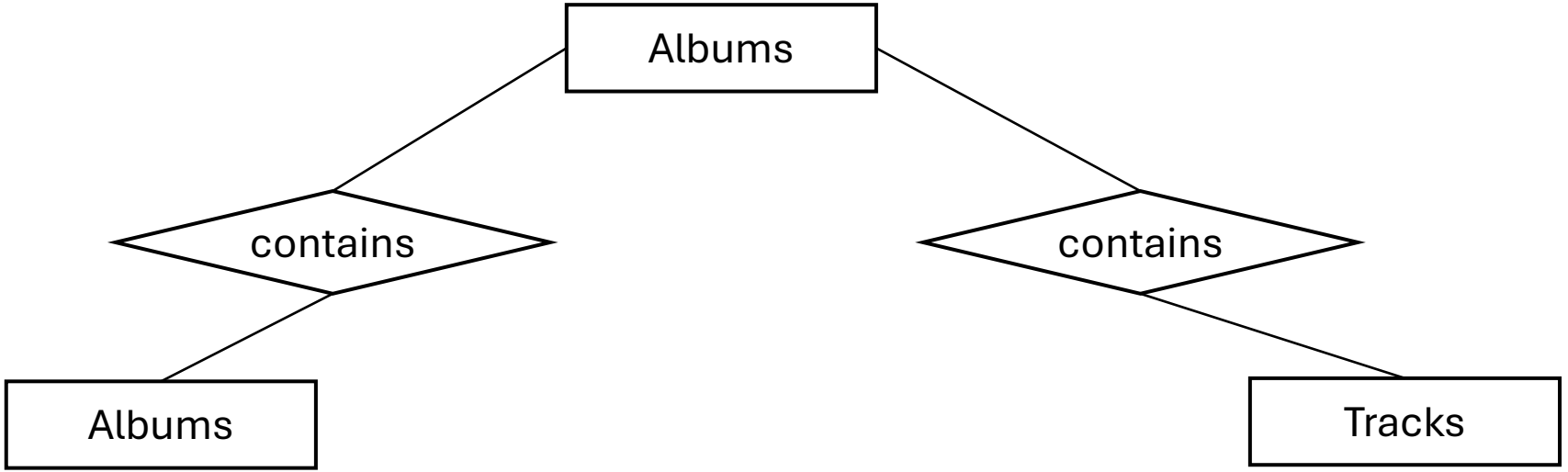


Crow’s Foot

Albums and Tracks



Chen: Chaining



Crow’s Foot: Chaining

Let’s take a break

More resources

https://www.gleek.io/blog/crows-foot-chen.html

https://app.sophia.org/tutorials/entity-relationship-model

bytes.usc.edu/cs585/s24-d-a-t-aaa/lectures/DataModeling/more/notations.pdf

www.inf.usi.ch *****

www.geeksforgeeks.org/introduction-of-er-model/

www.lucidchart.com/pages/er-diagrams

www.visual-paradigm.com/guide/data-modeling/what-is-entity-relationship-diagram/

ERD Generator

www.softwareideas.net

https://medium.com/@ericgcc/ *****

SQL Again!

SQL statement types

- **DDL** – Data Definition Language
- **DQL** – Data Query Language
- **DML** – Data Manipulation Language
- **DCL** – Data Control Language
- **TCL** – Transaction Control Language

www.geeksforgeeks.org

Create Table

With key constraint

- PK
- FK

Other constraint

- Not Null
- Unique
- Default

students(**sID**, fname, name, DoB)
courses(**cID**, cName)

Check yourselves!
If we want to keep grades of students that register a course, what is schema of grades table?

Examples : Primary Key

```
create table students (
        sID         number(8) primary key,
        fname       varchar(20),
        lname       varchar(30) unique,
        DOB         date not null
);
```

```
create table courses (
        cID         char(6),
        cName       varchar(20) default 'john',
        primary key (cID)
);
```

```
CONSTRAINT UC_ST1 UNIQUE (fname,lname)
```

Examples : PK (Multi-Attribute)+ FK (sID) + FK(cID)

```
create grades (
        sID     number(8) references students(sID),
        cID     char(6) references courses(cID),
        grade varchar(2),
        primary key (sID,cID)
);
```

INSERT INTO

Same as before

```
INSERT INTO table_name (column1, column2, column3,etc)
    VALUES (value1, value2, value3, etc);
```

```
INSERT INTO students ( sID, fname, lname) values ( 1, 'john','Doe');
```

Select

```
SELECT [distinct] column1,column2,etc
FROM table_name
WHERE column1 = value1 and
        column2 = value2 or
        etc = value_etc;
SELECT fname,lname FROM students
    where sID = 1 and ( fname = 'John' or lname = 'Doe');
```

DELETE

DELETE FROM table_name WHERE condition;

```
DELETE FROM students
    where sID = 1 and ( fname = 'John' or lname = 'Doe');
```

DROP/TRUNCATE

DROP TABLE table_name;

TRUNCATE TABLE table_name;

Describe

desc[cribe] table_name;

Alter table

ALTER TABLE table_name

ADD column_name datatype;

DROP COLUMN column_name;

RENAME COLUMN old_name to new_name;

RENAME TO new_table_name;

ADD PRIMARY KEY (keys);

DROP PRIMARY KEY;

ADD FOREIGN KEY (column1) REFERENCES table2(column2);

*** there are so many varieties to write this alter statement, please check out various sites for completeness

Examples: ALTER TABLE

ALTER TABLE students MODIFY sID number(8) primary key;

ALTER TABLE students primary key (sID);

ALTER TABLE grades FOREIGN KEY (sID) references students(sID);

ALTER TABLE grades ADD register_date date not null;

Summary

ERD notation

- Chen and Crow’s foot
- UML class diagram

More advanced SQL

- where clause
- alter table

GOOGLE is your only hope!